A Biodiversity Approach to Cyber Security

by

Jennifer Tracy Jackson

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Centre for Complexity Science

May 2017

Contents

Figures

Tables

Acknowledgements

I thank my supervisors Mark Leeson and Sadie Creese for their support in allowing me to pursue my own research subject and directions. A particular thanks to Mark who provided words of encouragement and support during difficult times.

Many thanks to all those that run the Centre for Complexity Science who, have not only given me the opportunity to undertake this research, but have also been patient in awaiting its completion.

Thank you to my husband for his unconditional support, and without whom, I would not have been able to undertake this quest. Also I would like to acknowledge his assistance in creating a suitably formatted word document thesis template.

Thank you to my Mum and Dad for always being there and having faith in everything I do.

Finally thank you to my three amazing children who did not exist when I started, but have somehow kept me sane.

Declaration

I declare that this thesis and the work presented within is my own except where explicitly acknowledged. This thesis is submitted for the degree of Doctor of Philosophy from the University of Warwick, United Kingdom. No part of the work presented in this thesis has been submitted in support of an application for another degree or qualification of this, or any other, university or institute of learning.

Some aspects are adapted from reports or publications written by the author during the course of this work as follows:

Parts of the text within chapter 3 (§3.4), and parts of the text and general concepts within chapter 5 have been adapted from the following publication:

(1) J. Jackson, S. Creese, and M. S. Leeson, "Biodiversity: A security approach for ad hoc networks," in IEEE Symposium on Computational Intelligence in Cyber Security, Paris, France, 2011 [1].

Parts of the text within chapter 3 (§3.3.4) have been taken from a report published on-line within the Warwick Research Archive Portal (WRAP):

(2) Jackson, Jennifer (published 2017, created 2011) "Multi-scale location analysis of vulnerabilities and their link to disturbances within digital ecosystems". Coventry: University of Warwick, Warwick Research Archive Portal. http://wrap.warwick.ac.uk/86134/ [2]

A number of references have been made to an epidemic malware model that was developed during a prior Complexity Science MSc, for which the design has not been included as part of this thesis. However further simulation work investigating realistic scenarios using the model was conducted during the PhD period and the resulting journal paper has been listed here as a contribution to the research. The work is referenced within chapter 4 (§4.3) and some research aspects regarding antivirus response times, malware transmission times, and Bluetooth transmission characteristics are used during simulation of results within chapter 8 (sections 8.2.2.5 and 8.3.6):

(3) J. T. Jackson, and S. Creese, "Virus propagation in heterogeneous bluetooth networks with human behaviors," IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 6, 2012 [3].

Jennifer Tracy Jackson

Preface

The Centre for Complexity Science seeks to develop the knowledge to understand, control and design complex systems, providing break-throughs in new applications of complexity science with solutions for society and real-world problems.

The Centre incorporates the earlier EPSRC Doctoral Training Centre in Complexity Science and now forms part of the wider MathSys Centre for Doctoral Training within the Faculty of Science at the University of Warwick.

This document is a thesis following a period of Ph.D. research. The work was supported through funding from the Engineering and Physical Sciences Research Council (EPSRC).

The text of this thesis was prepared using Microsoft Word 2007 and set in 11pt Palatino Linotype. The figures and tables were produced using Microsoft Visio 2007, and Microsoft Excel 2007. The references were managed by Thomson ISI ResearchSoft EndNote X2.0.4.

Pages:	372
Total Words:	76934
Excluding:	-6962
Qualifying:	69972

Qualifying word count excludes Appendix, Abbreviations, Trademarks and References.

Original submission: May 2017
Final submission: May 2018

Abstract

Cyber crime is a significant threat to modern society that will continue to grow as technology is integrated further into our lives. Cyber attackers can exploit vulnerabilities to access computing systems and propagate malware. Of growing concern is the use of multiple exploits across layers of the software stack, plus faster criminal response times to newly disclosed vulnerabilities creating surges in attacks before signature-based malware protection can take effect. The wide scale adoption of few software systems fuels the problem, allowing identical vulnerabilities to be exploited across networks to maximise infection in a single attack. This requires new perspectives to tackle the threat. Biodiversity is critical in the functioning of healthy ecosystems. Whilst the idea of diversity benefiting computer security is not new, there are still gaps in understanding its advantages.

A mathematical and an agent-based model have been developed using the ecosystem as a framework. Biodiversity is generated by individualised software stacks defined as genotypes with multiple loci. The models allow the protection offered by diversity to be quantified for ad hoc networks which are expected to become prevalent in the future by specifying how much diversity is needed to tolerate or mitigate two abstract representations of malware encompassing different ways multiple exploits target software stack layers. Outputs include the key components of ecosystem stability: resistance and resilience. Results show that diversity by itself can reduce susceptibility, increase resistance, and increase the time taken for malware to spread, thereby allowing networks to tolerate malware and maintain Quality of Service. When dynamic diversity is used as part of a multi-layered defence strategy with additional mechanisms such as blacklisting, virtualisation, and recovery through patching and signature based protection, diversity becomes more effective since the power of dynamic software updating can be utilised to mitigate attacks whilst maintaining network operations.

Chapter 1

Introduction

This chapter establishes the motivation, hypothesis, research contributions, and structure for the work presented within this thesis. The motivation and rationale behind the consideration of biodiversity in the context of cyber security stems from both a) the impact that wide-scale cyber attacks such as those caused by malware can have when systems use the same non-diverse software or underlying technology, and b) the benefit biodiversity can have within a natural ecosystem in providing resistance against attack from disease and pests.

Included is an introduction to the concept of *biodiversity for cyber security* through several motivating factors. These include an ever-changing cyber threat landscape fuelled by advancements in technology, risks associated with computing monocultures, and the range of benefits provided by biodiversity within natural systems. Gaps in current research knowledge are highlighted together with an emphasis on wireless mobile computing such as ad hoc networks which are predicted to become more prevalent in the future. Understanding the benefits and mechanisms of biodiversity underlying natural systems and applying them to this digital wireless domain may enhance cyber security against such malware attacks.

1.1 Motivation

1.1.1 The Changing Cyber Threat Landscape

One of the biggest security problems modern society currently faces is the growing threat from cyber attacks. Cybercrime is estimated to cost the global economy US$575 billion annually [4], and maintaining an adequate level of security is a co-evolving process between improved defensive techniques and ever more sophisticated attack methods. Advancements in technology fuel this process but also simultaneously change the threat landscape. The world purchased more than 1.4 billion smartphones in 2015 [5] and it is predicted there will be 50 to 200 billion total connected devices by 2020 [6] [7] [8]. This has the potential for them to be integrated into every aspect of our lives creating an attractive target for online criminals. Cyber attackers exploit vulnerabilities within the software, firmware or underlying fabric of the devices, as well as the user to gain access to important data, deny the use of services, spy, control systems, spread viruses, and sometimes cause irreversible damage. Worryingly it has been estimated that up to 70% of attacks go undetected [4]. Most software programs have vulnerabilities and since it is difficult to remove all vulnerabilities, the problem is likely to become worse as the use of wireless supported mobile computing and the Internet of Things (IoT) continues to grow and change the threat landscape.

1.1.2 The Risks of Computing Monocultures

The increased use of computing devices and wide scale adoption of few operating systems (OS) and common protocols continues to pose a significant threat. Computing monocultures refers to the widespread use of the same hardware, firmware, or software. Although different patching habits of individuals can create some level of diversity between devices using the same firmware or software, the diversity is restricted to the locality and functionality of the patch. Similarly different versions of the same software, for example different versions of operating systems, may utilise much of the same underlying libraries. Therefore large commonality of code described in this way adds to the monoculture argument. For example of the 1.4 billion smartphones purchased in 2015 98 percent were dominated by two operating systems: five out of six ran the Android OS, and one in seven ran Apple's iOS [5]. This made Android devices the most targeted by attackers [9] [10] [5], a trend that has been on-going for several years [11]. A similar scenario is seen with desktop personal computers (PCs) where Microsoft Windows dominates the OS market, and it is predicted that over the next five years leading-edge IoT devices will experience the same scenario [7]. Having a small number of different operating systems or application software is more economically efficient because of the ease of maintenance and compatibility. It also has greater user appeal because of the need to learn only a few different types of applications and systems. However, much effort is spent protecting the resultant computer networks from attacks and malware, which in some cases can spread to a large number of devices in a matter of minutes [12] [13]. In 2015 Symantec reported that "Attackers Are Moving Faster, Defenses Are Not" [14] in response to attackers exploiting zero-day (publically unknown) vulnerabilities much faster than vendors could create and roll out patches. Patch times can range from a day [5] to several months [14], however even generating patches within a few hours may not be fast enough to stop the short term spread. Additionally, it has been reported that zero-day attacks can last up to

30 months before the vulnerability is even disclosed [15]. The number of new mobile vulnerabilities being discovered is increasing every year, with the Common Vulnerabilities and Exposures (CVE) database reported that vulnerabilities with Android in 2016 were estimated to be twice that of 2015 (131) [16]. Current security solutions for mobile devices remain limited in their ability to protect, particularly against zero-day attacks, with manufacturers being slow to address fundamental security issues for IoT devices. Additionally it is predicted that over the next five years attackers will not just be targeting applications and operating systems but will look for additional vulnerabilities at layers lower down the software stack independent of operating systems [7] such as low level drivers and protocols. The use of multiple exploits (code or data directed at a specific vulnerability) across layers of the software stack will pose a significant threat, especially if they are targeting zero-day vulnerabilities. The 2010 Stuxnet worm for example is known to have used four separate zero-day exploits [17].

The risk associated with software monocultures has long been recognised within the computing industry [18] [19], however the physical technology and infrastructure to produce, and maintain alternative versions of software is only now becoming possible [20]. As the number of devices and vulnerabilities grow, traditional security methods will become less effective. To keep up with the sophistication of attack methods there will need to be greater automation of defences and new paradigms of defence mechanisms including those to alleviate the monoculture risks.

1.1.3 Agricultural Monocultures and Biodiversity within Natural Systems

The risks of monocultures are well known within the agricultural industry which has experienced the resultant problems first hand. A single species is often selected for its productivity or disease resistance properties and is grown over a large area for economic efficiency [21]. But this efficiency creates risks: Land cultivated in this way removes the naturally diverse communities, reduces the soil quality, and results in the need for fertilizers to protect crops from pests and diseases.

The range of plants, animals, insects and other organisms living within an ecosystem is termed biodiversity. Biodiversity is linked to the stability and productivity of ecosystems buffering them from pest invasions, disease epidemics and extreme environmental events [22]. Biodiversity is also critical to the functioning of such ecosystems and the services they provide. The agricultural industry is now becoming more appreciative of the essential benefits biodiversity brings and is slowly changing its habits through modernisation of traditional methods such as crop diversification and crop rotation [23] to help reduce infestations of pests in the soil. The benefit of biodiversity has also been evinced in other areas of natural systems. It has been shown to reduce the spread of diseases between animals such as Lyme disease [24], and the hantavirus affecting deer mice [25]. High levels of biodiversity have also been found to increase resistance against extreme climate events, which are now becoming more frequent world-wide [26] [27].

1.2 Hypothesis

The intuition that diversity might be desirable has existed within the security profession for many years. In the 1970s N-version programming [28] was proposed within the field of fault tolerance to increase the reliability of systems that used software. It was known that identical software running on independent systems would fail in exactly the same way with the same inputs. Interest in this approach as a security mechanism grew as computers became ubiquitous, attacks became more common, and the risks of a software monoculture was acknowledged [19] [18]. A biological perspective on diversity as a security mechanism however has largely been overlooked and requires an understanding of ecological processes and interactions, and their effects on the system [29]. Current research is mainly focused at point solutions for creating diverse software [30] [31] [32] [33] [34], although there has been some work on creating diverse networks [35] [36] [37] and measuring diversity within networks [38] [39] [40]. Despite the recently growing research in this area there is still a large gap in understanding the actual benefits of diversity as a security mechanism, particularly from an ecological perspective, even whilst evidence surrounding the benefits of biodiversity in natural systems is continuously growing.

It is expected that peer-to-peer wireless networks such as ad hoc networks will become more mainstream than they are currently. This drive will be as a result of billions more connected devices such as through the evolving IoT [8], and developing protocols such as fifth generation (5G), which supports direct device to device communication [41]. Such topologies are decentralised, rely on physical locality to form local communication links, and may change due to the mobility of devices.

Creating diversity of software to the benefit of security within such topologies has largely been unexplored, let alone from an ecological perspective. There are similarities between peer-to-peer mobile wireless networks and natural communities due to their movement and short range communication patterns making them a good candidate for studying the effects of biodiversity as a security mechanism. Additionally, the modelling of multi-exploit malware propagation targeting vulnerabilities across layers of a software stack has so far been neglected in the literature.

The focus of this research combines these two domains where the hypothesis for this work is therefore:

"Incorporating biodiversity within peer-to-peer mobile wireless computer networks makes them more resistant to multi-exploit malware propagation."

1.3 Contributions to Research

The original and significant contributions of this thesis are:

- *Definition of an Ecosystem model of an ad hoc network* (§5).

  Aspects published in the conference proceedings of the IEEE Symposium on Computational Intelligence in Cyber Security, 2011 [1]. The model proposes that by applying biodiversity strategies at different scales of a network, the destructive effects arising from security attacks can be counterbalanced with the constructive effects of biodiversity to maintain ecosystem function and services, and hence benefit overall resistance and resilience.

- *Modelling of multi-layer multi-exploit malware within diverse computing systems which includes* (§6, §7)*:*
    - Representations, including *analytical*, of malware types with multiple exploits targeting multiple software layers with two different (logical AND and OR) relationships (§6).
    - *Genetic matching* of malware types to devices forms part of the novel approach of simulating malware propagation in diverse computing devices (§6, §7).

The representations allow the *susceptibility* of a network to be determined, and allow *simulation* of such malware in a network where the diversity remains *static* (§6, §7) (the software on each device remains fixed during the simulation scenario) or is *dynamic* (the software on each device can change during the simulation according to the rules of the diversity algorithm) (§7).

- *Definition of metrics to measure the diversity of any computing network* (§5, §6, §7).

Single measures of diversity in computing systems have been defined in the literature; however several metrics are necessary to define diversity of multi-layer software stacks across a network, including those to define the software stack granularity, the number of different software, their distribution, and their structural composition. A genetic approach is used where several definitions from ecology have been adopted. The Nei genetic diversity index [42], which has not been used previously, has been adopted in its monoploid form to measure the distribution of different software. It is very rarely stressed in the literature that it can be applied to any number of chromosome sets since most studies focus on diploid chromosomes of animals and plants. It is used here to measure the global performance of the dynamic diversity algorithm and calculate theoretical maximum diversity values for a given network configuration.

- *Development of a mathematical epidemic model which includes* (§6):
    - Enhancements to the compartmental (applicable to both deterministic and stochastic) SI/SIR models to *incorporate malware types with multiple exploits across multiple software layers* in a wireless peer-to-peer ad hoc network where the diversity remains *static.*
    - A *method* has been developed to *calculate optimum amounts of diversity* necessary to *tolerate or mitigate different types of multi-exploit, multi-layer malware.*
    - *Ecosystem outputs* including resistance and resilience.

Note that enhancement of the SIR model to incorporate static diversity for an exploit targeting only single software configurations has already been proposed in the literature [43].

- Development of an *agent-based simulation framework* within the Mathworks Matlab environment to understand how biodiversity can make wireless peer-to-peer computer networks more resistant to malware (§7). The source code for the model is available at the permanent link : http://wrap.warwick.ac.uk/98458.

The simulation framework allows the diversity of networks to either remain static, or be modified dynamically. It allows for testing, simulating and experimenting with different diversity algorithms, networks, attacks, and additional security mechanisms to prove and explore the hypothesis. The simulation framework incorporates the following aspects:

- o A *mobility model* controlling how and when individual devices communicate with one another. The following standard models have been used:
    - Uniformly distributed random encounter.
    - Random Waypoint which has been *further developed* to model the selection of devices to form a communication link with and the successful data transmission.
- o A *diversity model* controlling what software is installed on each device and when. Within this a *dynamic diversity algorithm has been developed based upon local information.* The algorithm can incorporate optional security mechanisms to enhance the effectiveness of diversity, and constraints that may limit the diversity achievable.

    - A *malware model* to inject malware into the network at a predefined time and monitor the health of each device as the simulation progresses. The SI/SIR compartments have been used.
    - *Metrics* including biodiversity levels as the simulation progresses, and *ecosystem outputs* including resistance and resilience.

Further contributions of note include:

- *A comprehensive review covering how biodiversity works in nature* and where lessons can be learned and applied to ad hoc networks (§2, §5, and work published in [1]).
- *A comprehensive review of current research associated with diversity as a security mechanism* (§3.4).
- *A comprehensive review of the location of vulnerabilities* at different scales of an ad hoc network and their link to undesirable security events (disturbances) (§3). A self contained study is published online [2].
- *Simulations of malware propagation with different spreading mechanisms in Bluetooth peer-to-peer networks.* Published aspects included within the thesis are documented within the Declaration. Work published in the journal paper [3].

1.4 Thesis Structure

This chapter has given an introduction to the concept of biodiversity for cyber security. The next three chapters provide a comprehensive background that directly supports the work in the remainder of this thesis. Chapter 2 details the link between biodiversity and ecosystems in natural systems, and how biodiversity is critical to the functioning of such ecosystems and the services they provide. Chapter 3 explores computer security in detail, the extent of current diversity research, and the enabling technologies that may allow the diversity of computing and software possible. Chapter 4 details methods of modelling mobile networks, malware and epidemiology. Chapter 5 draws on the background material and presents an ecosystem model of an ad hoc network. Chapters 6 and 7 present the two different diverse system models developed and Chapter 8 details their results and analysis. Chapter 9 draws together the conclusions by summarising the work presented, and providing ideas for future work.

Figure 1-1 provides a graphical representation of the thesis structure:

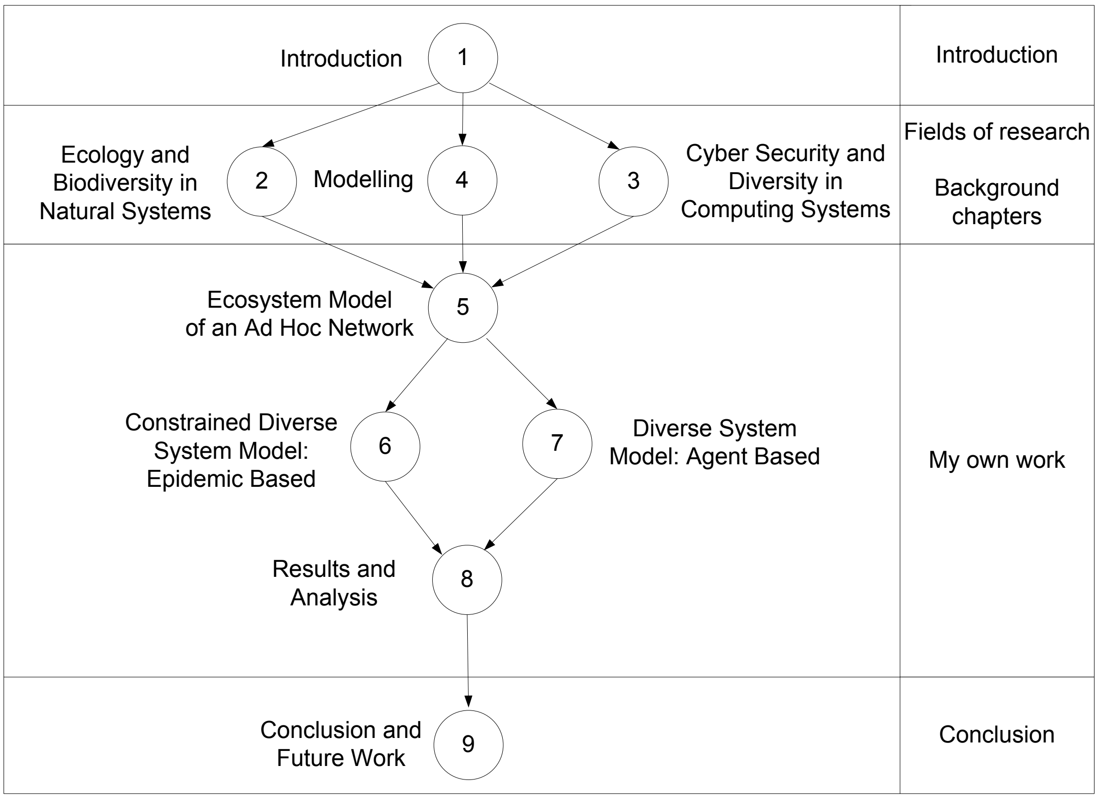

Introduction
1
Introduction
Ecology and Biodiversity in Natural Systems
2
Modelling
4
3
Cyber Security and Diversity in Computing Systems
Fields of research
Background chapters
Ecosystem Model of an Ad Hoc Network
5
Constrained Diverse System Model: Epidemic Based
6
7
Diverse System Model: Agent Based
My own work
Results and Analysis
8
Conclusion and Future Work
9
Conclusion

Figure 1-1 – Thesis structure

1.5 Summary

This introduction has established the concept of biodiversity for cyber security. The focus of the biodiversity inspired security research is wireless peer-to-peer mobile networks since they are predicted to become prevalent in the future computing market. The hypothesis given for this work is that incorporating biodiversity within peer-to-peer mobile wireless computer networks makes them more resistant to multi-exploit malware propagation. The final sections of this chapter outlined the contributions made by this work and the structure of the thesis.

Chapter 2

Ecology and Biodiversity in Natural Systems

2.1 Introduction

Understanding biodiversity from an ecological perspective, its relationships, and how its effectiveness is measured against external inputs is important for considering analogous relationships and measures of diversity within mobile wireless peer-to-peer networks and its effectiveness against malware. This chapter is split into two sections:

The Biodiversity and Ecosystem Relationship: The first section discusses biodiversity and its relationship with other components of an ecosystem. It discusses how biodiversity links to ecosystem functionality, how biodiversity is affected by external disturbances, and how the effect of biodiversity on limiting the severity of disturbances is measured.

Measuring Biodiversity: The second section details the metrics for measuring biodiversity at the genetic level only, which are referenced during later chapters of the thesis.

2.2 The Biodiversity and Ecosystem Relationship

2.2.1 Biodiversity within Ecosystems

An *ecosystem* is comprised of interacting organisms such as plants, animals, insects etc and their physical environment. The global behaviour of an ecosystem is the result of *local peer-to-peer interactions* of such organisms and with their environment resulting in *distributed* (sharing of tasks), *self-organising* (global coordination from local interactions), and *emergent* (collective behaviour or property) properties. *Biodiversity* encompasses the variety of genes, species, or functional traits within an ecosystem and is critical to the *functioning* of such ecosystems and the emergent *services* they provide. *External influences* can impact on biodiversity and function and affect these services. Ecosystem health, and in particular its relationship with biodiversity, is often assessed by looking at the outputs of ecosystem functions and services where *productivity*, *stability,* and *disease transmission* are measures often used within field studies and theoretical models.

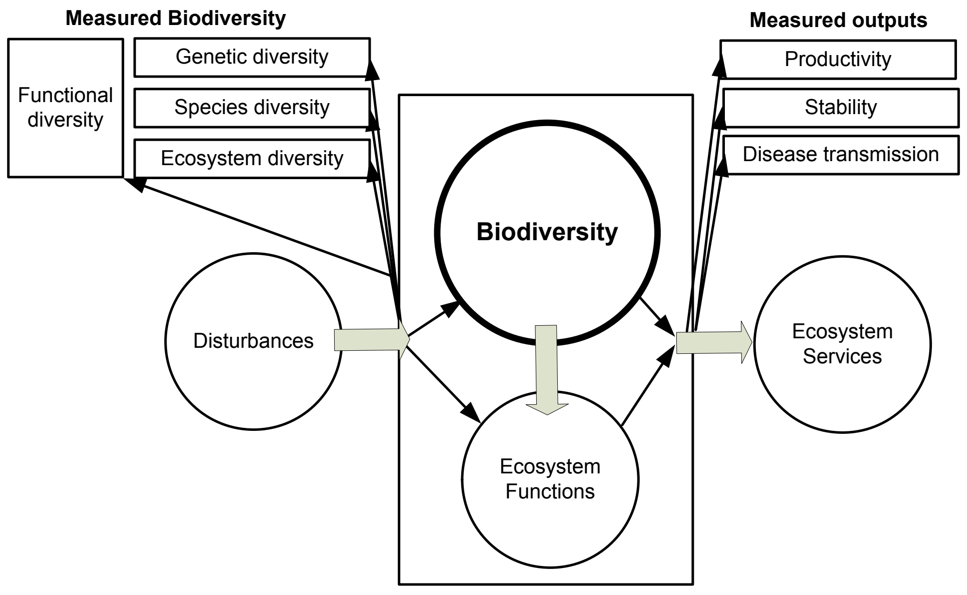


Figure 2-1 – Biodiversity and ecosystem relationship

2.2.1.1 *Biodiversity*

There are generally three levels of biodiversity defined in the literature: *genetic diversity*, *species diversity*, and *ecosystem diversity*. Most theoretical and experimental studies focus on the species level when considering ecological consequences of biodiversity because it is easier to work with and measure [22], however biodiversity is hierarchical and over the past decade there has been a steadily growing interest in the genetic level, with research suggesting that genetic diversity can also have significant effects on ecological processes [44]. In addition to these three levels another dimension of diversity is often discussed, especially in relation to ecosystem function, and that is *functional diversity*. This encompasses functional traits at all three levels of diversity but research is again often focused at the species level.

(1) Genetic diversity is the variety of differences between the genetic makeup of individuals. It is often measured within species at an individual scale but does not necessarily have to be limited to that. Genotypes determine the actual set of genes carried by an individual and phenotypes are the observable characteristics and traits coded for by those genes.

(2) Species diversity is usually measured within a geographical region or ecosystem at a community scale by quantifying the number of different species and their distribution. It is different to genetic diversity in that groups of individuals with the same characteristics are divided into distinct groups which are usually well known and documented. The classification of species is usually via a taxonomy approach using a hierarchical branching structure with various kingdoms defining the top level, such as the *animal kingdom*.

(3) Ecosystem diversity includes the measurement of diversity of communities, geographical regions or complete ecosystems. For example species diversity can be measured at and between different scales of geographical areas [45].

(4) Functional diversity is about differences in functional traits. Ecosystem function depends on functional diversity more than on the number of different species alone. For example, species may have the same role creating redundancy but low functionality; alternatively, species may have different roles creating low redundancy but high functionality.

2.2.1.2 Ecosystem Functions

Ecosystem functions are the ecological processes that take place within an ecosystem as a result of environmental factors and individual functionality, in particular the interaction of the individual with others and the environment. They have been categorised in different ways such as in terms of material, energy and information flow [46], or broken down into categories such as *regulating functions* (e.g. water and nutrient regulation, pollination), *supporting functions* (e.g. soil formation such as chemical weathering of rocks), and *provisioning functions* (e.g. raw materials such as biomass and plant production) [47]. Biodiversity has a strong influence over ecosystem function and is discussed further in section 2.2.2.1.

2.2.1.3 Ecosystem Services

Ecosystem services are the benefits that ecosystems provide to humanity and are derived from the many functions operating within an ecosystem. They are of a particular concern to ecologists since their demise or loss can be devastating [48] [49] [50]. Services can also be broken down into *regulating services* (e.g. air and water quality, buffering against extreme natural events such as drought, controlling pests and diseases), *provisioning services* (e.g. food products such as fish, crops and livestock, water, fuels such as wood and gas) [50], and sometimes additionally *cultural services* (e.g. providing iconic landscapes and recreational opportunities) [47] and further *supporting services* (e.g. crop pollination) [46] [51].

2.2.1.4 Disturbances

Disturbances are influences on an ecosystem which can be both natural and artificial such as rain or human interaction, and can also be severe such as a flood or a drought. Disturbances can impact biodiversity which in turn affects functions and services. There are two aspects of disturbances: *disturbance events* and *the natural disturbance regime* [52].

(1) A disturbance event is an incident that disrupts an ecosystem usually over a relatively short period of time. Disruptions can include the spread of a disease, changes in the physical environment or resources.

(2) The natural disturbance regime shapes an ecosystem over long time scales and includes many disturbances with varying intensities at different spatial and temporal scales such as changing temperatures and seasons [53] [54] [55]. This generates natural levels of biodiversity by varying the conditions in which different species can operate.

2.2.1.5 Measured Outputs

Productivity and *stability (*including resistance and resilience), are often measured to assess the output of an ecosystem and how well it can cope with the effects of disturbances and changes in biodiversity. Often in the literature function and productivity are grouped together. For example the function of producing biomass often leads to assessing biomass productivity. Stability is about assessing how well the ecosystem can cope under different scenarios, such as how the productivity changes and how quickly the ecosystem recovers from a disturbance like a disease epidemic [56]. Productivity, stability and disease transmission are discussed further in section 2.2.2.2.

2.2.1.6 Relationships

There are relationships between disturbance, biodiversity, and ecosystem function. Within the literature some studies focus on just disturbance and its effect on species diversity [57] [53] [58] [59] [60] or genetic diversity [61] [62] [63], some consider the relationship between biodiversity and ecosystem function (§2.2.2.1) [64] [22] [65] [46], whilst others consider the effects of disturbance severity on the measured outputs as a result of species [66] [67] [27] or genetic [68] [44] [69] diversity (§2.2.2.2). The latter two relationships are discussed further in the next section since these both consider biodiversity as a controlling mechanism on the output of an ecosystem.

2.2.2 Biodiversity Relationships

2.2.2.1 Biodiversity Mechanisms Underlying Ecosystem Function

Biodiversity has a large influence on ecosystem function and since the measured outputs of an ecosystem are based upon the productivity and stability of functions and services, this section details the mechanisms (as pictured in Figure 2-2) that link biodiversity to ecosystem function. This includes *Niche differentiation* (§2.2.2.1.1), *facilitation* (§2.2.2.1.2), *multiple trophic levels* (§2.2.2.1.3), and *genetic variation* (§2.2.2.1.4).

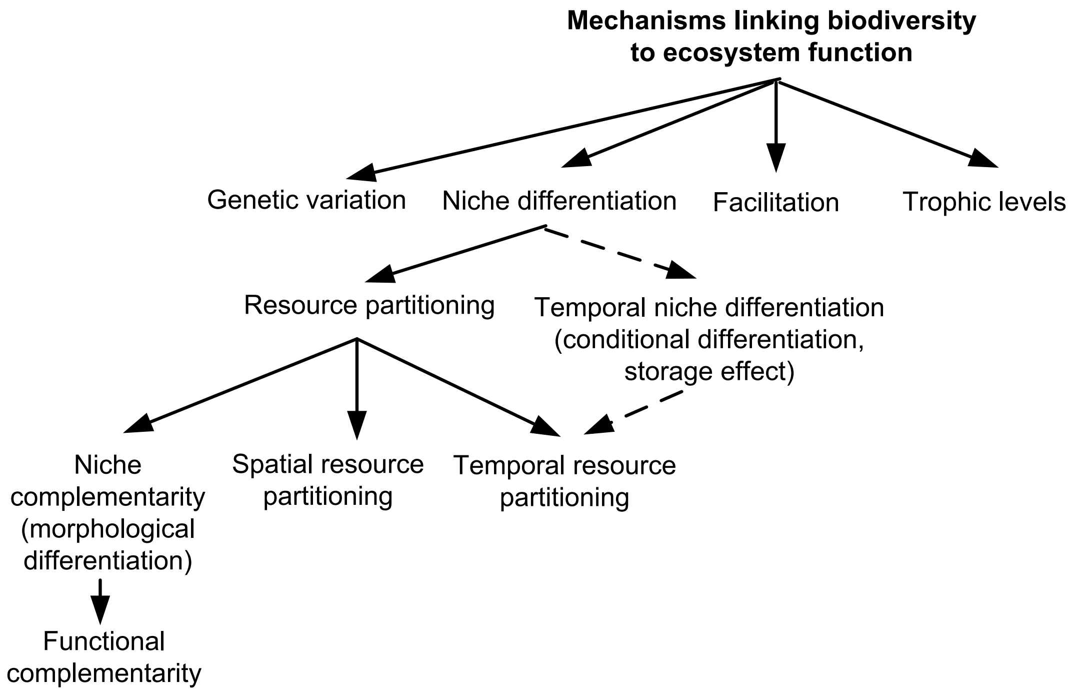

Mechanisms linking biodiversity to ecosystem function
Genetic variation
Niche differentiation
Facilitation
Trophic levels
Resource partitioning
Temporal niche differentiation (conditional differentiation, storage effect)
Niche complementarity (morphological differentiation)
Spatial resource partitioning
Temporal resource partitioning
Functional complementarity

Figure 2-2 – Biodiversity mechanisms linking ecosystem function

2.2.2.1.1 Niche Differentiation and Functional Complementarity

A niche is multidimensional and describes both the place and role in which an individual or species lives. The full range of possible conditions and resources that a species can occupy and use is called the *fundamental niche* (Figure 2-3 (a)) [70]. When a species interacts with another species there may be some overlap in one or more dimensions (Figure 2-3 (b)) creating competition of resources. The niche space then becomes restricted due to the competition (Figure 2-3 (c)) and this is called the *realised niche*. Niche overlap determines how strongly two species might compete with each other. If species are too similar the lesser competitor will either be excluded from an area or go extinct (Gause's exclusion principle) [71]. When species coexist, competition can drive them into different niches. This process is called *niche differentiation* of which there are several types (Figure 2-2). One of the most discussed is *resource partitioning* where species divide up a resource such as food at different places (*spatial resource partitioning*), at different times (*temporal resource partitioning*), or in different ways (*niche complementarity, or morphological differentiation)*. Often temporal resource partitioning is discussed as a separate form of niche differentiation and is referred to as *temporal niche differentiation*, *conditional differentiation* or *the storage effect* [72] where species have different competitive abilities under different environmental conditions. The mechanisms of niche differentiation is not just limited to species, niche complementarity has also been found during various genotypic diversity experiments [73] [69] [74].

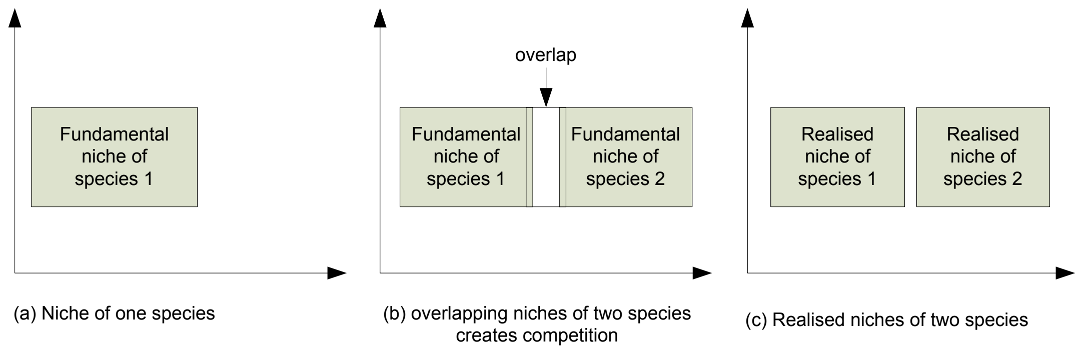

Fundamental niche of species 1
(a) Niche of one species
overlap
Fundamental niche of species 1
Fundamental niche of species 2
(b) overlapping niches of two species creates competition
Realised niche of species 1
Realised niche of species 2
(c) Realised niches of two species

Figure 2-3 – Fundamental and realised niches of coexisting species

Additionally, *functional complementarity* has been discussed in the literature as a specific type of niche complementarity where different species occupy different *functionally* distinct niches, benefiting ecosystem function [65] [22] and providing an important link between biodiversity and productivity [65] [75]. With negligible niche overlap, termed *perfect complementarity*, more of the total niche space is used, increasing ecosystem functioning but causing fragility due to the dependence on specific species. With large niche overlap there is large ecosystem function, but this quickly saturates as species diversity increases making them *functionally redundant* (Figure 2-4) [22]. Redundancy can improve the stability of the ecosystem if species are lost, but can competition between species when the resource is limited [65]. This suggests that both functional redundancy and functional complementarity are needed to benefit ecosystem services rather than just the number of different species.

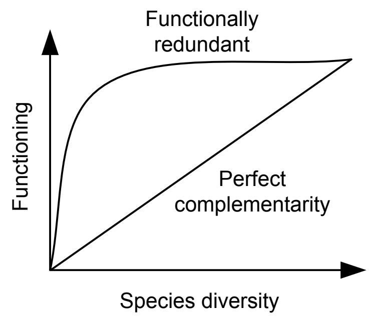


Figure 2-4 – Relationship between species diversity and ecosystem function [22]

Functional complementarity also occurs at the genetic level. Whilst genes can provide unique functionality, functional redundancy can also occur during the evolutionary process producing genes with overlapping functionality. The most common method is through direct gene duplication [76] caused by errors during

DNA replication such as through reproduction. Another method is through natural selection where previously dissimilar genes evolve to provide similar functionality through partial functional overlap [77].

2.2.2.1.2 *Facilitation*

Facilitation describes interactions between species or individuals, but can also apply at the genetic level [78] [79] [44], creating positive benefits for at least one without causing harm to the other. Facilitation can be either *mutual* where both species benefit, or *commensal* where only one species benefits. Increasing species diversity in the presence of facilitation can lead to increased ecosystem functioning [80] [81].

2.2.2.1.3 *Multiple Trophic Levels*

Many of the experimental studies have involved plant or microbial populations, often within a single *trophic level* (hierarchical level in an ecosystem such as the position in the food chain) [22] [82] [50] however it has been recognised that diversity across *multiple trophic levels* has the potential to impact ecosystem functions even more strongly [50]. The levels (Figure 2-5) consist of *primary producers,* at the bottom, followed by *primary consumers, secondary consumers,* and *tertiary consumers,* which consume species within the levels below them. There are also *decomposers* that break down dead or dying tissue from other species at different levels. The trophic pyramid however is often a very simplified picture of reality, where interactions between levels are very complex.

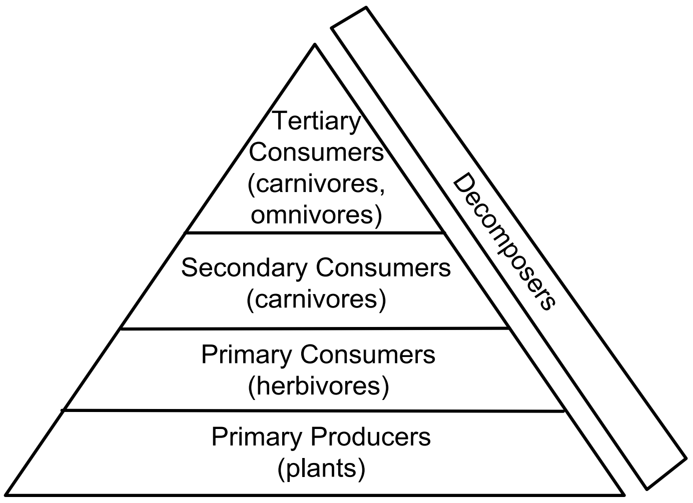

Tertiary Consumers (carnivores, omnivores)
Secondary Consumers (carnivores)
Primary Consumers (herbivores)
Primary Producers (plants)
Decomposers

Figure 2-5 – The trophic pyramid

2.2.2.1.4 Genetic Variation

Whilst there is evidence that the previous mechanisms are relevant at both the species and genetic levels [44], genetic variation appears only at the genetic level. Genetic variation is the variation of genes within a population and is the driving force behind functional differences between individuals and species. It is also a prominent component of evolutionary change and determines genetic diversity [83] [44] (§2.3.2).

Chromosomes are located within every cell but the number of sets can vary between species. There can be one set (*monoploid*), two sets (*diploid*), three sets (*triploid*) and more than three sets (*polyploid*). Animals and plants have two sets of chromosomes and are therefore diploid as shown in Figure 2-6. Each chromosome pair contains genes, representing short sections of DNA, which are located at a specific site called a *locus* [84].

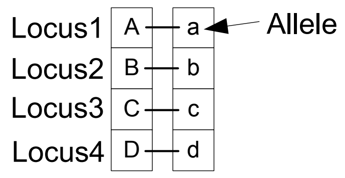


Figure 2-6 – Chromosome pair with multiple loci and alleles

Simplistically, loci determine traits or functions. Single genes can determine discrete traits such as eye colour, whereas the additive effect of multiple genes can determine continuous traits such as height. Genes may come in several different variants called *Alleles.* When both of the chromosome copies within the pair contain the same allele this is called *homozygous* and when they are different they are called *heterozygous*. A *genotype* represents the actual genes found within an individual's chromosome. Differences between alleles and genotypes, and their frequencies in a population, signifies the amount of genetic variation.

Genetic variation is caused by multiple factors. If two or more alleles coexist in the population at a specific locus, this is termed *genetic polymorphism*. Many species have genetic polymorphism at different loci [83]. *Reproduction* processes such as *crossover* (DNA exchange by parents) and *mutation* (random change, potentially creating a new allele) as well as the *migration* of individuals and *genetic drift* (occurrence of alleles randomly fluctuate over time) [85], can change the frequency and distribution of alleles, and introduce different combinations of genes leading to individualised genotypes.

2.2.2.2 *The affects of Biodiversity on Limiting Disturbance Severity*

There are two outputs of ecosystems that are commonly measured in relation to the effects of biodiversity on limiting disturbance severity. These are *productivity* and *stability* [64] [22] [86] [27] (see Figure 2-7). When disturbances occur, the productivity and stability can be affected in different ways depending upon the disturbance severity and the level of biodiversity within the ecosystem. When disease spread is considered as a disturbance event, such as in the case of an epidemic, properties involving the dynamics around disease transmission is also analysed.

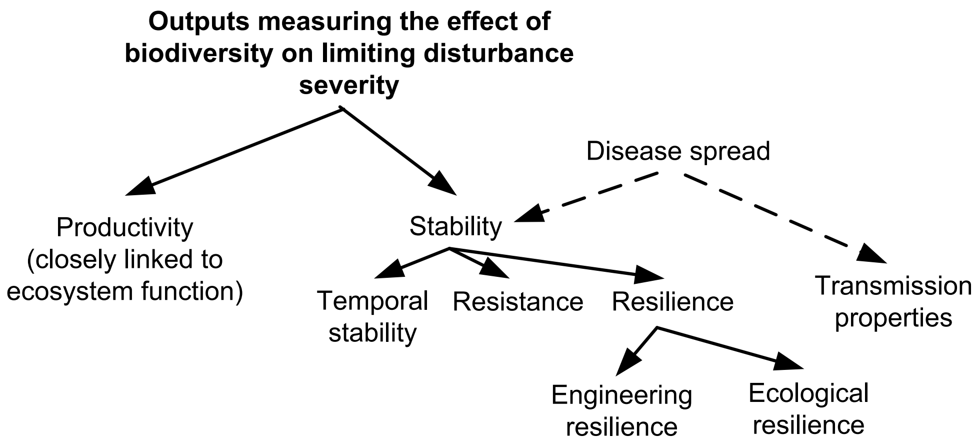

Outputs measuring the effect of biodiversity on limiting disturbance severity
Disease spread
Productivity (closely linked to ecosystem function)
Stability
Temporal stability
Resistance
Resilience
Transmission properties
Engineering resilience
Ecological resilience

Figure 2-7 – The effect of biodiversity on limiting disturbance severity

2.2.2.2.1 *Productivity*

Productivity is about the efficient use of input resources to generate outputs. It is a measure of how much and how quickly something is being produced. Productivity has been used to measure how well a particular ecosystem function is performing under different conditions in relation to diversity [87], and within ecological studies it is generally measured by the rate of increase in the total community biomass (total mass of living matter) in an area [66]. Changes in productivity in relation to disturbances or biodiversity change can be measured over time to assess ecosystem stability (§2.2.2.2.2) [88] [27].

2.2.2.2.2 *Stability*

Stability in relation to ecosystems can have two meanings, either the measurement of the temporal variability of an ecosystem property (*temporal stability*), or the measurement of an ecosystem's ability to defy change such as that from disturbances [89] [90] [22] [27]. Often the temporal attribute measured is the variance in population densities, or changes in productivity, such as that of biomass, over time (see Figure 2-8 (a)) since most biodiversity and ecosystem functioning studies focus on plants or microbial communities [88] [90] [22] [27]. There may also be a tolerance threshold, below which ecosystem functions and services become so degraded that it impacts the ability of the ecosystem to survive or recover. When the stability of a system is a measure of its ability to return to equilibrium following disturbance, two dimensions of stability are used, termed *resistance* (sometimes persistence) and *resilience* [22].

(1) Resistance describes how much of an ecosystem property changes in response to disturbance. The less the property changes the more resistant it is (see Figure 2-8 (b)). For example the resistance of productivity to climate events has been studied in grasslands in relation to diversity [27], as well as resistance of productivity to plant invasions where the invading plant biomass [91], and the invading plant cover [92] were measured in relation to biodiversity with the studies showing that biodiversity can act as a good barrier.

(2) Resilience, more specifically *Engineering resilience* [93] assumes that stable ecological systems operate at a single global equilibrium (one stable state) so that the resilience is a measure of the time taken to return to this global equilibrium following a disturbance (see Figure 2-8 (b)). The faster the ecosystem can recover, the more resilient it is [21] [27]. Ecosystems may react differently to different types of disturbances in which case the resilient and resistant characteristics will change.

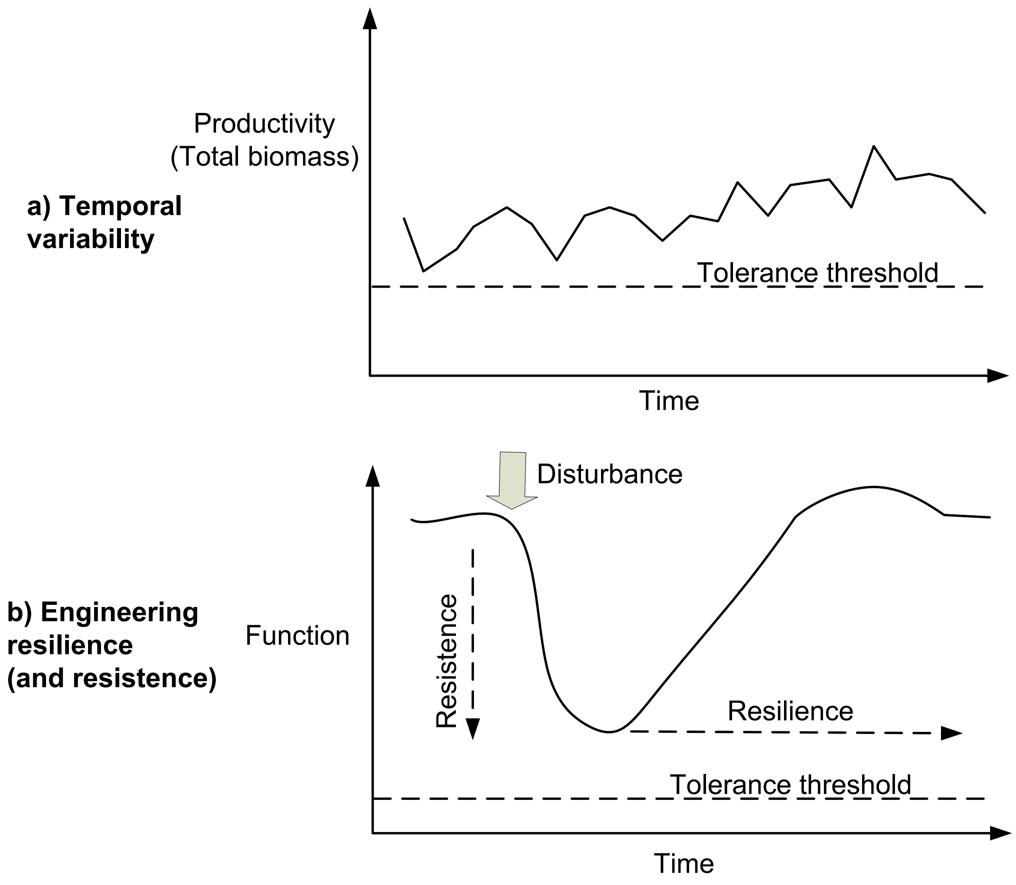


Figure 2-8 – Methods of measuring ecosystem stability

2.2.2.2.3 The Case of Disease Spreading

The spread of a disease is considered as a disturbance event [55] especially if it turns into an epidemic. Controlling the spread of diseases is often defined as a *regulating service* offered by ecosystems (§2.2.1.3) [50], and therefore has been used as another output component of assessing ecosystem health. Whilst some studies in relation to disease spread and biodiversity examine stability components, such as that involving species diversity and productivity changes [94] or resistance of alleles (§2.2.2.1.4) against pathogens in genetic studies [95], others instead focus on changes in actual disease transmission of a population [56]. Experimental research suggests that the effect of *biodiversity loss* on the spread of diseases can have two outcomes; either it can decrease, or increase (majority of cases) transmission [56]. This can be linked to two theories regarding biodiversity and disease spread: The *Dilution Effect* and the *Amplification Effect:*

(1) The Dilution Effect [24] [96] [97] [56], is any factor that causes a relative reduction in: the number of individuals that are susceptible to the disease and can pass it on (suitable hosts) relative to the total number of individuals, or their encounter rates, which can *decrease the transmission* of disease. For example a decrease in the relative number of those susceptible through an increase in the number of different species.

(2) The Amplification Effect [98] [97] [56] is caused by factors that cause a relative increase in: the number of individuals that are susceptible to the disease and can pass it on (suitable hosts), or their encounter rates, which can *increase the transmission* of disease, for example an increase in the number of susceptible individuals when species that are added to increase diversity are also susceptible.

For genetic diversity studies, disease transmission is studied in relation to genetic variation (§2.2.2.1.4), both in terms of genotypes and alleles. There is a general consensus that genetically homogenous populations are more vulnerable to disease transmission than genetically diverse populations [99] [100], which has been seen in agriculture where disease epidemics have destroyed monocultured crops [101] (§1.1.3). Studies also suggest that genotypes with high allelic diversity are needed in a population to constrain transmission [102] [103] [99] particularly when exposed to multiple parasites [103] [99].

2.3 Measuring Biodiversity

2.3.1 Introduction

As outlined in section 2.2.1.1 genetic, species, and functional diversity measures are used to describe biodiversity in ecosystems. The majority of practical studies focus on species diversity because it is easier to measure than genetic diversity [104] [44]. However, measurement at the genetic level can determine diversity within and between the species of whole ecosystems by considering differences in genotypic structure at the individual scale. This section reviews biodiversity measures at the genetic level only, these being referenced during later chapters.

2.3.2 Genetic Diversity

When analysing genetic diversity in relation to genotypes (§2.2.2.1.4) there are two types of measures: those based directly upon genotypes as a whole entity, and those based upon alleles which make up the genotypes. These two aspects are reviewed below.

2.3.2.1 *Genotypic Measures*

Genotypic measures focus directly on the genotype and ignore its allelic construction. A selection of measures described below is used in the literature to assess diversity.

2.3.2.1.1 *Genotypic Richness*

Genotypic richness (G_r) is the number of different genotypes that has been measured within a population. Observational studies can count the number of genotypes, whilst experimental studies can create the required number of genotypes using clonal species [44].

2.3.2.1.2 *The Proportion of Different Genotypes*

The proportion of different genotypes (G_p) within a population of size N is defined in Equation (2-1) [105] as the genotypic richness per population. It has a maximum value of 1 when all individuals within the population have a unique genotype and approaches 0 when there are very few genotypes.

$$G_p = \frac{G_r}{N} \tag{2-1}$$

2.3.2.1.3 *Genotypic Diversity*

Genotypic diversity (G_d) [106] [105] takes into account the frequency of all the different genotypes (g_i), where g is the frequency of the ith genotype, as shown in Equation (2-2) giving an indication as to how the genotypes are distributed across the population. It has a minimum value of 1 when there is only one genotype present in the population, and a value of G_r when multiple genotypes are present and are evenly distributed, up to a maximum value of $G_r = N$ when all individuals have a unique genotype.

$$G_d = \frac{1}{\sum_{i=1}^{i=G_r} {g_i}^2} \tag{2-2}$$

2.3.2.1.4 *Genotypic Evenness*

Genotypic evenness (G_e) [105] as given in (2-3) specifies how evenly or dominantly the genotypes are distributed amongst the population. When a single genotype dominates, providing that there is more than one genotype present in the population, the evenness approaches 0. When the genotypes are evenly distributed, the evenness has a maximum value of 1.

$$G_e = \frac{G_d}{G_r} \tag{2-3}$$

2.3.2.2 *Allelic Measures*

Allelic measures concentrate on the genetic variation of alleles across a population where alleles are positioned at different loci within a genotype (§2.2.2.1.4). A selection of measures described below is used in the literature to assess diversity.

2.3.2.2.1 *Allelic Richness*

Allelic richness (A_r) [44] is the average number of different alleles per locus that has been measured across a population. It is on a par with genotypic richness but is now focused at the allelic level. Similar to genotypic richness it does not consider how many instances of each allele are present. As shown in Equation (2-4) The number of different alleles (a) at each locus (l) is summed and then divided by the total number of loci (L).

$$A_r = \frac{1}{L}\sum_{l=1}^{l=L} a_l \tag{2-4}$$

2.3.2.2.2 *The Nei Genetic Diversity*

The Nei Genetic Diversity (D) [42] is defined as the probability that at a single locus any two alleles chosen at random from the population are different to each other. This principle applies for monoploid (haploid), diploid and any other polyploidy chromosome sets (§2.2.2.1.4) but is very rarely stressed in the literature [42] [107] since most studies measuring genetic diversity in this way focus on diploid chromosome sets of animals and plants. For diploid chromosome sets the genetic diversity measure for a single locus is referred to as the *expected heterozygosity* which is a measure of how different the two allele pairs are (§2.2.2.1.4). For monoploids the terminology of heterozygosity cannot be applied but the Nei Genetic Diversity is still valid since it assumes that any two alleles chosen at random can be from different individuals. The frequency (f_i) of each different allele (i) at each locus can be calculated using Equation (2-5) as the number of times the allele is present (n_i) divided by the total number of alleles (A) across the population. The value (A) is equivalent to the population size (N) for monoploids and twice the population size $(2N)$ for diploids since a diploid has two alleles for each gene.

$$f_i = \frac{n_i}{A} \tag{2-5}$$

The probability that two alleles chosen at random will be the same (p_s) is given by Equation (2-6) and is also a measure of homozygosity (§2.2.2.1.4) for a population with diploid chromosomes. This is summed over all the different allele possibilities (a) (Not to be confused with A which is the total number of alleles across a population, for which the same allele may occur multiple times).

$$P_s = \sum_{i=1}^{i=a} (f_i)^2 \tag{2-6}$$

Subtracting this from unity gives the probability (p_d) that two alleles chosen at random will be different and denotes the genetic diversity at a single locus (l), which is given in Equation (2-7). For a population with diploid chromosomes this will be a measure of *heterozygosity*.

$$P_{d_l} = 1 - \sum_{i=1}^{i=a} (f_i)^2 \tag{2-7}$$

The final diversity index (D) is usually calculated by averaging the diversity across all loci (L) as given in Equation (2-8). The genetic diversity index has values between 0 where every individual in the population has the same set of alleles, and 1 if every individual has a different allele at every locus.

$$D = \frac{1}{L} \sum_{l=1}^{l=L} P_{d_l} \tag{2-8}$$

2.3.2.2.3 *The Shannon Diversity Index*

The Shannon Diversity Index [108] (Shannon entropy) was originally used to quantify the uncertainty of information content in strings of text. The greater the numbers of different letters, and the more equal their frequency within the text, the more difficult it is to correctly predict which letter will come next. The same concept can be applied to alleles where the more alleles there are at a locus and the more equal their distribution amongst the population, the more diverse the population becomes [44]. The Shannon entropy for a given locus (H_l) is given in Equation (2-9), and is firstly calculated in a similar manner to the Nei Genetic Diversity by measuring the frequency (f_i) of each different allele (i) at the locus. Different logarithmic bases have been used for the index such as the natural logarithm, and the base 2 logarithm [109]. The equation is summed over all the different alleles possibilities (a).

$$H_l = -\sum_{i=1}^{i=a} f_i \, ln(f_i) \tag{2-9}$$

Similar to the Nei Genetic Diversity measure, the Shannon entropy (H) can be averaged across all loci (L) as given in Equation (2-10).

$$H = \frac{1}{L}\sum_{l=1}^{l=L} H_l \tag{2-10}$$

The maximum diversity occurs when all alleles are equal in frequency. The upper limit for a single locus is governed by the number of different allele possibilities at that locus (a) and can be simplified to Equation (2-11).

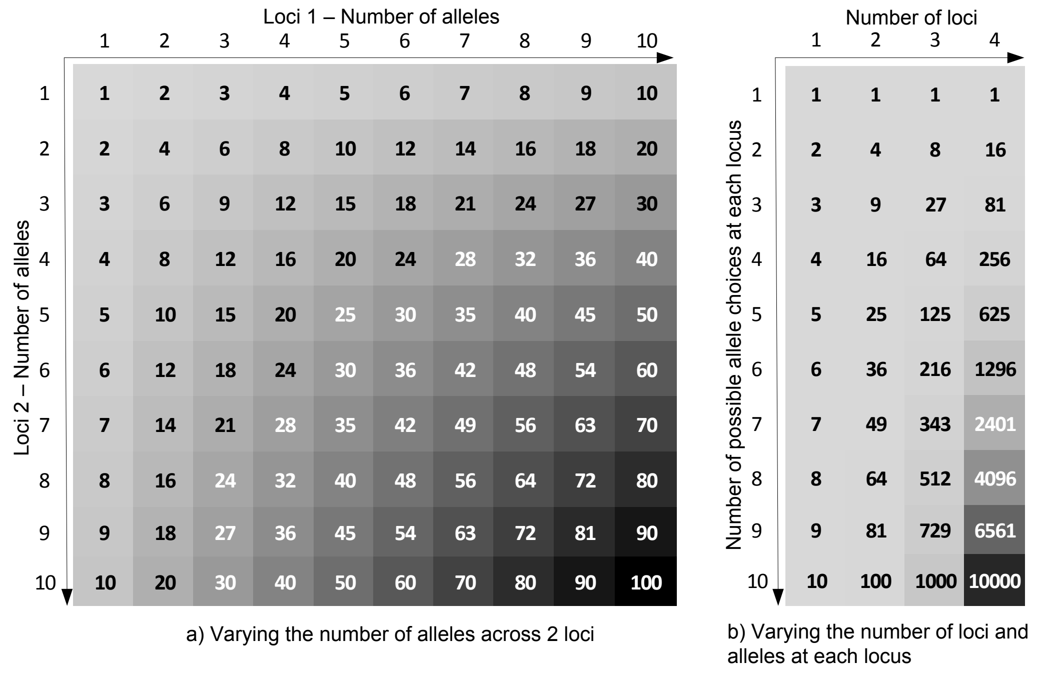
$$H_{l_{max}} = ln(a) \tag{2-11}$$

2.3.2.3 Maximum Number of Unique Genotypes

For a monoploid set of chromosomes, the maximum number of unique genotypes ($G_{r_{max}}$) that can be created is the product of the number of different allele possibilities (a) at each of the loci (l) as given in Equation (2-12).

$$G_{r_{max}} = \prod_{l=1}^{l=L} a_l \tag{2-12}$$

Where (L) is the total number of loci. Figure 2-9 (a) shows the number of unique genotypes for two loci having up to 10 alleles, and Figure 2-9 (b) shows the number of unique genotypes for four loci with the same number of alleles at each locus. From the opposite perspective, the number of loci and alleles needed to represent *at least* a specific number of genotypes will in general have a number of solutions.



Loci 1 – Number of alleles (columns); Loci 2 – Number of alleles (rows)

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

a) Varying the number of alleles across 2 loci

Number of loci (columns); Number of possible allele choices at each locus (rows)

	1	2	3	4
1	1	1	1	1
2	2	4	8	16
3	3	9	27	81
4	4	16	64	256
5	5	25	125	625
6	6	36	216	1296
7	7	49	343	2401
8	8	64	512	4096
9	9	81	729	6561
10	10	100	1000	10000

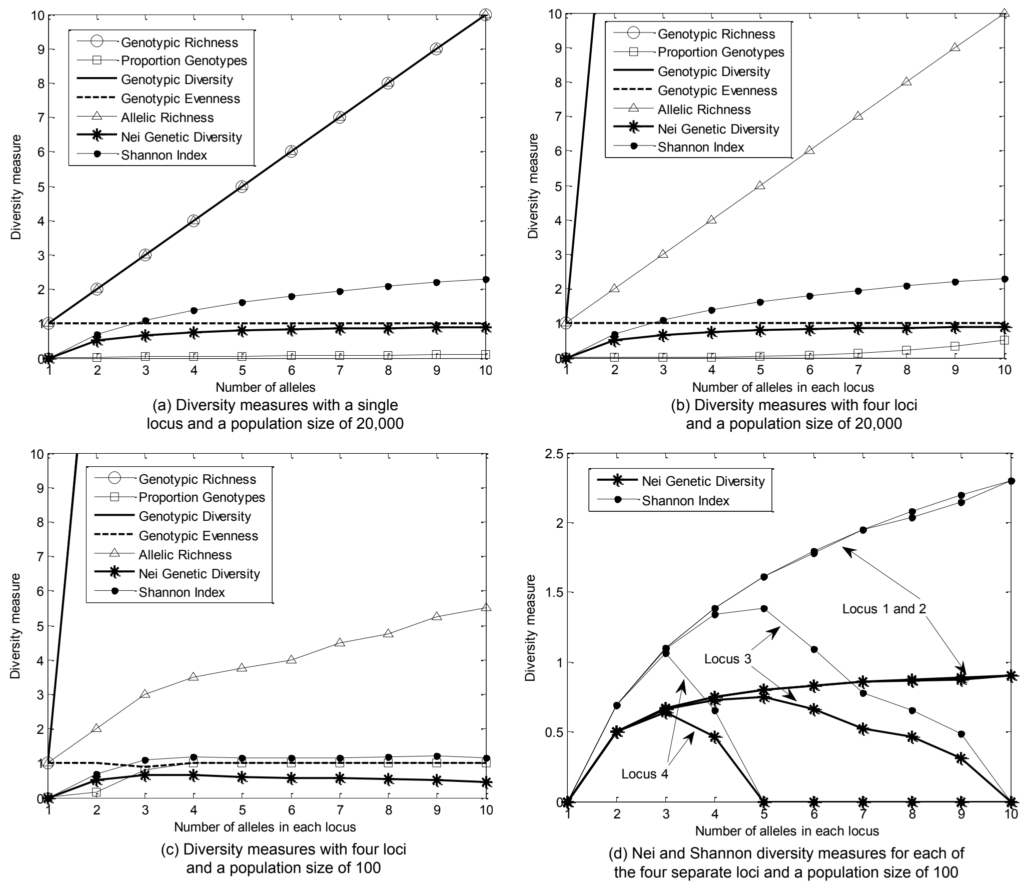
b) Varying the number of loci and alleles at each locus

Figure 2-9 – Maximum number of unique genotypes

For example, to generate at least 50 genotypes, there needs to be at least 2 loci with 7 and 8 alleles (Figure 2-9 (a)) or 4 loci and 3 alleles in each (Figure 2-9 (b)).

2.3.2.4 *Comparison of Measures*

Figure 2-10 (a) and (b) show comparisons of the genotypic and allelic measures of diversity for a simulated population having genotypes comprising a single locus and four loci respectively. It is an illustration to support the mathematical equations highlighting differences between what they show. The number of different allele possibilities (a) is the same at each locus and is varied between 1 and 10. The population size (N) of 20,000 has been chosen such that it is twice the maximum number of possible genotypes $(G_{r_{max}})$ from a four locus, 10 allele combination. This is to allow the simulation of genotypes to occur at least twice and be evenly spread across the population so that maximum diversity is achieved and can be observed in Figure 2-10 (a) and (b). For a single locus as shown in Figure 2-10 (a), the Genotypic (G_r) and Allelic (A_r) Richness increase together linearly with the number of alleles since a single locus with one allele can have only one possible genotype, two alleles can have two genotypes, and so on. The Genotypic Diversity (G_d) also follows the same relationship, since when the genotypes are evenly distributed, its value is equal to the Genotypic Richness. It also follows from even distribution that the Genotypic Evenness (G_e) measure is flat at unity across any number of alleles. The Nei and Shannon measures both show the maximum allelic diversity values that can be achieved when the alleles, and hence genotypes are evenly distributed. The difference between the measures being that the Nei Genetic Diversity asymptotically approaches unity as the number of alleles increases whereas the Shannon Index increases with the number of alleles.



(a) Diversity measures with a single locus and a population size of 20,000

(b) Diversity measures with four loci and a population size of 20,000

(c) Diversity measures with four loci and a population size of 100

(d) Nei and Shannon diversity measures for each of the four separate loci and a population size of 100

Figure 2-10 - Comparison of genetic diversity measures

The Proportion of Genotypes (G_p) measure remains low for any number of alleles up to 10, indicating that although the alleles and genotypes are evenly distributed leading to maximum diversity under these constraints, the number of unique genotypes in comparison to population size is very small.

With four loci as shown in Figure 2-10 (b) the averaged allelic measures across all loci are the same as that for a single locus since the chosen population size is large enough to achieve even distribution and maximum diversity given the locus and allele constraints. Differences are seen in the increased Genotypic Richness (G_r), Genotypic Diversity (G_d), and hence the Proportion of Genotypes (G_p) in comparison to population size which reaches a half when four loci with ten alleles are used.

Figure 2-10 (c) shows the same measures for four loci but with a limited population size of 100. Figure 2-10 (d) shows the Nei and Shannon measures separated out for each individual locus. The genotype assignment to individuals within the population is set so that the minimum number of alleles are used to achieve the maximum number of genotypes.

As shown in Figure 2-10 (d) when there are up to three alleles within each locus, the population size is *greater* than the potential number of genotypes (see Figure 2-9 (b)) and so all alleles occur within the population and are distributed as evenly as possible. Additionally, the Genotypic Richness (G_r) is limited by the number of loci and alleles. When there are four or more alleles in each locus the population size becomes *smaller* than the potential number of genotypes. Under this condition the Genotypic Richness (G_r) is limited by the population size. The actual genotypes of the population are a subset of those available for which there could be many different subsets, with potentially some alleles either not being expressed, or dominating at a particular loci. This means that even when every individual in the population has a different genotype, the allelic diversity (Nei and Shannon) may not necessarily be maximal. This is illustrated in Figure 2-10 (d) when only a minimum number of alleles are used to achieve richness. Domineering alleles in loci three and four reduce the diversity to zero at these particular loci when the number of alleles in loci one and two is increased to maintain genotypic richness. This thereby reduces the overall allelic diversity measures of Nei, Shannon, and Allelic Richness as shown in Figure 2-10 (c). This example provides a key illustration of the differences between genotypic measures and allelic measures, where it may be possible to maximise genotypic richness and diversity without fully exploiting the potential allelic richness and diversity. This makes the use of both types of measures important for assessing genetic diversity.

2.4 Summary

An ecosystem comprises interacting organisms and their physical environment, resulting in distributed, self-organising, and emergent properties. Biodiversity encompasses the variety of genes, species, or functional traits within an ecosystem and is critical to the functioning of such ecosystems and the emergent services they provide. Ecosystem functions are the ecological processes that take place and the ecosystem services are the benefits provided to humanity. External influences, termed disturbances, can impact on biodiversity and function and affect these services. There are several mechanisms that link biodiversity to ecosystem function including niche differentiation (particularly functional complementarity), facilitation, interactions between trophic levels, and genetic variation. Ecosystem health, and in particular its relationship with biodiversity, is often assessed by looking at the outputs of ecosystem functions and services where productivity, stability, and disease transmission are measures often used within studies. Stability can have multiple meanings such as the variance of an attribute, or the ability to defy change in which the two dimensions resistance and resilience are often used.

The literature suggests that biodiversity loss can cause either dilution or amplification of susceptible individuals and thereby reducing or increasing disease transmission. These ecological concepts are mapped onto peer-to-peer networks in the form of an ecosystem model of an ad hoc network and are described in chapter 5. Security attacks such as malware forms unwanted disturbances to the ecosystem model.

Diversity measured at the genetic level can determine diversity within and between the species of whole ecosystems by considering differences in genotypic structure at the individual scale. There are two types of genetic diversity measures: those based directly upon genotypes as a whole entity, and those based upon alleles which make up the genotypes. Whilst genotypic measures are useful in identifying the uniqueness of the population and the distribution of genotypes, allelic measures can additionally show the distribution of alleles which can be analysed either independently at each locus or as an average across the whole genotype. When the measures are analysed together they provide a useful picture of the genetic diversity of the population from both genotypic and allelic aspects. The following measures are used as metrics in the measurement of diversity within the ecosystem model of an ad hoc network (defined in 5.3.2.1): Genotypic Richness, Genotypic Diversity, Allelic Richness, and Nei Genetic Diversity Index. The Genotypic Richness, Allelic Richness and the Shannon Index are referenced in section 3.4.4 during a review of diversity measures of computing systems for security.

Chapter 3

Cyber Security and Diversity in Computing Systems

3.1 Introduction

The purpose of this chapter is to introduce the background material associated with: the practical viability of diversity, how greater numbers of connected devices is driving more peer-to-peer wireless networks, how malware and vulnerabilities are associated with different layers of the software stack, and what gaps there are in this field of research. This chapter is split into three sections:

Computing Systems: The first section discusses enabling technologies of future computing systems that have the potential to aid in the realisation of biodiversity as a security mechanism. These include automated software generation and dissemination, virtualisation and hardware support, and the modularity of software stacks. Topology is considered with a focus on networks conducting peer-to-peer communication.

Cyber Security: The second section on cyber security predominantly discusses malware, which is a form of cyber attack rife in monoculture software environments. It summarises the different types of malware along with the stages of a successful malware attack. The location and types of vulnerabilities exploited by malware within the software stack are discussed. The implications of attacks using multiple and publically unknown exploits are highlighted.

Diversity: The third section explores the literature on the current state of research associated with diversity within computing systems. Three main areas of research are reviewed including the diversification of software at the code level, diversification at the network level and the metrics used to evaluate diversity within such systems. This section concludes with an evaluation of the open areas of research within this field.

3.2 Computing Systems

3.2.1 Enabling Technologies

The future of computing systems lies within a globally connected world of devices and people, and will combine advancements in enabling technologies to provide access anywhere and at anytime. Some of these enabling technologies could also be utilised to realise diversity. Particular attention is given to automated software generation, including dissemination and updating, virtualisation and hardware support, and the modularity of software stacks.

3.2.1.1 Automated Software

The dissemination of software traditionally involved a pre-installation on a new device, or through the purchase of a disk. Nowadays software can be readily downloaded via the Internet, updates are often automated, and users can choose from a broad range of application software. Franz [110] identifies one of the fundamental enablers of diversity to be the ease of obtaining software, making it possible to distribute and patch unique versions. The advancement of *dynamic software compilation* and *cloud computing* could be harnessed to provide the necessary computing power to generate large volumes of these unique versions as and when required. In addition to this, efforts are being sought to prevent the need to restart software or computers when patches are applied. Much research has been conducted around *dynamic software updating* (DSU) which would allow the unique versions to be updated or modified without affecting functionality or run-time performance [111] [112].

3.2.1.2 Virtualisation

Virtualisation is seen as one of the key enabling technologies for the future Internet. It is the artificial creation of a resource such as a hardware platform, storage device or server by combining or partitioning physical hardware or software and isolating it from the rest of the system [113]. For example the resources of a single computer could be partitioned so it appears there are two isolated computers available instead of one. Virtualisation has grown rapidly because of its use in cloud computing [114] and Bring Your Own Devices (BYODs) [115]. It has been used for many years in desktop computers, but more recently in mobile devices with software such as 'Horizon Mobile' by VMware [116], and open source software led by the Xen project and backed by AMD and Google [117]. Virtualisation has use in *networks* [118] [119], *servers*, *services* [120] [121], *physical objects* [122] [123], and *devices* (embedded) [113], increasing hardware utilisation, security, and efficient administration [124].

Virtualisation within devices could prove to be a useful tool in the realisation of software diversity due to its ability to switch between isolated software programs, operating systems, or entire software stacks, and could provide an alternative to, or complement the research field of dynamic software updating. Devices could be pre-installed with only a low level virtualisation and management layer, so that the enabling technologies of dynamic software compilation and cloud computing could be used to provide hardware-independent functionality and individually tailored operating systems and drivers as and when required [113]. Virtualisation can also isolate malware prone applications by providing some protection against known and unknown viruses through protecting the disk and files. If an infection occurs software can be reloaded to its original, known good state and thus remove the malware. Virtualisation can be *partial*, for example through sandboxing (Figure 3-1 (a)) of malware prone applications such as web browsers. Sandboxes examine certain system calls for malicious behaviour, then rewrite or block them as appropriate. Virtualisation can also be *full* using *virtual machines* (VMs) (Figure 3-1 (b)) to isolate whole operating systems [125]. VMs are created and managed by Hypervisors [126] which either sit directly on top of the physical hardware (type 1 hypervisor) or sit on top of the host operating system (type 2 hypervisor). Although virtualisation has the potential to aid diversity, there are a number of design issues that would need to be addressed before it can be practically used (§9.3.2.4.)

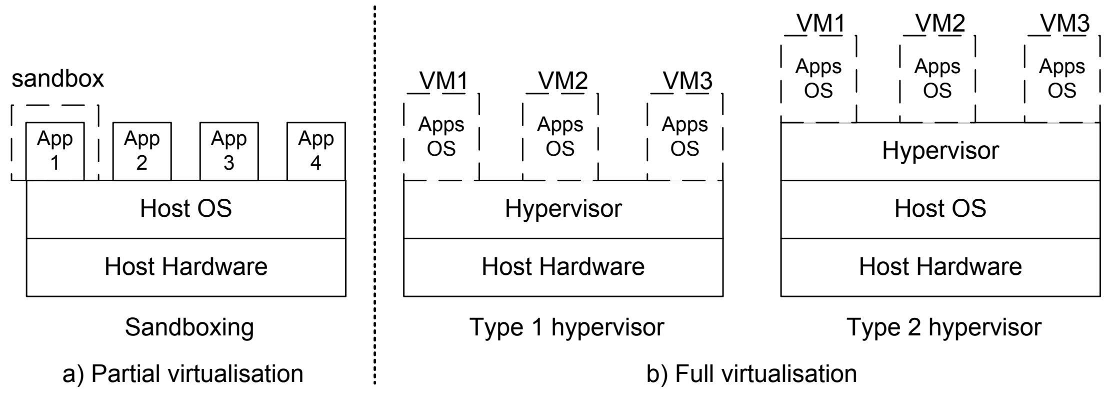


Figure 3-1 – Virtualisation scenarios

It is not just virtualisation however that could support diverse software, hardware enabling technologies could also be important for the successful deployment of diversity. Chip designs made up of small processors for parallel software tasks could have the potential to accommodate diverse software onto a single chip [127] [128].

3.2.1.3 Software as a Modular Structure

All computing devices are equipped with an underlying hardware and software architecture. The latter is comprised of numerous software components organised into layers that perform specific functions and is called a *software stack*. Complementary to the software stack is the *network protocol stack*, which is also comprised of layers, but contains protocols defining the communication from one device to another within the network [129]. Operating Systems have well defined software stacks and adhere to the relevant network protocol stacks to communicate across the network.

Within the future it is likely that this modularity will remain, but with increased functionality. There has already been an explosion of functionality of user software with the introduction of 'App stores' but the increase in functionality is likely to extend to other layers of the stack as more emerging products enter the IoT. The ability to partition software into layers and functionality, whether source code or binary files, could be beneficial for creating diverse computing systems where alternative versions can be generated with the same functionality using the same or different techniques at each layer of the stack.

Four different software stacks supporting computing devices are described here and shown in Figure 3-2 to illustrate the similarities between them in terms of software layers and functionality. The first three software stacks: the Android [130] [131], iOS [132] [133] and Windows 8 [134] [135] are all distinct operating systems that can be used with mobile devices. The Windows 8 architecture has a split software stack with a shared kernel. One half caters for a modern 'Style' with touch screen capability and the other half encompasses the old classic desktop structure. The fourth software stack: the generic open source Linux OS [136] [137] is designed as a modular structure so that different distributions such as Ubuntu or Debian can be used with the same underlying core libraries, with a pick and mix of different software packages and versions. Although the layers and software components across all four operating systems are named and partitioned differently, the general functionality remains consistent across the architectures. The layers can be partitioned into four main categories. Starting from the lowest layer that sits just above the hardware, the categories are: kernel, core OS libraries, application services, and applications.

	Android	Windows 8		iOS	Linux Eg. Ubuntu / Linux Mint / Debian / Fedora etc
Applications	*Applications:*	*Applications:*		*Apple Apps*	*Applications:*
		Style (modern)	**Desktop (classic)**		
	Browser (android browser) Email (GMail,Outlook,Molto etc) Word processor (officeSuite Pro, Google docs, etc)	Browser (iexplorer) Email (Outlook) Word processor (MS word)	Browser (iexplorer) Email (Outlook) Word processor (MS word)	Browser (Safari) Email (iOS mail, Boxer etc) Word processor (Quickoffice, elements)	Browser (Firefox, Opera etc) Email (Thunderbird, Opera mail etc) Word processor (Apache, OpenOffice)
Application services	*Application Framework:*	*Languages and runtime APIs:*	*User Interface:*	*Cocoa Touch:*	*GNU desktops & interfaces*
	Activity Manager Window Manager Package Manager Resource Manager Location Manager Telephony Manager Notification Manager Internet messaging (XMPP)	User interface Language support, Network, Storage, Media, Security Runtime lib, Database (SQL) H/W accelerator (Open GL) ...	Window management (Windows Forms), Media streaming (Silverlight), Graphics (GDI/GDI+) user interface,	User interface framework, Multitasking, High level system services,	Desktops (KDE, GNOME, LXDE, XFLE, Cinnamon, MATE, LXQt, Budgie, etc) Gui interface/graphics (FLTK,GNUstep, GTK+) Window Manager(EFL) Windowing (X11,Wayland) Gaming (SFML)
Core OS libraries	*Libraries and runtime:*		*Languages and API:*	*Media:*	*GNU open source packages*
	Android core libraries & Dalvik Virtual Machine, Web kit, Graphics (SGL) Security (SSL) H/W accelerator (Open GL) C language library (Libc) Media framework Surface manager Database(SQLite),		Languages: C,C++,VB, .NET etc, Runtime libs, Win32 API, Database (SQL) H/W accelerator (Open GL)	Multimedia streaming, Graphics & audio & video management, Text kit, H/W accelerator (Open GL)	Media library (SDL) Database (MySQL) Web server (Apache) System (systemd,runit) Software management (APT) Graphics (Mesa, AMD Catalyst, Synaptic) Web scripting (PHP) C language library (glibc)
				Core services:	
				System level services e.g networking, data, media, system configuration Database (SQLite)	
Kernel	*Linux Kernel:*	*Core OS kernel drivers:*		*Core OS:*	*Linux Kernel:*
	Device drivers e.g camera / USB / display WiFi / Bluetooth etc, Low level networking, Power management, Memory management, File management, Synchronisation Inter-process communication	Device drivers e.g camera / USB / display WiFi / Bluetooth etc, Low level networking, Power management, Memory management, File management, Synchronisation, Inter-process communication		Device drivers e.g camera / USB / display WiFi / Bluetooth etc, Low level networking, Power management, Memory management, File management, Synchronisation, Inter-process communication hardware accelerator Security framework	Device drivers e.g camera / USB / display WiFi / Bluetooth etc, Low level networking, Power management, Memory management, File management, Synchronisation Inter-process communication Security (SELinux, TOMOYO)

Figure 3-2 – Comparing different operating system software stacks

- The *kernel layer* contains the low level functions such as device drivers that interact directly with the hardware such as the specific camera built into the device, the graphics card, and USB ports. There is also: low level networking functionality such as for a WiFi card, power management, memory management and file management as well as inter-process communication and threads between communicating software components. The Android operating system uses the open source Linux kernel [138] as the basis for its low level functions.

- The *core OS libraries layer* contains a multitude of libraries performing numerous functions supported by different programming languages. For example a library supporting graphics functions both feeds into the application services layer to support the drawing of graphical user interface (GUI) windows, and communicates with the graphics card device driver in the kernel. Other notable libraries at this layer include networking and web support, database functions such as support for the Structured Query Language (SQL), media support such as multimedia streaming, video and audio capabilities including CODECs (Coder Decoder: coding and decoding of media files), and security such as Secure Sockets Layer (SSL) for establishing encrypted links between web servers and browsers.

- The *application services layer* is the level at which application frameworks are created, with functionality such as window managers that control the position, style and timing of windows drawn on the display screen. The open source Linux software stack has separate windows managers and desktop software packages. The GNOME desktop, for example, uses the GTK+ toolkit containing a collection of applications to form a graphical environment which itself uses the X11 windowing application program interface (API). The multimedia streaming package in Windows 8 (Silverlight) is an application framework for browser multimedia applications and is used by Netflix for streaming films and television programs. The Android application framework is comprised of a number of managers controlling different aspects.

- The *applications layer* is where all the user software is found. Applications utilise the application services layer of an operating system, and sometimes libraries in lower layers to create interactive user software. Internet browsers for example sit at this layer for which there can be different products that are compatible with the same operating system such as Firefox and Opera (plus others) for Linux, or different versions of a product across operating systems such as Firefox for Linux or Firefox for Windows.

There are compatibility issues between different software stacks due to dependencies on lower layer libraries, with often only one choice available for a specific function. It is only at the application layer where there tends to be more choice of software, particularly with the introduction of 'App stores'. The evolving suite of open source Linux software modules however at lower layers provides a wider choice of functions that can be mixed and matched as appropriate with the added benefit of being compatible. The development of open source software within the IoT is also growing [139] [140] [141] [142] [143] [144] [145]. This increased use of open source could provide a natural method of software diversity since there can often be alternative choices of modules providing similar functionality. Additionally open source is constantly under scrutiny meaning bugs tend to be fixed quickly, and it costs less in monetary terms for the end user than proprietary counterparts making it a cost effective way of introducing diversity and fixing vulnerabilities.

3.2.2 The Future Topology of Connected Devices

In the past, society has seen the integration of mobile phone networks and the Internet using smartphone devices, third generation (3G) networks and protocols, local wireless access points using WiFi and wireless peer-to-peer communication using Bluetooth [146]. In the future, the IoT will combine enabling technologies with many different types of objects, for a vast range of applications requiring improvements in networks and services [8] [6] [147] (Figure 3-3). Traditional internet networks are based upon the application layer client-server model [129]. In the future, the IoT is likely to be constructed from different topologies utilising a multitude of communication protocols, depending upon the connected devices and their application (Figure 3-3). There will be more localised *peer-to-peer communication* such as device to device (D2D) or machine to machine (M2M) making more use of protocols such as Bluetooth, or the fourth generation (4G) WiFi Direct and LTE Direct, or their fifth generation (5G) equivalents when they are released [146]. They may also be connected in an ad hoc fashion, as and when the services are required, such as in the case of moving phones or vehicles creating localised *ad hoc networks* [148].

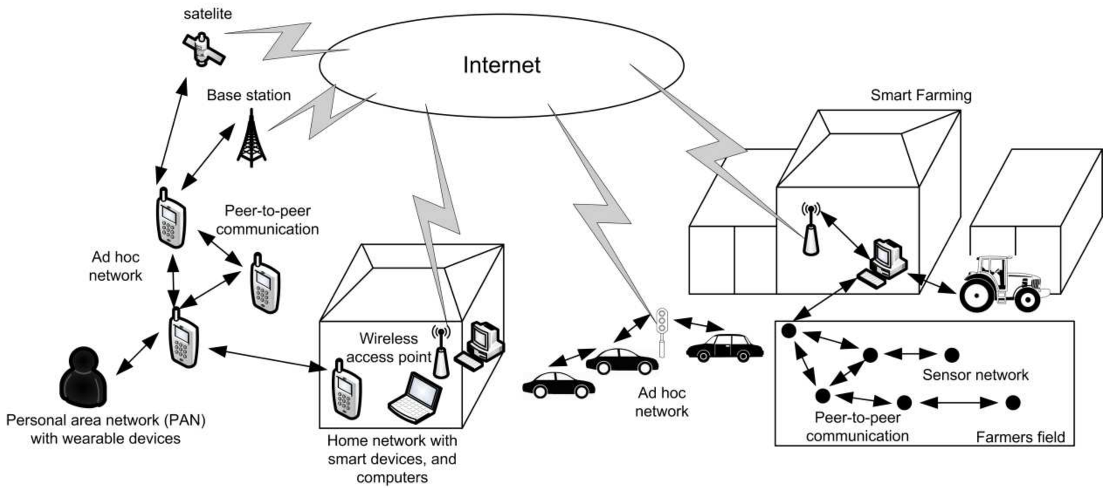

satelite
Internet
Base station
Smart Farming
Peer-to-peer communication
Ad hoc network
Wireless access point
Personal area network (PAN) with wearable devices
Home network with smart devices, and computers
Ad hoc network
Sensor network
Peer-to-peer communication
Farmers field

Figure 3-3 – Topology of the future Internet

3.2.3 Peer-to-Peer, Ad hoc, and Sensor Networks

Localised *Peer-to-peer communication* describes the direct communication between one device and another. This section describes different types of peer-to-peer networks and where mobile ad hoc networks fit in. Several networks communicate in a peer-to-peer fashion, although the underlying mechanisms and network topology may be different.

A *peer-to-peer overlay network* has a distributed architecture and generally operates at the application layer of a network protocol stack (§3.2.1.3) using the Internet as the underlying network and can operate over wired or wireless connections [149]. In a traditional client-server model, shown simplistically in Figure 3-4 (a) without including network detail, a user will communicate with a single server to transfer a whole file. By contrast, in a peer-to-peer overlay network (Figure 3-4 (b)) connections with multiple hosts are made with many small data requests to each. The peer-to-peer client then combines the data to recreate the file. BitTorrent is one of the most popular peer-to-peer file sharing protocols and is often used for downloading films [150].

Whilst peer-to-peer overlay networks provide logical peer-to-peer communication, *ad hoc networks* provide physical peer-to-peer connections. They are formed at the lower network layer of a network protocol stack (§3.2.1.3). They also have a distributed architecture, but devices used within ad hoc networks tend to interact closely with humans often following human mobility patterns (§4.2). Each node in the network acts as a router and a host which self-configure to form an arbitrary topology [151] (Figure 3-4 (c)). Nodes communicate through single-hop and multi-hop paths to each other in a peer-to-peer fashion. For nodes that are both mobile and wireless with multi-hop functionality, they are generally referred to as Mobile Ad hoc NETworks (MANET's) [148] [152].

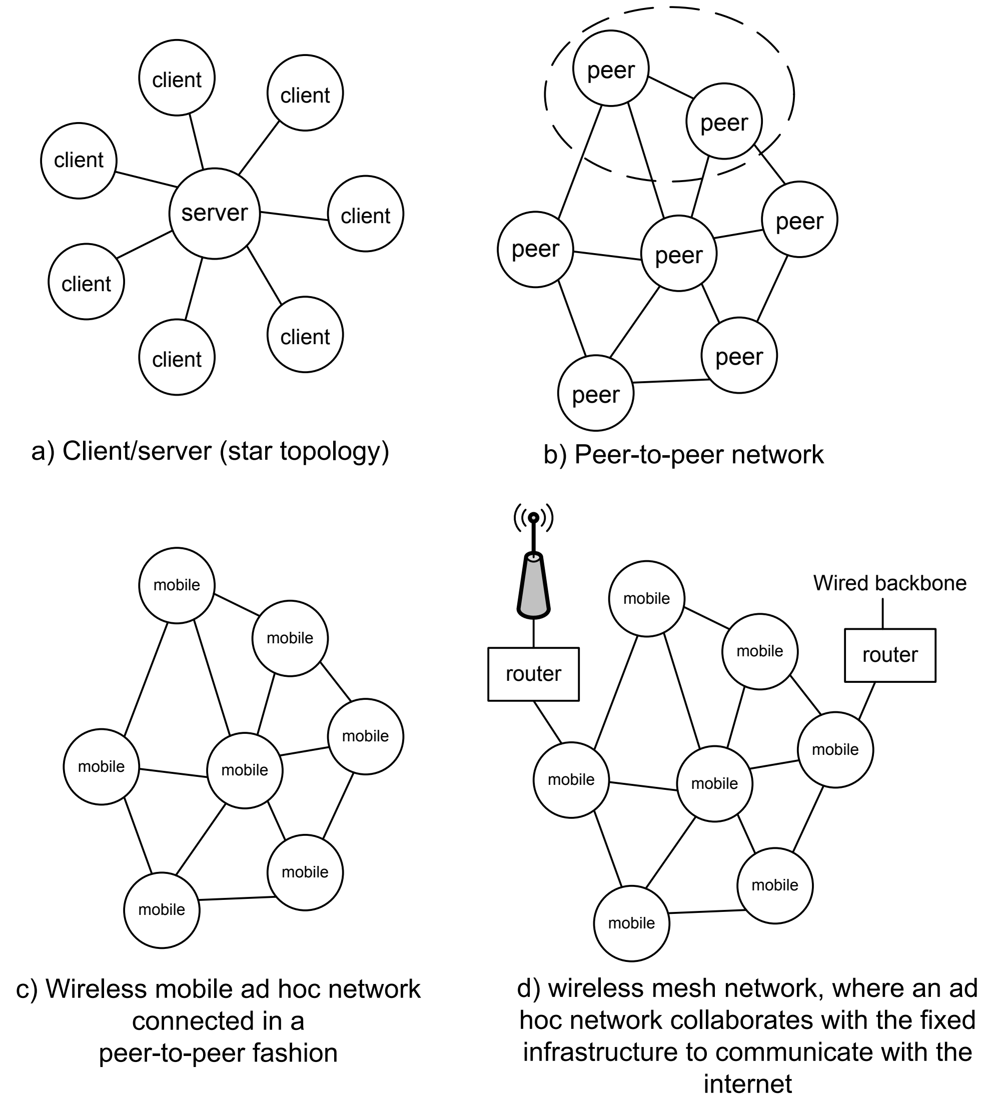

client
client
client
server
client
client
client
client
a) Client/server (star topology)
peer
peer
peer
peer
peer
peer
peer
b) Peer-to-peer network
mobile
mobile
mobile
mobile
mobile
mobile
mobile
c) Wireless mobile ad hoc network connected in a peer-to-peer fashion
router
Wired backbone
router
mobile
mobile
mobile
mobile
mobile
mobile
mobile
d) wireless mesh network, where an ad hoc network collaborates with the fixed infrastructure to communicate with the internet

Figure 3-4 – Network topologies

Mobile phones equipped with Bluetooth currently use single hop communication for transferring files directly between devices. In the future wireless communication standards will incorporate multi-hop functionality [153] allowing the creation of mobile ad hoc networks. It is likely that most ad hoc networks will not operate in isolation, requiring some kind of gateway to the Internet. *Mesh networks* (Figure 3-4 (d)) may provide this, where ad hoc devices collaborate with fixed infrastructure to enable access to the Internet [153]. Ad hoc networks are beneficial for temporary scenarios such as mobile phone communication, rescue operations, health care, and much more [154].

Sensor networks are sometimes discussed in the literature as a type of ad hoc network, but they can also be considered different from ad hoc networks [154] depending upon the type of network topology and application areas that are being considered [155]. Devices contain sensors and actuators and collaborate between themselves using wired and wireless technologies which may be static or mobile. They normally have a central device responsible for gathering sensed data called the sink or master and interact more closely with the environment for applications such as machine surveillance, tracking of goods, and precision agriculture [154]. The three most common topologies are mesh, star and tree (Figure 3-5), where the mesh topology could incorporate an ad hoc network if required for a specific application [155].

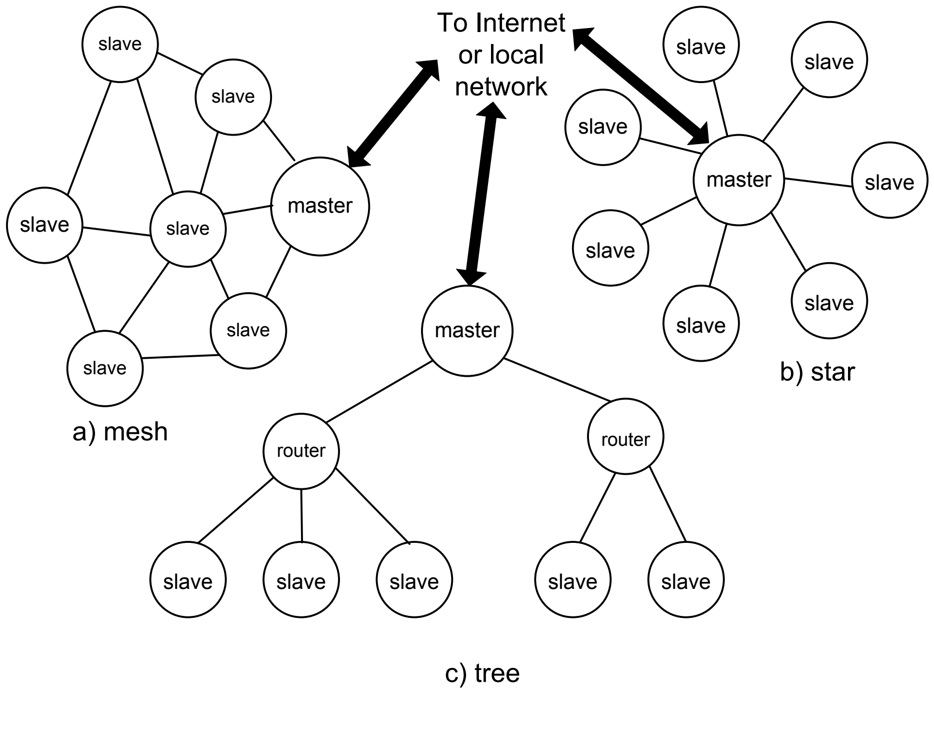


Figure 3-5 – Sensor network configurations

3.3 Cyber Security

3.3.1 Malware in a Monoculture Environment

The increased use of computing devices and wide scale adoption of a limited number of operating systems (OS) and common protocols continues to pose a significant software monoculture threat. Malware is prolific in monoculture environments since it can spread over networks taking advantage of software, such as widely used operating systems, that all have the same vulnerability. Malware is any malicious software used to interfere with computer operations, access private data and systems, or display unwanted advertising. It can infect or delete files, deny services by flooding the network, enable remote access to control devices, modify system applications, prevent functions from working or even turn off security features such as antivirus tools. The main types of malware include:

- *Viruses:* Attach to other programs to spread, and self-replicate when executed.
- *Worms*: self-replicate without needing other programs to spread. Sometimes require user interaction to initiate the spread (e.g. Cabir Worm [156]).
- *Trojans:* Appear as legitimate software (e.g. hidden within 'App stores' [157]) and can harbour spyware, ransomware, or adware.
- *Spyware*: Capture sensitive data or key presses to obtain login details.
- *Ransomware*: Extract money by encrypting files or locking the device until a ransom is paid (ransomeware targeting mobile users is increasing [157]).
- *Adware*: Launches unwanted advertisements.

3.3.2 Successful Malware Attacks

Malware takes advantage of vulnerabilities unintentionally (mostly) created in the design and implementation of software code. *Exploit code* is written and used within malware to exploit a vulnerability. The exploit code can comprise a small piece of software, a block of data or a chain of commands. A successful malware attack requires several steps as shown in Figure 3-6. The first is an entry point for an exploit utilising a vulnerability through which there is redirection of control on the target computer to download a payload. The payload could be the malware itself, or a downloader which then creates a backdoor for other types of malware to be installed. Malware then carries out its intended execution such as stealing data, or causing damage. If the malware has avoided detection and has been programmed to spread over the network, it will then start infecting other computers, either straight away, or after a trigger. Sometimes exploit kits are used which include pre-written exploit code targeting vulnerabilities in unpatched software. Some exploit kits run on web servers, with the purpose of identifying software vulnerabilities in client machines so that malware can be executed.

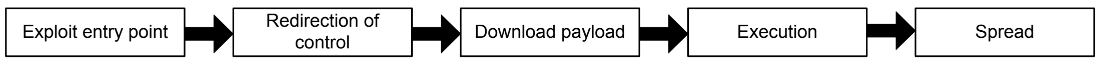


Figure 3-6 – Steps of a successful malware attack

3.3.3 Multiple Exploits and Zero-day Attacks

The use of *multiple exploits* could pose a significant threat in the future. Exploit kits contain multiple exploits targeting known vulnerabilities to gain entry to a computer, usually these kits only need to use one exploit to succeed but have a pool to choose from. For example the 2015 mobile 'Godless' malware contains multiple exploits that can gain root access to various versions of Android-based devices. Once the malware has achieved root access it can receive remote instructions to download other malicious software [158].

Some attackers however use exploits to target publically unknown vulnerabilities. These types of attacks are called zero-day and are growing more common. They can last up to 30 months [15] before the vulnerability is publically disclosed (Figure 3-7), and are often targeted at specific organisations such as the government. Additionally, these types of attacks can use multiple zero-day exploits to gain entry to the network, access information, propagate to other devices and perform malicious tasks. The 2010 Stuxnet worm for example used four separate zero-day exploits to gain entry and cause disruption to an Iranian nuclear power plant [17]. The first exploit targeted an automatic file execution vulnerability in a file shortcut of Microsoft Windows OS which was used to inject the worm via USB sticks into a computer system. The second targeted a shared print-spooler vulnerability using remote code execution (§3.3.4) which was subsequently used to spread the worm. The third and fourth targeted system-level privileges to gain control even when computers had been locked down to only allow specified software to run. This was the first threat to use so many publically unknown vulnerabilities.

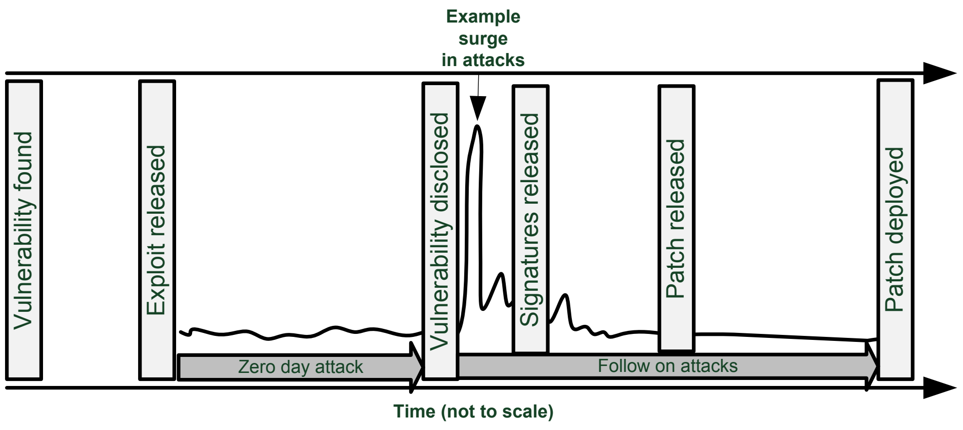


Figure 3-7 – Zero-day attack time line adapted from [15] and [14]

A more recent example in 2016 was a piece of malware named Trident [159] incorporated into the Pegasus spyware, which used three zero-day exploits to target iPhone devices. As shown in Figure 3-8 the first exploit targeted a vulnerability in the Safari WebKit at the application layer leading to memory corruption allowing the device to be infected when the user clicked onto a link. The second exploit targeted a kernel mapping vulnerability of the iOS at the core OS layer that leaked information allowing the attacker to calculate the kernel's location in memory. The third exploit targeted a vulnerability at the kernel layer that caused kernel memory corruption allowing the device to be silently jailbreaked so that surveillance software could be installed. The three exploits targeted different layers of the software stack (§3.2.1.3), and all three vulnerabilities needed to be exploited for the malware to be successful.

The damage caused by zero-day attacks does not stop at the original target. After a zero-day vulnerability is publicly disclosed, there can be a surge of attacks (Figure 3-7) within a few hours as other cybercriminals race to exploit the vulnerability before it can be blocked or patched by antivirus vendors. For example, shortly after the 'Heartbleed' and 'ShellShock' zero-day vulnerabilities were disclosed in 2014, between thirty and thirty five thousand follow on attacks were recorded [14] (Figure 3-7). A surge in attacks does not always happen this quickly but usually faster than vendors can deploy patches. Trident for example was patched in 10 days after it was disclosed [160], but many people still use old software that is no longer supported through patches such as Windows XP which accounts for around 18% of infections [161].

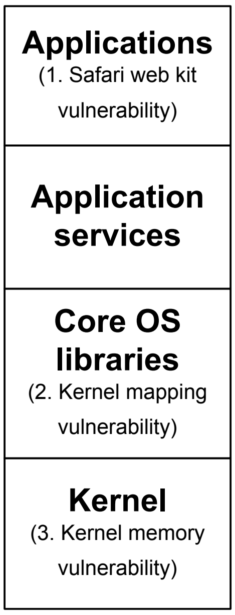

Applications
(1. Safari web kit vulnerability)
Application services
Core OS libraries
(2. Kernel mapping vulnerability)
Kernel
(3. Kernel memory vulnerability)

Figure 3-8 – Vulnerabilities in the software stack targeted by Pegasus exploits

3.3.4 Location of Vulnerabilities in the Software Stack

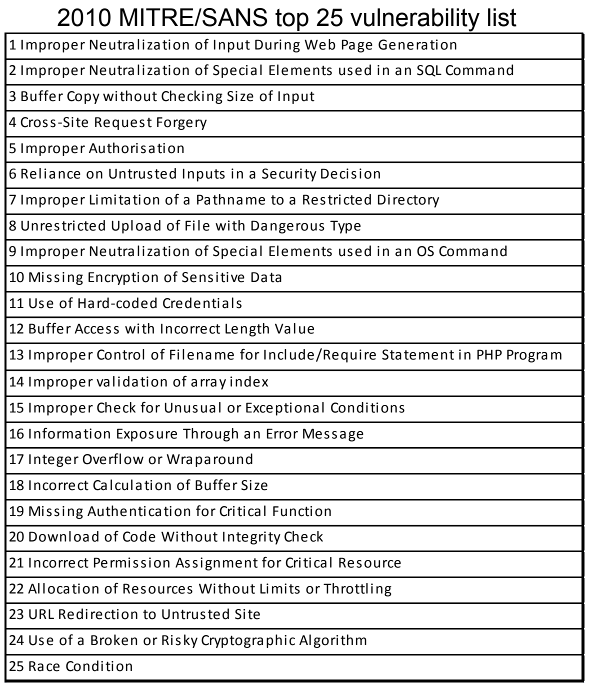
An analysis of the location of vulnerabilities within the software stack was included as part of an on-line published study by the author [2]. The 2010 top 25 most dangerous software errors reported by MITRE and the SANS institute was used during the study [162]. Figure 3-9 lists the vulnerability types and their location within the software stack. The purpose is twofold, firstly to stress the important fact that many types of vulnerabilities can affect multiple layers of the software stack and that malware can be injected or propagated using a single vulnerability at any layer. The actual exploit code however will be different for each unique vulnerability discovered. Secondly, vulnerabilities described here are referenced later in the chapter. A full description of the vulnerabilities and their location is included in the study [2].

Several of those listed relate to buffer or memory vulnerabilities (nos. 3,12,14,17,18) which can occur at any layer of the software stack and lead to the execution of malware. For example a *buffer copy without checking the size of the input* (no.3) often leads to the classic buffer overflow attack where the attacker writes data outside the bounds of the buffer to an adjacent location. This can change the behaviour of the program, overwrite local variables or a function pointer, or change a return address to point to malware.

In contrast, the leading vulnerability, *improper neutralization of inputs during web page generation* (no. 1) only affects Web applications but can lead to a range of attacks including an ideal entry point for malware. It occurs when untrusted inputs are not mitigated against. The most common attack method is via script injection, often called cross-site-scripting, where attackers inject JavaScript or other content into a web page that the web server application generates. The web page can then be accessed by other users, whose browsers execute the malicious script.

Code injection attacks often target online SQL databases by modifying improperly checked SQL queries (no. 2). Additionally they can be used in conjunction with memory corruption to redirect execution to the injected code, for example for malware, by modifying a code pointer in memory. Code reuse attacks can also be used, where instead of injecting new code small sections of legitimate code called gadgets are chained together to execute the exploit instead.

Other vulnerabilities that improperly deal with external inputs (nos. 4,6,7,9,15,16) can allow an attacker to by-pass security mechanisms. Cross-site request forgery (no.4) for example occurs when a web application insufficiently verifies requests by the user allowing an attacker to trick a user into making an unintentional request to the web server which is then treated as authentic. Other errors leading to security mechanisms being bypassed, include the setting of improper access, restrictions and permissions (nos. 5,8,19,20,21,22,25) and can lead to code execution for the propagation of malware.


2010 MITRE/SANS top 25 vulnerability list

1 Improper Neutralization of Input During Web Page Generation
2 Improper Neutralization of Special Elements used in an SQL Command
3 Buffer Copy without Checking Size of Input
4 Cross-Site Request Forgery
5 Improper Authorisation
6 Reliance on Untrusted Inputs in a Security Decision
7 Improper Limitation of a Pathname to a Restricted Directory
8 Unrestricted Upload of File with Dangerous Type
9 Improper Neutralization of Special Elements used in an OS Command
10 Missing Encryption of Sensitive Data
11 Use of Hard-coded Credentials
12 Buffer Access with Incorrect Length Value
13 Improper Control of Filename for Include/Require Statement in PHP Program
14 Improper validation of array index
15 Improper Check for Unusual or Exceptional Conditions
16 Information Exposure Through an Error Message
17 Integer Overflow or Wraparound
18 Incorrect Calculation of Buffer Size
19 Missing Authentication for Critical Function
20 Download of Code Without Integrity Check
21 Incorrect Permission Assignment for Critical Resource
22 Allocation of Resources Without Limits or Throttling
23 URL Redirection to Untrusted Site
24 Use of a Broken or Risky Cryptographic Algorithm
25 Race Condition

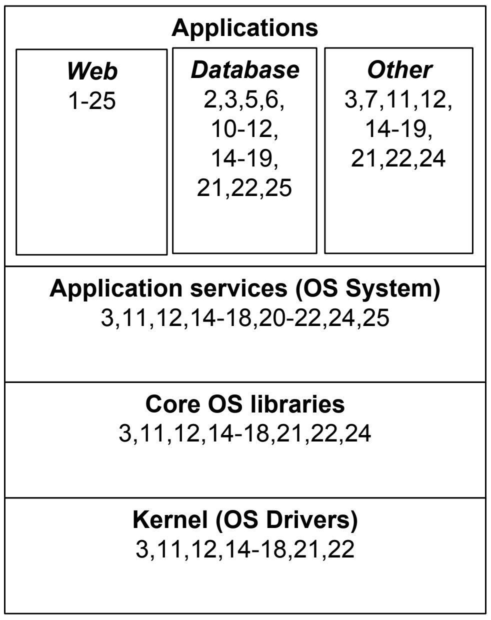


Figure 3-9 –Location of vulnerabilities in the software stack [162] [2]

3.4 Diversity

3.4.1 Diversity as a Security Mechanism

In the 1970s, N-version programming was proposed within the field of fault tolerance to increase the reliability of systems that used software [28]. It was known that identical software running on independent systems would fail in exactly the same way with the same inputs, so the idea was therefore to create N-versions of the software. Since then the concept of diversity within computer networks has expanded, with the majority of research focused upon applications such as improving communications [163-165], avoiding security attacks [35, 37, 39, 166, 167] [168], designing fault tolerant systems for harsh environments [169-172] improving test simulations [173], and in developing enabling technologies to support such concepts [127]. Interest in the use of diversity as a security mechanism within computing developed as computers became ubiquitous, attacks became more common, and the risks of a software monoculture were acknowledged [19], [18]. A biological perspective on diversity as a security mechanism was touched upon by Forrest [174] who recognised that diversity is an important source of robustness in biological systems, and its beneficial effects in computing systems should be investigated. Later, Crandall highlighted that biological diversity for computer security needed an ecosystem perspective [29]. There has been very little development in this research area until recently since the development of key enabling technologies (§3.2.1) such as dynamic software compilation, cloud computing, and virtualisation is only now making it possible to produce, disseminate and maintain the different versions of software needed [20].

Consequently there has been a renewed interest around diversifying software [175]. Diversity used as a security mechanism aims to make it more difficult for attackers to target multiple devices and networks during a single attack. The propagation of malware relies on being able to exploit the same vulnerability on multiple machines and so diversity makes attackers target each system individually. Without knowledge of the programs on a specific computer targeted attacks such as those using zero-day vulnerabilities become more difficult [14]. Diversity as a security mechanism is not just applicable to singular computers and their user software but may also manifest in other areas of defence such as network design [176] and network defence mechanisms. Diversity of network defensive techniques such as firewalls and intrusion detection systems is also related to the notion of defence in depth [177], which is a multi-layered defence strategy with complementary techniques to block, detect, monitor and remove suspicious activity to reduce the probability of a successful attack.

Diversity research relevant to malware and the security of computers, their interconnected network (as opposed to diverse network defence), and user software can be broadly partitioned into three categories:

1) Creating diverse code

2) Creating diverse systems

3) Measuring and analysing diversity

3.4.2 Creating Diverse Code

Creating diverse code involves practical code level manipulation techniques such as obfuscation, insertion, and randomisation of code, data, or binary files to generate different versions of software with the same functionality. Some techniques have been designed that can be applied at the source, or compilation and linking stage, usually prior to software distribution, whilst other techniques have been developed to be applied after distribution such as during installation, loading, or program execution [178]. Research has shown that diversifying software is possible using these techniques. Additionally, larger scale experiments have been carried out recently that prove diversity can actually be a viable method for wide scale use [179],[180]. There are some key types of attacks against which code level diversification is good at guarding. These include information leaks, memory corruption such as buffer overflows, as well as code injection and code reuse, the majority of which can allow the propagation of malware [181] [182] [178]. Vulnerabilities allowing these types of attacks are discussed in section 3.3.4.

3.4.2.1 Source Code Transformations

Generating different source code implementations has been widely researched within the field of fault tolerance, where the idea originated from N-version programming, and has often been a manual task [178]. Techniques for the automatic generation of source code are more recent. S*ource code transformations* is a technique used to automatically create a diverse set of program variants by undergoing different transformations given a baseline source code. Some of the transformations are purely random while others involve program analysis [179], and are all based on removing, adding or replacing statements in source code [180]. This technique has been demonstrated by performing diversity transformations on the server side of a client-server network [179]. Multiple cloned copies of the server software stack, called request handlers are generated to deal with incoming requests. Instead of using cloned copies, these multiple copies could all be different providing diversification.

3.4.2.2 Compiler Transformations

Compilers are used to translate high level source code into low-level machine code automatically. Some diversity techniques take advantage of this process already in place by extending existing compilers to automatically diversify machine code.

The *NOP insertion technique* [33] [34] works by randomly inserting non-alignment, no operation (NOP) instructions during compile time giving a large number of program variants. A NOP is an instruction that the processor fetches and executes without any effect on the processor register or machine memory. Although adding NOP instructions can positively impact diversity it can also negatively affect the performance of the generated binary file.

Another proposed method utilises the compiler optimisation algorithm. Compilers usually try to find the best binary implementation to give optimum performance out of numerous possibilities. Instead of choosing the best solution, the *alternative compiler solutions* could also be used to generate alternative unique binaries [110] [183] [20].

3.4.2.3 Address Space Randomisation

Address space randomisation randomises the locations of data and code objects in memory [184]. Address space layout randomisation (ASLR) is one of the most well known diversity techniques which randomises the layout of a section of memory for an executing program. A compiler equips the code for base address randomisation and then the operating system changes the virtual memory addresses at which the code is loaded [178]. The idea is to provide some protection from memory vulnerabilities without needing to remove them from the system such as those involving code injection buffer overflow attacks (§3.3.4). Since the randomisation on each machine is different, any exploit that depends on a specific relative memory address will generally fail. ASLR is in widespread use within operating systems such as Google Android, Linux, Microsoft Windows, and iOS [185] [186].

3.4.2.4 Data Space Randomisation

Data space randomisation (DSR) [187] is where the representation of different data objects or code in memory is randomised. Data space randomisation can be implemented in a variety of ways [185]. One way to modify the data is through encryption such as to logically XOR each data object in memory with a unique mask and then decrypt it before it is used. In the case of a memory vulnerability attack for example using code injection (§3.3.4), the attacker would only be able to write a random value into memory rather than the intended value [187].

3.4.2.5 Instruction Set Randomisation

Instruction set randomisation (ISR) creates a unique set of synthetic instruction sets randomly for each computer such as for the Intel x86 machine code [188] [189] [190] [191]. Translation from the synthetic instruction set to the instruction set of the actual target computer requires an interpreter or just-in-time compiler. Code injection attacks utilise the synthetic instruction set and therefore are unable to penetrate into the system.

3.4.2.6 Executable Code Randomisation

Randomisation techniques such as ASLR and ISR that rely on the 32-bit and 64-bit architectures can potentially be open to brute force attacks [168] [192] where an attacker has many attempts with different combinations until successful. *Executable code randomisation* is where executable code is broken into many functional blocks that can be shuffled in memory just before execution [181]. The number of unique permutations is higher than ASLR. With 500 blocks there are '500 factorial' permutations making a brute force attack difficult.

Another technique named *In-place code randomisation* [193] is based on the randomisation of the code sections of binary executable files. Firstly code is extracted from the executable binaries using a disassembler, and then transformations are conducted on small sections of code such as substitution with functionally equivalent alternatives, reordering of instructions, and reordering of register preservation code.

3.4.3 Creating Diverse Systems

Creating diverse systems, involves the creation of diverse networks or algorithms at a higher level of abstraction to analyse their behaviour in relation to either diversity alone, or the effectiveness of diversity against an attack model. There have been relatively few research papers associated with diversity algorithms to analyse overall network behaviour. Those that do exist are very wide ranging in their methodology and are often for a specific topology or purpose making them difficult to compare. Additionally some are preliminary studies or ideas and therefore have limited results in which to analyse the effectiveness of diversity adequately. These diverse systems are described below.

3.4.3.1 Colouring Algorithms

Colouring algorithms, which are widely investigated in graph theory [194] (§4.2.4), have been proposed [35],[36],[37] to try to minimize the number of neighbours running the same software package. In this type of algorithm each colour is assumed to be a different software package where each node in the network runs a single software package but each can be the same, or different. Colouring algorithms however tend to require a global perspective of the network, where knowledge of all the links between nodes are needed in order to assign colours. In an ad hoc network where, nodes are moving, and links between them are constantly changing these types of algorithms would not necessarily be practical. Additionally the compulsory assignment of software packages to nodes would be difficult in these changing scenarios. Colouring algorithms proposed for software diversity involve a fixed number of colours, usually 3 or 4 [35],[37] and are based on network topologies with fixed communication links. O'Donnell [35] used a network topology generated from email traffic logged over a fixed time period to investigate 4 types of colouring algorithms. The first is where each node randomly chooses a colour which remains fixed. The second is where each node at random intervals analyses its neighbours and chooses a new colour for itself if the current one is used frequently. The third allows pairs of nodes to swap their colours, and the fourth combines self updating and swapping which was found to produce the best colour distribution across the network. An attack was simulated by selecting one colour to be vulnerable with the goal of switching every node in the network to the vulnerable colour. This was achieved by introducing malicious nodes to lie about different aspects of the algorithm such as their colour or proposed swap. The analysis found that the fourth algorithm with the ability to switch between the two methods made it more difficult for the attacker because it was unclear which method the targeted node was going to carry out and proposed that diversity algorithms should contain diversity within them as well.

Yang [37] focused on sensor networks by partitioning sensor nodes into cells of either tessellating hexagons with three possible colours or squares with four possible colours (Figure 3-10). The links between sensors were modelled using graph theory. Each hexagon or square contained sensors with the same colour. Once a cell colour is compromised more than one sensor is infected, with the intention that a potential worm attack could be quarantined. A worm attack was simulated using a standard Susceptible - Infected model (§4.3.2) where each sensor was able to adopt either of two states: susceptible where it is susceptible to the worm but not yet infected, and infected where it has been infected by the worm. With four colours arranged in squares as shown in figure 3-10, the minimum distance between the same colour is the length of the cell (L). The number of connected sensors (which are assumed to be fixed in location) with the same colour is dependent upon L and the transmission range of the sensors R. When R is less than L the infection can be quarantined to a single cell so that the total number infected is dependent upon the number in the infected cell and their location of being within transmission range of each other. For a non-diverse system all sensors could potentially become infected (assuming R is large enough between individual sensors) since it would be equivalent to all cells having the same colour.

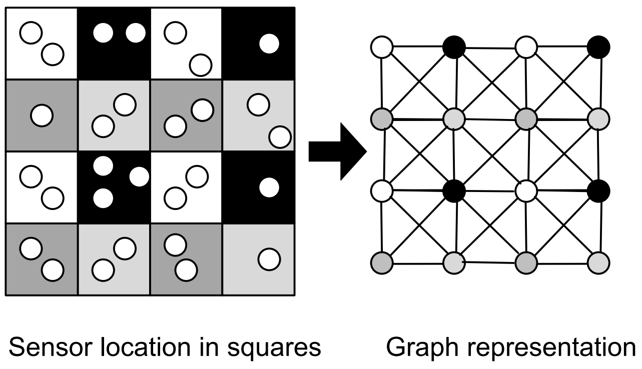


Figure 3-10 – Four colour, colouring algorithm [37]

3.4.3.2 Epidemic Based Attack Models

Epidemic models are widely used within ecology to study the spread of diseases and have also been used to model the spread of malware in computer networks (§4.3). Introducing software diversity into epidemic models has been considered by Hosseini [195] who used a scale free network topology often considered as a common structure of the Internet, together with a discrete-time deterministic SEIRS epidemic model with *L* diverse software packages. The SEIRS epidemic model has 4 states: Susceptible, Exposed, Infected, and Recovered, where once recovered the individuals become susceptible again to the same attack. The deterministic equations are modified such that by applying L diverse software packages, the rate of infection of propagation λ is adjusted to λ/L. Another similar model uses networks and epidemics to model the diversity and malware propagation of nodes [196]. The assumption is that compilers with "diversity engines" produce many different executable software variants to generate diverse node types. There are L node types and N nodes with M malware. For homogeneous mixing networks, the total number of nodes infected is MN/L. These equations assume that maximum diversity is being achieved so that the number of different software packages or node types are equally distributed, thus keeping the equations simplified. The colouring algorithm used by Yang [37] (§3.4.3.1) also included an epidemic Susceptible - Infected model to analyse how a worm might propagate in response to the diversity scheme developed.

3.4.3.3 Biological-Inspired Models

Genetic programming [197] is a large topic of research in which computer programs are encoded as a set of genes (§2.2.2.1.4) that evolve using an evolutionary algorithm to find programs that perform well against set criteria. Usually many programs are tested over lots of generations until a solution converges. It has been proposed that the parameters used to control how diverse the programs are can be used to develop a method for generating a pool of diverse programs (rather than converging to a single solution) [198]. It is unclear from the literature whether this method has been practically tested.

Holtschulte [199] describes a model inspired by the immune system of how computers on a network distribute and share patches to repair variants of software in response to an attack. The diversity being considered here is the diversity of the software patches generated by each node in response to an attack, rather than the diversity of the original software in the network which is the same. Nodes attempt to generate their own repairs or send requests to neighbours for software variants until a resistant variant is found. The research showed that the network topologies that allowed the largest amount of software sharing had the least diverse software variants, but were also the quickest to resist new attacks, presumably because when a resistant variant was found it could be distributed more quickly.

Another ecosystem related model, but does not fit into the epidemic model category, is that of Bi-partite relationships (Figure 3-11, individuals categorised into two sets with relationships between them). These observed relationships within ecosystems have been used to introduce ecological based diversity ideas into client-server software architectures where one set represents the servers and the other the clients connected with relationships as shown in Figure 3-11 [175]. The project proposed (but not simulated or implemented at the time of this writing) the definition of evolution rules to generate diversity in the client server networks. They highlighted that the rules should consider a trade-off between providing more servers for redundancy and increased cost.

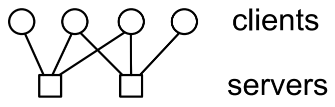


Figure 3-11 – Client–server bipartite graph

3.4.3.4 Other Models

The problem of deciding which software variants to assign to nodes in a network has been considered from an optimisation perspective so that the [200] overall network resiliency is optimised when placing diverse variants at routing nodes within a cloud based network consisting of routing nodes and client nodes (Figure 3-12). An attack model assigns a probability of an attacker being able to exploit a vulnerability for a particular variant within a constrained time frame. Subsequently any routing node in the network with this variant becomes compromised. The resiliency metric was computed based upon the number of surviving client-to-client connections offered by the network when under attack.

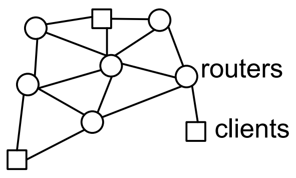


Figure 3-12 – Diversity assignment within a cloud

Diversity for the prevention of software piracy has also been proposed as an idea (but not simulated or practically tested) [167]. The model suggests two levels of diversification. Firstly each distributed copy is different, and secondly each installation of a specific copy is different. It is proposed that a database keeps track of the legitimate copies. When a user requests an update, it is tailored to each unique copy.

Instead of designing networks where devices differ from one another in terms of software, methods have been proposed to create diverse versions of software internally on a single device with a monitor analysing the outputs (Figure 3-13).

When the outputs differ an attack is assumed and the variants are reset to previously known good states. Using redundant programs has been widely studied within the field of fault tolerance. Using different versions of commercially available software has been investigated [201] as well as automatically generated software [202].

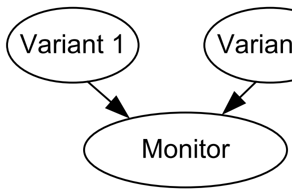


Figure 3-13 – Internal device level diversity

The use of virtual machines to create internal device diversity has also been proposed [203] [204]. A device level system named ChameleonSoft [203] partitions software programs into small chunks which run within separate capsules. A capsule is described as a smart micro sandbox/virtual machine encapsulating a single active code variant as part of a running application. The capsules manually or automatically use a pre-generated set of functionally equivalent variants which are intelligently shuffled at runtime to confuse the attacker. Confusing a targeted attacker can make it difficult to establish what vulnerabilities may be present or what resources are being used in a specific device at any given time.

3.4.4 Measuring and Analysing Diversity

A broad range of techniques have been proposed for measuring and analysing diversity of computer networks and are achieved either through the gathering of data, or through the use of diversity metrics. Some metrics are single statistical values, whilst others are multi-dimensional. Other techniques do not measure diversity directly but analyse other important properties such as the commonality of vulnerabilities between software. Descriptions of the methods used are detailed below.

3.4.4.1 Shannon Index

The Shannon Index (or Shannon Entropy) is used to measure species and genetic diversity in natural systems (§2.3.2.2.3). It has also been used for analysing diversity in a computer network after the response to an attack where diverse software patches and repairs were generated [199]. It has also been used to measure diversity (discussed as entropy) of a bipartite graph interconnecting hosts and vulnerabilities within a game theoretic model [40].

3.4.4.2 Number of Variants

A popular metric is just to simply use the number of different software variants. Hosseini [195] and Hole [196] both use the parameter L (number of software variants) within their epidemic models to describe the diversity. This metric is on a par with diversity richness (species, genotypic, or allelic) which is used as a diversity measure in natural systems (§2.3). This metric however does not take into account the distribution and number of each type used.

3.4.4.3 Resiliency

The Diversity Assignment Problem [200] as described in 3.4.3.4 was presented to specify how to optimize overall network resiliency when placing diverse variants at routing nodes. The resiliency metric was used as a measure of diversity and was computed based upon the number of surviving client-to-client connections offered by the network when under attack.

3.4.4.4 Multi-dimensional Properties

Measuring diversity has also been considered from a multi-dimensional perspective. It was proposed that diversity should be measured by considering 6 dimensions as shown in Figure 3-14 representing the functional capabilities of the network architecture [38]. Dimensions proposed were: operating systems, communications medium, service model, network protocol, transport protocol, and routing mechanism. The distance between network elements reflects their diversity, for example the distance between OSs Linux and Windows would be large and the distance between Network protocols IPv4 and IPv6 would be small. A point in the multi-dimensional space would be representative of the software stack on a unique device. Three dimensions have also been proposed representing aspects that are orthogonal to each other such as hardware, operating system, and application software.

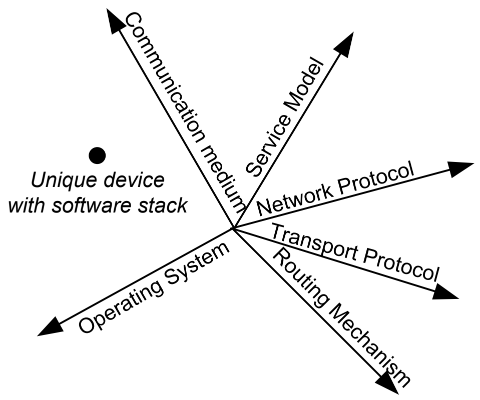


Figure 3-14 – Multi-dimensional diversity metric [38]

This is combined with the Shannon Entropy discussed in section 3.4.4.1 above so that the final entropy measure is the sum of the entropies of each dimension [39].

3.4.4.5 Analysing Software Binary Files

An attempt at measuring the existing diversity of systems has been carried out by collecting data and analysing variants of software binary files [205]: Three metrics were proposed to measure diversity; 1) The probability of a successful targeted attack which is based upon the number of instances of the most frequent variant of a given file and the total number of instances of that file. 2) The ratio of the number of variants to the total number of instances of all the variants of a file. The bigger the ratio, the more variants the file has and subsequently more attacks are needed to compromise all the instances of the file. 3) The coefficient of variation. This is the ratio of the standard deviation to the mean. If the ratio is small the instances are distributed uniformly.

3.4.4.6 Common Vulnerabilities

Another approach to measuring diversity in current software has been through the analysis of vulnerabilities. One study analysed the commonality of vulnerabilities of 11 different operating systems over a 15 year period [206]. Data was extracted from the National Institute of Standards and Technology (NIST) National Vulnerability Database (NVD). Every pair of OSs were analysed for common vulnerabilities. Common vulnerabilities were found to exist, and not surprisingly, there were more common vulnerabilities between different versions of the same OS such as between Windows 2008 and Windows 2003 than between completely different OSs. It was also found that one vulnerability affected nine OSs, which related to a well known problem in the design of the TCP leading to denial of service problems. This means that vulnerabilities introduced at the design stage can propagate into the code no matter how it is implemented. In general though the commonality of vulnerabilities were deemed sufficiently low enough to declare that building a system with diverse OSs may be a useful security technique.

Another study focused on application software during a one year period [207]. The research highlighted that the majority of the software products, including those providing the same service and those that ran on multiple operating systems, either did not have the same vulnerability or cannot be compromised with the same exploit. However it was noted that although different distributions of the same product could not be attacked by the same exploit code they had at least an 80% chance of suffering from the same type of vulnerability. In general, again it was concluded that using different commercial software applications could be an effective security technique.

3.4.5 Diversity Open Research

There are currently open research questions regarding where and when diversity should be introduced [208], or whether it should be applied everywhere at all levels and layers. There are currently a wide range of ideas and methodologies proposed for network level diversity often targeted at a specific topology or purpose, however the majority are limited to conceptual ideas and minimal analysis. Despite the growing research in this area there is still a large gap in understanding the actual benefits of diversity as a security mechanism [209], particularly from an ecological perspective.

There is no well defined metric for measuring diversity within computer networks. A broad range of techniques have been proposed but none capture both the granularity of diversity at different layers of a software stack and the distribution of diversity at the same time. Additionally none consider practical constraints associated with compatibility issues, user preferences or devices unable to participate due to hardware limitations.

The tools and technologies enabling wide-spread software diversity to become a reality are slowing merging together, however many of the methodologies are still early stage proposals and larger scale experiments analysing their practical effectiveness are still limited or yet to be undertaken.

3.5 Summary

This chapter discussed three areas of technology and research. The first focused on the Internet. Software and protocols of the future are likely to remain modularised, perhaps with even more functionality and choice, particularly with the continuously evolving open source paradigm. Software stacks can be partitioned into four main layers, although these layers can be broken down further to define specific functionality. The modularity of software, together with improved virtualisation, and better automated software generation and dissemination, could allow individually tailored software stacks to be dynamically created providing a powerful tool for enabling diversity. The Internet will comprise different topologies utilising a multitude of communication protocols depending upon the devices and their application. There will be more localised peer-to-peer communication, with ad hoc networks featuring more prominently in the future.

The second section on cyber security focused upon malware which is prolific in monoculture environments since it can spread over networks taking advantage of software, such as widely used operating systems with the same vulnerability. Malware can exploit a multitude of different types of vulnerabilities which can appear at different layers of the software stack. Publically unknown vulnerabilities are particularly dangerous as they are used in zero-day attacks, where the damage can go unnoticed for long periods of time. The use of multiple exploits across layers poses a significant threat, especially if they are targeting zero-day vulnerabilities.

The third section explored the current state of research of diversity within computing systems. Diversity as a security mechanism increases the difficulty for attackers to target multiple devices and networks during a single attack. It prevents the attacker from having detailed knowledge of each computer, forcing them to be targeted individually, and in turn increasing the difficulty of propagating malware. Research has shown that diversifying software is possible through diversification at the code level. Code level diversification however does not consider the dynamics of diversity at multiple layers of the stack or the dynamics at a network level in the face of an attack where it may not be possible for all devices to apply a diversity technique. Diversity analysis at the network level allows the effects of the creation and distribution of diverse code to be analysed using different methods, both from centrally generated sources and via distributed methods. It also enables the resistance of a network to be simulated under a range of different conditions in the face of a malware attack. There are currently open research questions regarding diversity. There is still a large gap in understanding the actual benefits of diversity as a security mechanism and particularly from an ecological perspective. There is no well defined metric for measuring diversity within computer networks. Those proposed do not capture both the granularity of diversity at different layers of a software stack and the distribution of diversity simultaneously. Additionally none of the research considers practical constraints associated with compatibility issues or user preferences.

Chapter 4

Modelling

4.1 Introduction

This chapter details the modelling of peer-to-peer communication networks with an emphasis on mobile ad hoc networks, highlighting their comparability with natural systems. Compartmental based methods for modelling the propagation of malware at a system level are reviewed. These epidemic models are widely used for modelling the spread of diseases within natural systems. Details regarding the properties of the deterministic and stochastic SIR (Susceptible, Infected, Recovered) models are given which are used by development work in later chapters. Agent-based epidemics are considered for modelling malware at the individual level as well as infection genetic models where matching algorithms are used to match pathogens to hosts. These principles are also used by development work in later chapters.

4.2 Modelling Ad Hoc Networks

Devices utilising direct peer-to-peer communication, particularly those within ad hoc networks can be compared with natural systems since the devices, for example mobile phones, interact closely with humans following their mobility and interaction patterns [210]. Ad hoc networks are expected to become more prominent in the future Internet either as a separate topology or integrated with sensor and peer-to-peer overlay networks (§3.2.3), so the focus of this research will be limited to networks which are ad hoc. There are a number of methodologies for modelling such networks which are summarised in the following sections.

4.2.1 Mobility Models

Mobility models consider the movement patterns of devices within a mobile network and can be used to visualise individual or aggregated travel paths. There are two types of mobility models generally used: traces and synthetic models [211]. Traces are generated from observed data and can provide accurate information when using large datasets. Synthetic models attempt to represent the mobile behaviour realistically without the need for trace data. A number of synthetic models exist for ad hoc networks [211]. One model that is used in many simulation studies is the Random Waypoint model [212] [213] [214] [215] [216] [217]. It was developed to represent the mobility patterns of people with mobile devices within a confined environment such as a room [218]. As pictured in Figure 4-1 each node starts by remaining stationary for *pause* time seconds. It then selects a *destination* point within a bounded rectangular area. The node then moves to that destination at a selected *speed*. Values for the destination, speed and pause time are chosen independently and at random from uniform distributions between upper and lower bounds. When the destination is reached the cycle of pause, choosing a destination, and moving at speed is repeated until the end of the simulation.

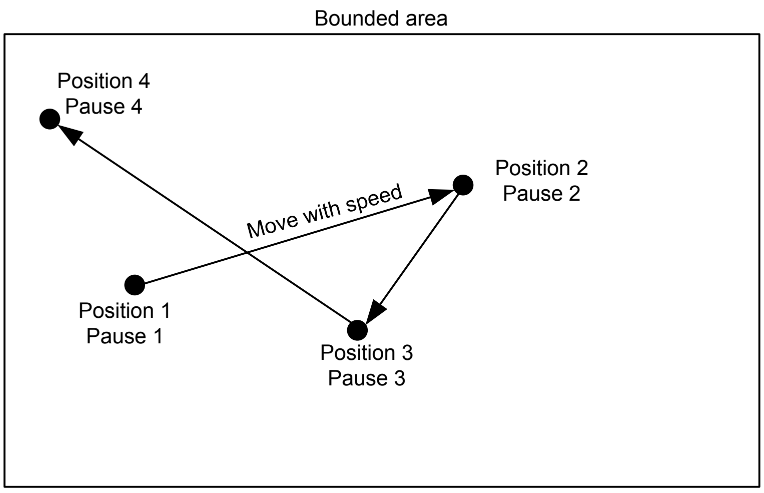


Figure 4-1 – Random waypoint mobility model

4.2.2 Simulators

Mobility models are often integrated into event-based simulators to allow the detailed modelling of new or improved protocols such as those necessary to provide ad hoc routing. Simulators such as Opnet, NS3, and QualNet [219] are used to model detailed characteristics at different layers of the network stack under realistic conditions. As well as mobility models the simulators can include other realistic characteristics such as radio transmission, buffer space for the storage of messages, and data traffic models. A downside of these simulators is that much effort is required to learn the details of the simulator architecture and programming language. These types of simulators can be seen as a type of agent-based model with very detailed characteristics for the agents (nodes) in the network.

4.2.3 Agent-Based Models

Agent-based Models (ABM) attempt to capture the complexity of individual behaviour and have been widely used across a growing number of fields [220]. Such models, however, do not necessarily need the detailed characteristics used within simulators, and can follow a set of simple rules at a higher level of abstraction, sometimes generating emergent behaviour [221]. ABMs allow a wide choice of design parameters and rules making each model different but tailored to each research question. They have been used for modelling ad hoc networks [219] [222] [223] and use software such as Netlogo and Matlab to model high level behaviour.

4.2.4 Graph Theory

Graph theory is another technique for modelling communication networks. A graph is made up of vertices (nodes or points) which are connected by edges (links or lines). Graph theory is used to measure properties such as the degree distribution (probability that a vertex chosen uniformly at random has degree k, where degree k is the number of edges connected to a vertex) or clustering coefficient (measure of how strongly nodes in a graph cluster together). Graph theory has been used to study the architecture of the Internet [224] and analyse the behaviour of routing protocols of ad hoc networks [225] [226].

4.2.5 Homogeneous Mixing Models

When the networks to be analysed are considered to be large, homogeneous mixing models can be used to model the network as a whole entity. Here it is assumed the system is the average of the individual nodes where nodes make contact with each other in a peer-to-peer fashion at random. Such assumptions originated from the modelling of infectious diseases within human populations using deterministic and stochastic methods and have since additionally been used to model the propagation of malware within mobile wireless networks. This is discussed in more depth in the next section (§4.3).

4.3 Epidemic Modelling of Mobile Malware

4.3.1 Mathematical Models of Epidemics

There are two main types of mathematical models that are used to describe the spreading characteristics of epidemics: *deterministic*, and *stochastic* which can be used to make system level predictions [227] [228]. The deterministic model always performs the same way for a given set of initial conditions and is used to model large populations (or networks), whereas for the stochastic model randomness is present and the output result is a probability distribution. Stochastic models are able to model smaller populations and are often considered to be more realistic.

A mechanism that links these models is the concept of *compartments* where individuals are assumed to be in one of a number of different compartments (states or classes) at any given time. These compartments represent the individuals' health status with respect to the disease. For example the population could be divided into those who are Susceptible (S), those who are Infected (I) and those who have Recovered (R). For both the deterministic and stochastic based models, *the number within each compartment* is simulated as the epidemic progresses. Malware epidemic models have used a multitude of different compartments. The SI variant has been used for modelling a mobile phone virus using two compartments where there are no recovery mechanisms [229]. The SEIS model includes an extra Exposed (E) compartment as there may be an incubation period before the virus attacks [230]. The extra susceptible (S) in the model name denotes the fact that instead of recovery, the devices become susceptible again. The SEIRD model was proposed to model virus propagation specifically via Bluetooth and MMS to investigate the

Commwarrior virus [231]. The additional Dormancy (D) compartment represents the condition when the virus drains the battery by sending out many MMS messages. The author of this thesis proposed a SEPTICOX model incorporating conditions where the phones were switched off or offline for Bluetooth based networks which required a number of additional compartments: Prevented (P), Treated (T), Contained (C), Offline (O), eXposed off-line (X) [3].

Details regarding the properties of the deterministic and stochastic SIR models are described in the following text which are used as a reference for development work in chapter 6 of this thesis. Note that the work assumes that once devices have fully recovered from a particular malware (through patching or anit-virus tools) they cannot be re-susceptible to the same malware so that the 'R' compartment is designated the end state. Thus the closely related SIRS model [228], where there is no end state (compartment) and re-susceptibility can occur following recovery, has not been detailed within the background material (see chapter 6)

4.3.2 The Deterministic SIR Model

In the deterministic SIR [228], where individuals mix homogeneously (§4.2.5), and the population is considered to be large, the law of mass action is applied to the rates of transmission between two compartments where the rate of interaction is proportional to the product of the numbers in each compartment. The transition rates from one compartment to another are mathematically expressed as derivatives, hence the model is formulated using differential equations.

4.3.2.1 Model Equations

The basic *SIR model* was initially developed by Kermack and McKendre [232] and is comprised of three compartments as shown in Figure 4-2. The S compartment represents those that are susceptible to a disease or virus but not yet infected, I represents those that are infected and infectious with the disease, and R represents those that have recovered from the disease. N defines the total population size and is assumed to be fixed.

$$\begin{tikzcd} S \arrow[r, "\beta I/N"] & I \arrow[r, "\gamma"] & R \end{tikzcd}$$

Figure 4-2 – SIR model

β is known as the infection rate (or effective contact rate) and is defined as [233]:

$$\beta = \tau c \tag{4-1}$$

Where τ is the probability of an infection given contact between a susceptible and an infected individual, and c is the average rate of contact between susceptible and infected individuals. The rate at which those susceptible become infected is attributed to the proportion of the population who are already infected I/N multiplied by the infection rate β.

γ is the rate of recovery of an individual, and can also be written as:

$$\gamma = \frac{1}{d} \tag{4-2}$$

Where d is the duration of the infection.

The model is described using differential equations, where the transition rates from one compartment to another are expressed as derivatives:

$$\frac{dS}{dt} = -\frac{\beta SI}{N} \tag{4-3}$$

$$\frac{dI}{dt} = \frac{\beta SI}{N} - \gamma I \tag{4-4}$$

$$\frac{dR}{dt} = \gamma I \tag{4-5}$$

4.3.2.2 *Discrete Model*

The model can also be represented in discrete form using difference equations, where the number in each compartment at the next time step $(t + 1)$ is formulated by the rates and the number in each compartment at the current time step (t). This approach is convenient for computer simulation of the model:

$$S(t+1) = S(t) - \frac{\beta I(t)S(t)}{N} \tag{4-6}$$

$$I(t+1) = I(t) + \frac{\beta I(t)S(t)}{N} - \gamma I(t) \tag{4-7}$$

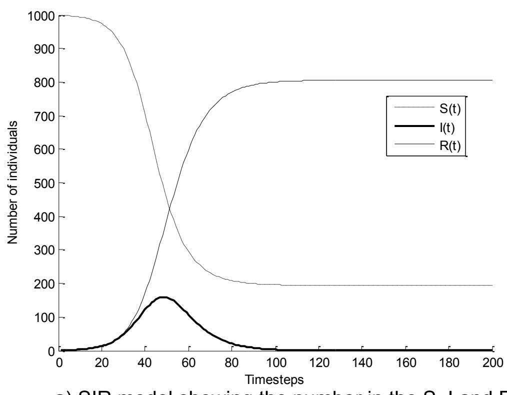
$$R(t+1) = R(t) + \gamma I(t) \tag{4-8}$$

The total population size (N) is assumed to be fixed so that:

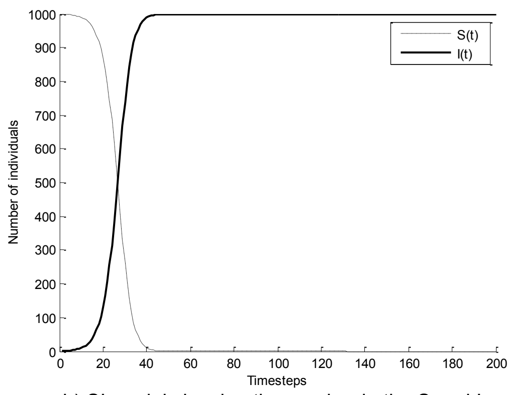
$$N = S(t) + I(t) + R(t) \tag{4-9}$$

4.3.2.3 *Deterministic Epidemic Example*

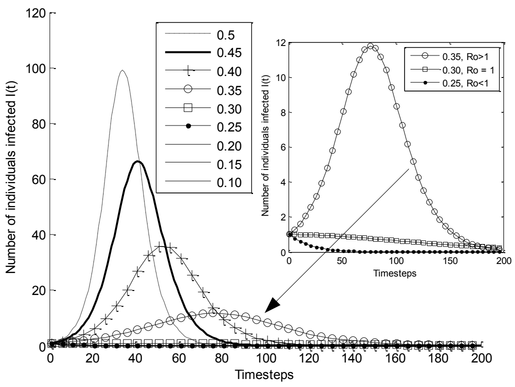
To illustrate the mathematics an example showing an epidemic following the SIR equations is given through simulation in Figure 4-3 (a) .When an epidemic occurs susceptible individuals become infected and move to the infected compartment faster than infected individuals can recover (where $\beta > \gamma$) creating a peak of infections. Figure 4-3 (a) shows the number within each compartment as the epidemic progresses. The SIR model was simulated from difference equations (§4.3.2.2) using Mathworks Matlab. The number infected $I(t)$ increases and then falls as recovery takes place. To show an epidemic occurring the condition $\beta > \gamma$ needs to occur as stated above. In this example values are chosen to represent this condition where $\beta = 0.3$ and $\gamma = 0.15$. With no recovery, the SIR model reduces to two states S and I, which is also known as the *SI model*. Under this condition all of those susceptible will eventually become infected, and stay infected as shown in Figure 4-3 (b).



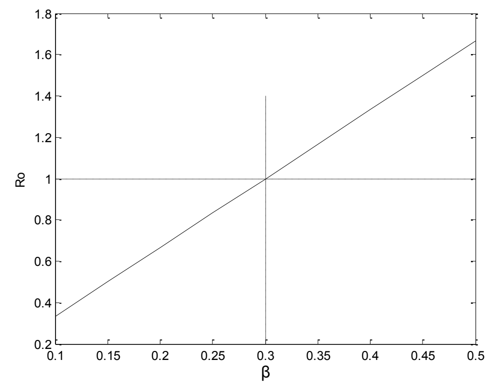
a) SIR model showing the number in the S, I and R states where β=0.3,γ=0.15. All those susceptible become infected and eventually recover.



b) SI model showing the number in the S and I states only where, β=0.3, γ=0 and all those susceptible eventually become infected.



c) SIR model showing the number in infected for varying values of β.



d) Corresponds to graph c showing values for the varying values of β.

Figure 4-3 – Deterministic SIR model using difference equations.

4.3.2.4 End Time of the Epidemic

When there is no chance of recovery, the dynamical equations can be simplified so that Equation (4-4) becomes:

$$\frac{dI}{dt} = \frac{\beta SI}{N}, \qquad where\ N = S + I \tag{4-10}$$

where the time at which the epidemic reaches its final state has an analytical solution. Substituting $S = N - I$ into Equation (4-10) results in a *logistic equation* for I [228]:

$$\frac{dI}{dt} = \frac{\beta I}{N}(N - I) \tag{4-11}$$

With a solution [228]:

$$I(t) = \frac{I(0)N}{I(0) + (N - I(0))e^{-\beta t}} \tag{4-12}$$

The end of the simulation T_1 is specified to occur when the number infected is within 1 of its final value $I(t) = N - 1$ so that Equation (4-12) becomes:

$$N - 1 = \frac{I(0)N}{I(0) + (N - I(0))e^{-\beta T_1}} \tag{4-13}$$

Rearranging for T_1:

$$T_1 = \frac{1}{\beta} ln\left[\frac{(N-1)(N - I(0))}{I(0)}\right] \tag{4-14}$$

4.3.2.5 The Reproduction Number R0

A key metric used in epidemiology to determine whether a disease will spread or not is the reproduction number. It is defined as the number of secondary cases produced by a single infection within a susceptible population [233]. The reproduction number can be derived by considering that an *epidemic occurs* if the number of infected individuals increases, where:

$$\frac{dI}{dt} > 0 \tag{4-15}$$

Substituting in Equation (4-4) becomes:

$$\frac{\beta SI}{N} - \gamma I > 0 \tag{4-16}$$

$$\frac{\beta SI}{N} > \gamma I \tag{4-17}$$

$$\frac{\beta S}{N\gamma} > 1 \tag{4-18}$$

At the outset of an epidemic, where $t = 0$, everyone except the initial infected individual is susceptible. At this point S can therefore be approximated to N, and the equation simplifies to:

$$\frac{\beta}{\gamma} > 1 \tag{4-19}$$

$$\mathit{The\ Reproduction\ Number} \quad R_0 := \frac{\beta}{\gamma} \tag{4-20}$$

For R_0 values greater than 1 an epidemic occurs, and for R_0 values equal to or less than 1, the epidemic dies away. Figure 4-3 (c) shows the number of infected individuals for varying values of β when γ is fixed at 0.3. When $R_0 > 1$ ($\beta > 0.3$), there is an epidemic as the number of infected individuals increases above the initial value of 1. As the value of β is reduced, the peak value of the infection is reduced, the time of the peak moves to the right and the spread of the infection increases. When $R_0 = 1$ ($\beta = 0.3$) the epidemic dies away as the number of infected individuals never goes above 1 as shown by the sub-graph in Figure 4-3 (c). When $R_0 < 1$ ($\beta < 0.3$) the infection dies away even faster, with the corresponding curves reduced to zero very quickly. Figure 4-3 (d) shows the corresponding R_0 relationship with β, as β is varied between 0.1 and 0.5 (as given in Figure 4-3 (c)), additionally showing R_0 when $\beta = 0.3$ for a fixed $\gamma = 0.3$.

4.3.2.6 *The Balance Equation*

Another important attribute of an epidemic is its final state, which is the total fraction of the population that was infected. A balance equation [234] can be derived that describes the final state of the system when t→ ∞, by dividing the differential equations of the SIR model (Equations (4-3) and (4-5)).

$$\frac{dS}{dt} / \frac{dR}{dt} = -\frac{\beta SI}{N} / \gamma I$$

$$\frac{dS}{dR} = -\frac{\beta S}{N\gamma} = -R_0 \frac{S}{N} \tag{4-21}$$

This implies the solution:

$$S(t) = S(0)e^{-\frac{R_0}{N}R(t)} \tag{4-22}$$

During an epidemic those within the infected state will eventually move to the recovered state, so at the end of the epidemic there will only be those still susceptible $S(\infty)$, or recovered $R(\infty)$. This means that:

$$S(\infty) = N - R(\infty) \tag{4-23}$$

Assuming that at $t = 0$, no individuals have yet recovered, so that $R(0) = 0$, then $S(0) = N - I(0)$, and Equation (4-22) can be rearranged to:

$$N - R(\infty) = (N - I(0))e^{\frac{-R_0}{N}R(\infty)} \tag{4-24}$$

which can also be expressed as a fraction of N:

$$1 - r(\infty) = (1 - i(0))e^{-R_0 r(\infty)} \tag{4-25}$$

Where $R(\infty) = r(\infty)N$

Solving for $r(\infty)$ determines the fraction that were infected at the end of the epidemic. This equation can be solved numerically using the approximation that $i(0) = 0$, and is graphed in Figure 4-4.

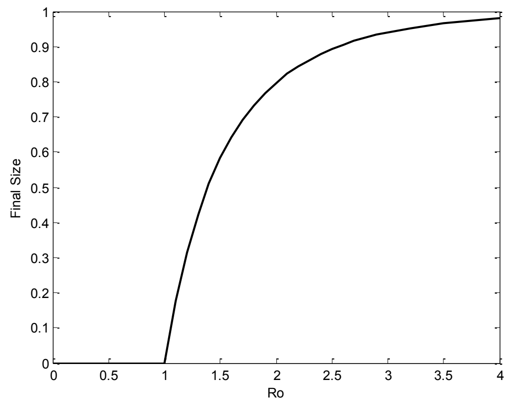


Figure 4-4 – Final size of the epidemic as a fraction of the population size

4.3.3 The Stochastic SIR Model

Stochastic SIR models are often described using discrete or continuous time Markov chains or stochastic differential equations [235]. A probabilistic model takes into account that there may be some element of randomness in at least one of the parameters. Predictions from that model are probability distributions, for example distributions of the possible numbers of those susceptible, infected or recovered. The Markovian *standard stochastic SIR epidemic model* [228] [234] assumes a closed homogeneous uniformly mixing community just as for the deterministic general epidemic model (§4.3.2). An important feature of the stochastic model is that due to the inherent nature of randomness, a major outbreak is not always guaranteed when $R_0 > 1$. For example, during the initial outbreak the infected individual or a small number of individuals that have already become infected may recover by chance before they can infect others. This is termed an initial fade-out [236] or a minor epidemic outbreak [234]. An overview of the general stochastic SIR model is given here together with some important properties and approximations that have been developed in the literature, on the assumption of a large population. These are used as a reference for the developed model described in chapter 6.

4.3.3.1 Rate of Contact

For a stochastic SIR model the infectious individuals have contact with other individuals randomly in time at a constant average rate c. Each contact is with an individual selected uniformly at random from the population. The time between contacts is described by an exponential distribution which is a type of probability distribution that describes the time between events in a Poisson process that occur continuously and independently at a constant average rate.

To coincide with the deterministic model the mean contact period μ_c is the reciprocal of the deterministic contact rate c, so that the exponential probability density function (PDF):

$$PDF: \qquad ce^{-ct}, \quad when\, t \geq 0 \qquad (4\text{-}26)$$

$$0, \quad when\, t < 0$$

A PDF of a c value of 0.3 is pictured in Figure 4-5.

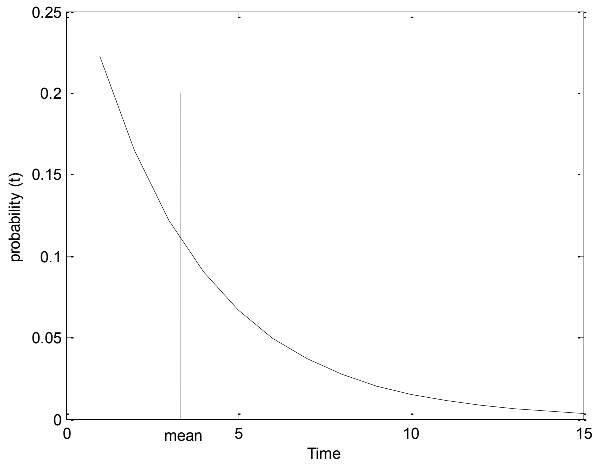


Figure 4-5 – PDF for a c value of 0.3

$Mean\ contact\ period$:
$$\mu_c = \frac{1}{c} \tag{4-27}$$

$Variance\ contact\ period$:
$$\sigma_c^2 = \frac{1}{c^2} \tag{4-28}$$

$Standard\ deviation\ contact\ period$:
$$\sigma_c = \frac{1}{c} \tag{4-29}$$

4.3.3.2 Infectious Period

Infected individuals remain infectious for a time period and then recover. As with the contact rate, the infectious period is described by an exponential distribution. To coincide with the deterministic model the mean infectious period μ_I is the reciprocal of the deterministic recovery rate γ. So that:

$Mean\ infectious\ period$:
$$\mu_I = \frac{1}{\gamma} \tag{4-30}$$

$Variance\ infectious\ period$:
$$\sigma_I^2 = \frac{1}{\gamma^2} \tag{4-31}$$

$Standard\ deviation\ infectious\ period$:
$$\sigma_I = \frac{1}{\gamma} \tag{4-32}$$

4.3.3.3 Discrete Time Markov Model for a Single Compartment

These types of distributions can be modelled using a Markov process [228] which is used to develop stochastic epidemic models and form approximations of some important properties. As with the deterministic model, the stochastic Markov model analyses how the system progresses but will have different sample paths every time it is run.

A discrete time Markov model is used to illustrate how the stochastic epidemic is modelled. Suppose an epidemic model has just one compartment, and hence one Markov process, $X(t)$ representing the number of individuals within that compartment at time t. For a discrete time model the processes are defined on a discrete time scale $\{0, \Delta t, 2\Delta t, \dots\}$, and the states, each representing a possible number of individuals within the compartment, are discrete random variables $\{0,1,2, \dots N\}$. The times between successive jumps of the process are exponentially distributed with parameter α. The rate diagram can be drawn as follows:

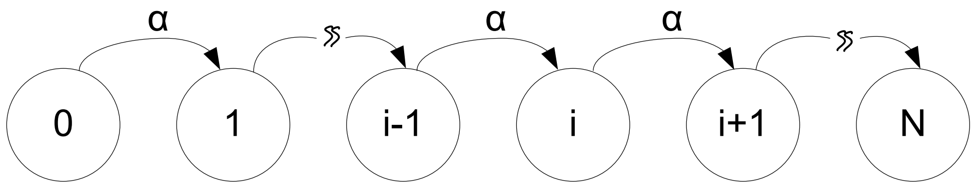


Figure 4-6 – Rate diagram for a Markov process

Where state i represents the state where there are i individuals within the compartment $(X = i)$. The rate diagram can be expressed as a rate matrix R whose elements define the transition rates from one state to another.

$$R = \begin{array}{cc} & \mathit{state\ to} \\ \mathit{state\ from} & \begin{bmatrix} 0 & \alpha & 0\ldots \\ 0 & 0 & \alpha \\ 0 & 0 & 0 \\ \ldots & & \\ & & \end{bmatrix} \end{array}$$

The probability P that within a small time interval Δt the number of individuals X within the compartment has increased by one is given by:

$$P(X(t + \Delta t) - X(t) = 1) = \alpha\Delta t \tag{4-33}$$

And the complement, where the probability P that within a small time interval Δt the number of individuals X within the compartment has remained the same is given by:

$$P(X(t + \Delta t) - X(t) = 0) = 1 - \alpha\Delta t \tag{4-34}$$

The time step Δt is chosen sufficiently small such that the number of infected individuals changes by at most one during the time interval. This means that the rate matrix is largely zero valued otherwise it would need to include the rates to other states where the number of infected could change by more than one. To ensure that the transition probabilities lie in the time interval, the time step Δt must satisfy:

$$\alpha\Delta t \leq 1 \tag{4-35}$$

The probabilities can be expressed as a probability matrix P whose elements now define the transition probabilities from one state to another.

$$P = \begin{array}{cc} & state\ to \\ state\ from & \begin{bmatrix} 1-\alpha\Delta t & \alpha\Delta t & 0 \dots \\ 0 & 1-\alpha\Delta t & \alpha\Delta t \\ 0 & 0 & 1-\alpha\Delta t \\ & \dots & \\ & & \end{bmatrix} \end{array}$$

The corresponding probability diagram is given:

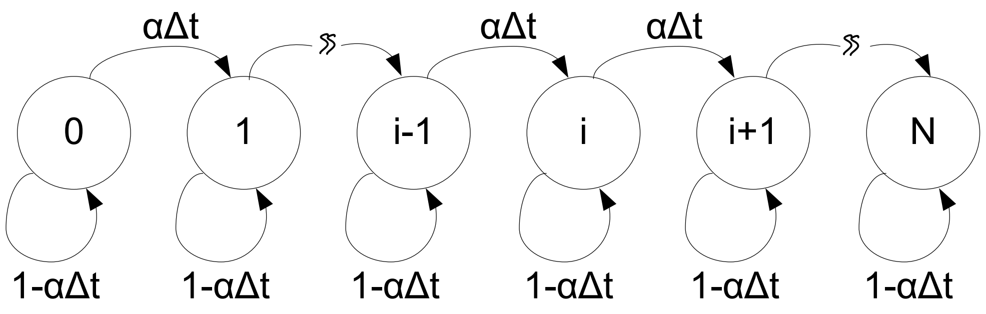


Figure 4-7 – Probability diagram for a discrete Markov process

This process can be coded into software to visualise the different sample paths every time it is run.

4.3.3.4 Discrete Time Markov Model for a Stochastic SIR

With multiple compartments the Markov process becomes a vector, so that each Markov state has a vector component for each necessary compartment. With an SIR model it is considered that only the processes of S and I compartments are needed since $R = N - S - I$. Within a small time interval $[t, t + \Delta t]$, the probability P of an infection is given by the simultaneous transitions $S \rightarrow S - 1$, where one individual leaves the S compartment, and $I \rightarrow I + 1$ where one individual enters the I compartment. Similarly, within a small time interval $[t, t + \Delta t]$, the probability P of recovery is given by the simultaneous transitions $I \rightarrow I - 1$, where one individual leaves the I compartment, and $R \rightarrow R+1 = N-S-I+1$, where one individual enters the R compartment. The probability equations are therefore:

$$P(S(t+\Delta t), I(t+\Delta t) - S(t), I(t) = (-1,1)) = \beta \frac{S(t)I(t)}{N} \Delta t \tag{4-36}$$

$$P\big(S(t+\Delta t), I(t+\Delta t) - S(t), I(t) = (0,-1)\big) = \gamma I(t)\Delta t \tag{4-37}$$

Where $\beta = c\tau$ is the infection rate and γ is the recovery rate.

And the complement:

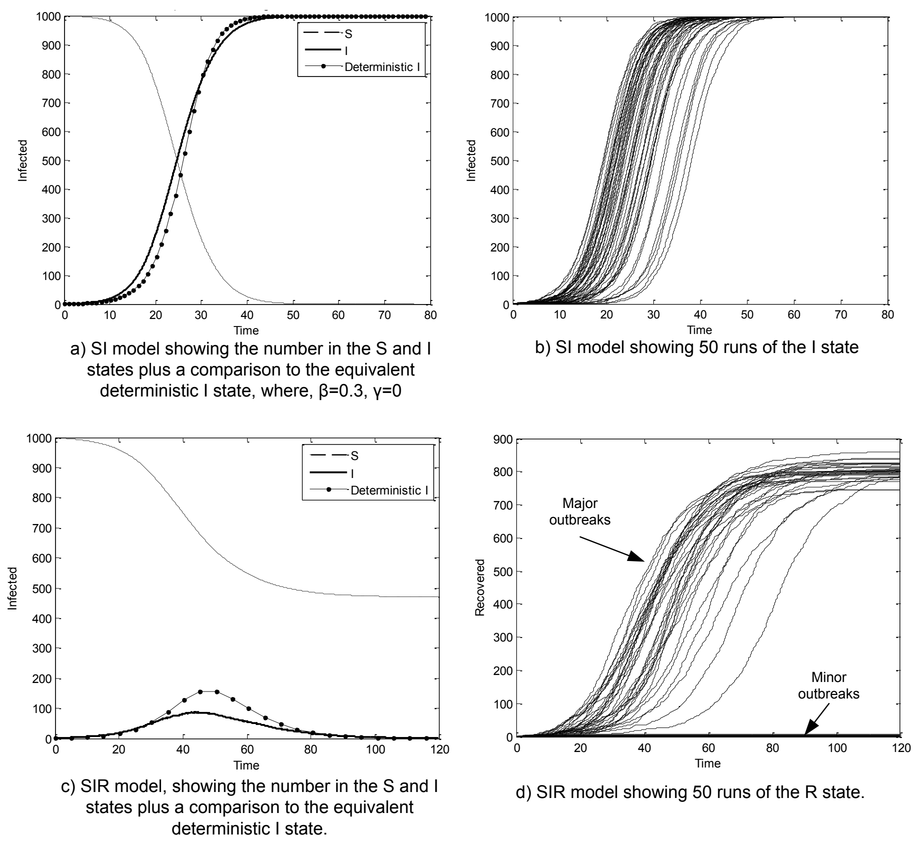
$$P\big(S(t+\Delta t), I(t+\Delta t) - S(t), I(t) = (0,0)\big) = 1 - \left(\beta \frac{S(t)I(t)}{N} + \gamma I(t)\right)\Delta t \tag{4-38}$$

To ensure that the transition probabilities lie in the time interval, the time step Δt must satisfy:

$$\left(\beta \frac{S(t)I(t)}{N} + \gamma I(t)\right)\Delta t \leq 1 \tag{4-39}$$

4.3.3.5 Stochastic Epidemic Example

To illustrate the mathematics an example showing an epidemic following the SIR Markov process is given through simulation in Figure 4-8 using Mathworks Matlab. Parameters were set to show a comparison with the deterministic model (§4.3.2.3). When there is no recovery, $\gamma = 0$, the SIR behaves as an SI model. For an average contact rate of 0.3 ($c = 0.3, \tau = 1, \therefore \beta = 0.3$) the mean result of a large number of runs (1000) is shown in Figure 4-8 (a) together with the deterministic solution. Fifty of the individual runs for those infected are shown in Figure 4-8 (b). When recovery is added, where $\gamma = 0.15$ the mean result of 1000 runs is shown in Figure 4-8 (c). There is a large difference between the stochastic solution and the deterministic result. The reason for this can be seen by inspecting individual runs of the recovered compartment in Figure 4-8 (d), where, for a proportion of the runs there is either very little or no recovery. These runs account for the condition where minor outbreaks have occurred and have resulted in the mean being very different from the deterministic result. Under the SI scenario there is no chance of recovery and so the possibility of a minor outbreak does not occur. Under these circumstances the mean of the stochastic SI model is closer to the deterministic SI model.



a) SI model showing the number in the S and I states plus a comparison to the equivalent deterministic I state, where, β=0.3, γ=0

b) SI model showing 50 runs of the I state

c) SIR model, showing the number in the S and I states plus a comparison to the equivalent deterministic I state.

d) SIR model showing 50 runs of the R state.

Figure 4-8 – Stochastic SIR model.

4.3.3.6 Important Stochastic SIR Properties

The following summarises some important properties of the stochastic SIR model which are used during development work in chapters 6 and 8.

Probability of an outbreak, early stage approximation: When the number in the population is considered to be large, the initial phase of the epidemic can be approximated by a homogeneous branching process [234], which shows that when $R_0 \leq 1$ the final size of the epidemic is bounded in probability and the epidemic will only be minor. However when $R_0 > 1$ the epidemic will have both a minor element, which is bounded with a probability p, and a major element, which is unbounded with a probability $1 - p$. For a standard stochastic model with a closed homogeneous uniformly mixing population and *one initially infected*, $(I(0) = 1)$ the probabilities can be summarised as:

$$There\ will\ be\ a\ minor\ epidemic\ outbreak\ with\ probability\ \frac{1}{R_0} \tag{4-40}$$

$$and\ a\ major\ outbreak\ with\ the\ remaining\ probability\ 1 - \frac{1}{R_0} \tag{4-41}$$

Final size approximation: The final fraction infected $r(\infty)$ are those that end up in the recovered state at the end of the outbreak. This fraction is a solution to the balance equation (§4.3.2.6), which is the same equation as for the deterministic model, except it is assumed that a negligible fraction of the population is initially infected, so that:

$$1 - r(\infty) = e^{-R_0 r(\infty)} \tag{4-42}$$

This equation always has the solution $r(\infty) = 0$, corresponding to a minor outbreak, and when $R_0 > 1$, there is another unique solution of $r(\infty)$ between 0 and 1 corresponding to a major outbreak [234] (Figure 4-4).

Final size distribution of a major outbreak: The final size of a *major outbreak* will be normally distributed around the final size approximation. The notation for a normal distribution is defined here as:

$$Norm\ (\mu, \sigma^2)$$

Where $Norm$ denotes a *normal* distribution, μ denotes the *mean,* and σ^2 denotes the *variance* which is the square of the standard deviation. The *threshold theorem* [234] derives the normal distribution of a major outbreak as:

$$\sqrt{N}\ \left(\overline{r_N(\infty)} - r(\infty)\right) \rightarrow\ Norm\left(0, \frac{r(\infty)\big(1 - r(\infty)\big)(1 + v^2(1 - r(\infty))R_0^2)}{(1 - (1 - r(\infty))R_0)^2}\right) \tag{4-43}$$

Where N is the number of individuals, $r_N(\infty)$ is the mean final size with N individuals (excluding initial infectives) and where $\overline{r_N(\infty)} \rightarrow r(\infty)$ so that the mean μ becomes zero, and the variance is defined by the second term. v^2 is the squared coefficient of variation of the infectious period.

$$v^2 = \frac{\sigma_I^2}{\mu_I^2} \tag{4-44}$$

$$= \frac{\left(1/\gamma\right)^2}{\left(1/\gamma\right)^2} = 1$$

$$(for\ an\ exponentially\ distributed\ infectious\ period)$$

4.3.4 Agent-Based Epidemics

The concept of compartments used in the mathematical models can also be incorporated into other models such as agent-based models. SI and SIR variants have been incorporated into mobile phone models [237]. Within agent-based models nodes are modelled as separate entities so that the result of individual interactions and their behavioural rules determine the spreading dynamics of the epidemic. As pictured in Figure 4-9 each node maintains its own health status which can also be aggregated into a network level perspective depending upon the nature of the simulation model. Agent-based models tend to be difficult to compare since the design of agents, their interactions, and behavioural rules depend upon the specifics of the scenario being modelled. Agent-based models can also have stochastic elements generating output results that are probability distributions.

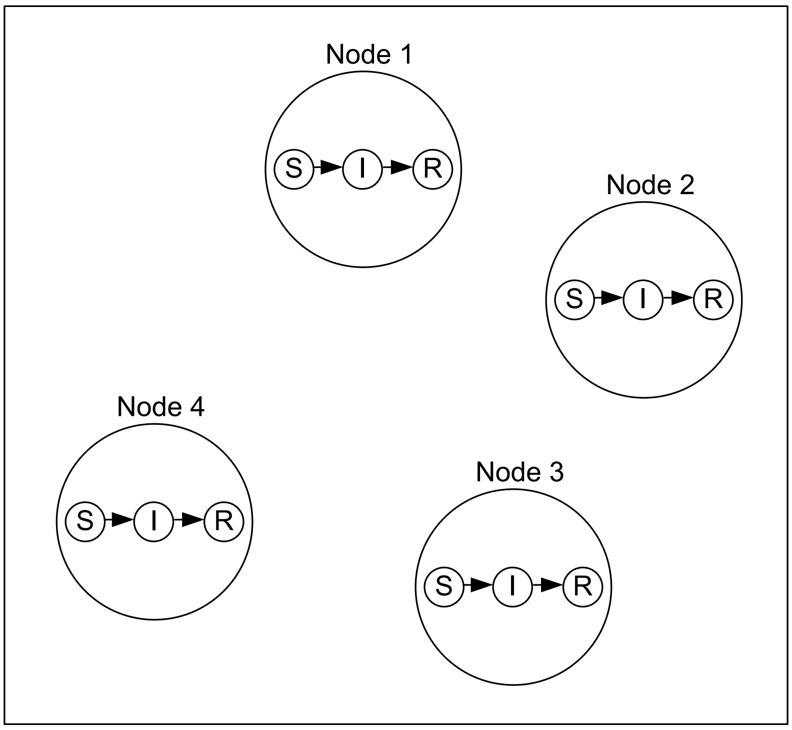


Figure 4-9 – Epidemic agent-based model

4.4 Infection Genetics

It may not just be individual interactions and behaviours that determine the spread of diseases. Another set of models that try to mimic infection are those studied within evolutionary ecology. These types of models attempt to analyse the mechanics of the infection at the individual scale and assume that individuals differ in terms of their genetic make-up. Both the individuals (potential hosts of the disease) and the parasite or pathogen causing the infection are modelled. Matching algorithms are used to match the pathogens to hosts which subsequently then cause them to become infected. Two prominent models are *gene-for-gene* [238] [239] [240] which is based on plant-pathogen interactions and *matching-alleles* [239] [241] [240] [43] [99] based on self/non-self recognition systems in invertebrates. Both models include a genotype for the host and a genotype for the pathogen or parasite. Figure 4-10 shows host and parasite genotypes with two loci each, where A1 and A2 represent two different alleles at locus 1, and, B1 and B2, represent two different alleles at locus 2. Within the gene-for-gene model the parasite alleles within the parasite genotype are labelled as either *a* - avirulent (weakly infectious) or *v* - virulent (highly infectious), and the host alleles within the host genotype are labelled as either *s* - susceptible or *r* - resistant. A host can resist (*R*) a parasite if the host has a resistant allele at any locus for which the parasite has an avirulent allele at the corresponding locus, otherwise infection occurs (*I*). For the matching-alleles model a parasite's genotype must exactly match a host's genotype to successfully infect the host. For example genotype A1, B1 of the host matches genotype A1, B1 of the parasite.

These general principles of genetic matching are incorporated within development work of later chapters to model malware propagation in diverse computing devices and form a novel aspect of the work.

Parasite genotypes are compared to the host genotype to determine whether an Infection (I) will be transmitted or whether it is resisted (R)

Host Genotype		Parasite Genotype			
		locus1 locus2	locus1 locus2	locus1 locus2	locus1 locus2
locus1	locus2	A1 (*a*) B1 (*a*)	A1 (*a*) B2 (*v*)	A2 (*v*) B1 (*a*)	A2 (*v*) B2 (*v*)
A1 (*r*)	B1 (*r*)	*R*	*R*	*R*	*I*
A1 (r)	B2 (*s*)	*R*	*R*	*I*	*I*
A2 (*s*)	B1 (r)	*R*	*I*	*R*	*I*
A2 (s)	B2 (s)	*I*	*I*	*I*	*I*

a) Gene-for-gene, A1 or B1 resistant (2 loci, 2 alleles)

Host Genotype		Parasite Genotype			
		locus1 locus2	locus1 locus2	locus1 locus2	locus1 locus2
locus1	locus2	A1 B1	A1 B2	A2 B1	A2 B2
A1	B1	*I*	*R*	*R*	*R*
A1	B2	*R*	*I*	*R*	*R*
A2	B1	*R*	*R*	*I*	*R*
A2	B2	*R*	*R*	*R*	*I*

b) Matching Alleles (2 loci, 2 alleles)

Figure 4-10 – Two different infection genetic models

4.5 Summary

Ad hoc networks are expected to become more prominent in the future Internet. They can be compared with natural systems since devices such as mobile phones interact closely with humans following their mobility and interaction patterns. Mobility models consider the movement patterns of devices, with one reference model that is widely used being the Random Waypoint. Mobility can be combined with simulators or other agent-based models to analyse individual and network level behaviour. Homogeneous mixing models can be used to model the network as a whole entity, and have particularly been used within epidemic models to analyse network level behaviour of malware propagation. These models use compartments to signify the health status of the network. Properties of the deterministic and stochastic SIR epidemic models have been reviewed. Compartmental methods can also be incorporated into agent-based models where the result of individual interactions and their behavioural rules determine the spreading dynamics of the malware. Another class of models used to analyse disease spread study the mechanics of infection at the genetic level. They assume both individuals and pathogens differ in terms of their genetic make-up. Matching algorithms are used to match pathogens to hosts which subsequently become infected. These modelling concepts are used during development work of later chapters. Specifically homogeneous mixing and the RWP model is integrated together with a compartmental approach of monitoring device infection status in a high abstract level ABM. Additionally matching algorithms are used to match malware to device configurations. Mathematical modelling of malware incorporating both deterministic and stochastic methods are also used during development work.

Chapter 5

Ecosystem Model of an Ad Hoc Network

5.1 Introduction

This chapter links the background material presented in chapters 1 to 4 with the work that follows, and is comprised of two sections:

An Ecosystem Perspective of an Ad Hoc Network Environment: The first section describes an ad hoc network environment as an ecosystem using comparable terminology and relationship analogies to natural ecosystems as described within chapter 2.

A Diverse System Model: The second section firstly outlines the requirements for a diverse system model applicable to ad hoc networks together with constraints that highlight the first steps taken in proving the hypothesis of this thesis. Secondly a threat model of malware utilising multiple exploits across layers of the software stack is defined. Thirdly an overview of two developed models is described. The first is based upon the mathematical epidemic approach, and the second is an agent-based approach.

5.2 An Ecosystem Perspective of an Ad Hoc Network Environment

Organisms within natural ecosystems (§2) and devices within ad hoc networks (§3) both interact in a peer-to-peer fashion, are distributed, and self-organise. It is likely that peer-to-peer wireless networks such as ad hoc networks will become more mainstream than they are currently and therefore forms the basis of the network topology in which to investigate diversity (§3.2.2). If an ad hoc network, together with its users and application environment, is regarded as an ecosystem as shown in Figure 5-1 then comparable terminology can be defined and relationship analogies can be made to natural ecosystems. Note that the definition of an ecosystem here should not be confused with the term 'software ecosystem' which has recently been used to describe the progressive development of a software product or service incorporating development framework tools, organisations, external developers and users such as the Android platform [242].

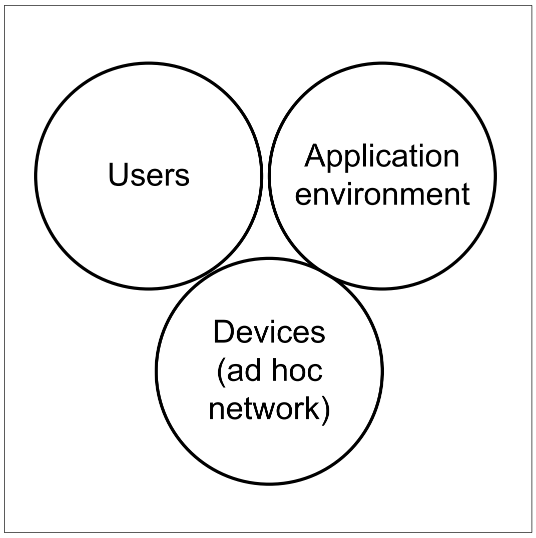


Figure 5-1 - Ad hoc network ecosystem

Figure 5-2 shows the ecosystem diagram equivalent to that pictured in Figure 2-1 of Chapter 2, but for an ad hoc network environment, showing its relationships with software and hardware diversity. Within natural ecosystems, biodiversity is linked to functions and services and its ability to maintain them when faced with unwanted disturbances. It is proposed that by applying biodiversity strategies within an ad hoc network, the destructive effects arising from security attacks can be counterbalanced with the constructive effects of biodiversity to maintain ecosystem function and services, and hence benefit overall resistance and resilience. Although the focus here is on ad hoc networks, many of the principles described are also applicable to computer networks in general. Analogous relationships between software and hardware diversity and ad hoc network ecosystem functions and services are described in the following text.

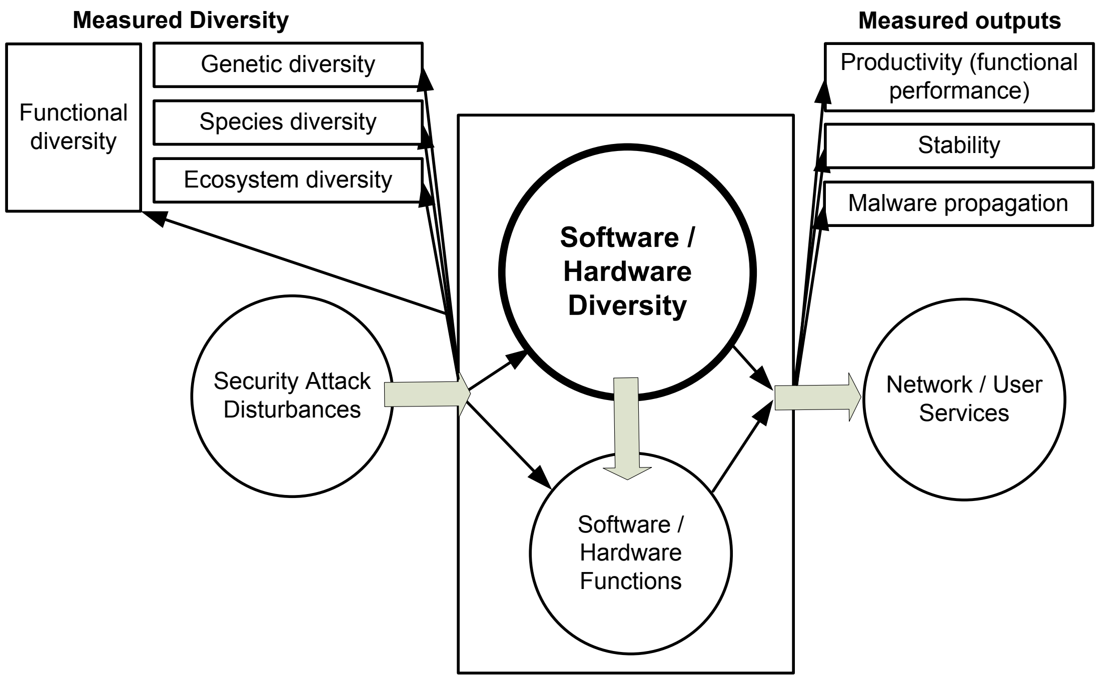


Figure 5-2 - Diversity relationships in an ad hoc network ecosystem

5.2.1 Scales and Diversity Definitions

An ad hoc network environment can be partitioned into three scales on a par with those discussed in natural systems: Individual (I), community (C) and ecosystem (E). The individual scale comprises the independent devices (or nodes) and includes software stacks, protocol stacks, physical hardware, and individual behavioural characteristics and constraints. The community scale includes communities of nodes forming part of a network, or a complete network. This scale is concerned with topology and node distribution, data flow and community behaviours. The ecosystem scale incorporates multiple clusters of nodes or multiple networks and interactions between them, the environment and the users. It also includes beneficial outputs such as the resulting services.

Many principles of diversity are applicable at multiple scales (§2.2.2), with some diversity metrics relevant to both species and genotypes (§2.3). Ecologists tend to describe diversity in relation to species since it is the easiest to measure and experiment with however there is growing evidence that the same relationships have been observed at the genotypic level. Partitioning individuals by genotype composition or common characteristics into species is well defined in natural systems, however categorising elements of an ad hoc network, in theory can be conducted on a sliding scale depending upon the chosen granularity. For example a software program could form a genetic element of an ad hoc device, or it could be categorised into a species of software. The most important aspect is the relevant diversity relationships that exist regardless of the scale. For an ad hoc network ecosystem the definitions of diversity are partitioned as follows, however the relationships in 5.2.6 are described where they are analogous to that observed in nature regardless of the partition.

(1) Genetic Diversity: This describes the variance in structural composition of devices in terms of software and hardware components and is applicable at the individual scale. There are methods for defining and measuring diversity in terms of genetics that is applicable to defining and measuring the diversity of software composition across devices (§2.3) which is discussed further in this chapter (§5.2.6, §5.3.2) and used in subsequent chapters (§6.3, §7.4).

(2) Species Diversity: Species diversity could potentially have multiple meanings depending upon the chosen granularity and focus as described above. When categorising ad hoc devices as complete entities, species could mean a type of device, such as a local ad hoc router or a gateway to the internet, and would be applicable at the community scale. Alternatively, when describing software as individuals from a pool of available programs, species could mean a type of software program, such as a web browser, or an instant messaging application. This is because there are analogies between natural species diversity mechanisms and the way in which software is developed and adopted by users. Relationship analogies are described further in section 5.2.6.

(3) Ecosystem Diversity: This is the diversity between distinctly separate groups or networks of devices, or the diversity incorporating both networks and users and the environment. It is applicable at the ecosystem scale and can be measured in terms of genetics or species.

(4) Functional Diversity: This is the variance and breadth of functionality and services, as a result of software, hardware, devices or networks interacting with their users and the environment. Consequently functional diversity spans individual, community and ecosystem scales.

5.2.2 Software and Hardware Functions

As with natural ecosystems, an ad hoc network ecosystem is comprised of many interacting components; not just the devices themselves, but interacting layers of software and hardware generating a range of processes and functionality at different scales. For example functions can arise from single or multiple interacting software and hardware modules, or be generated by single or multiple devices. Functions can be partitioned into *regulating*, *supporting*, and *provisioning* categories as they share similar analogies to those described within natural systems (§2.2.1.2). Table 5-1 gives some examples for each type. Network access, for example, is a regulating function since it controls how and when the network can be accessed for data transmission. Data and program storage is a supporting function since it allows all of the software programs to be stored in memory along with any data that is generated or shared: without it the devices would not be able to operate as intended. The sending of text messages is a provisioning function since it can be viewed as a product that is produced from within the ad hoc network.

5.2.3 Network and User Services

Network and User Services are the beneficial services provided by the interaction of all the components in the ad hoc network ecosystem, and, like natural systems, are derived from multiple underlying functions. Services can also be partitioned into regulating, supporting, and provisioning categories (§2.2.1.3) as shown in Table 5-1. Examples of regulating services include the quality of service of data or communication traffic, or, in the broadest sense, the control of malware spreading. Supporting services include distributed data storage and data gathering which is driven by the natural topology of an ad hoc network. Provisioning services include the beneficial product outputs such as an electronic health care service, or a multimedia data streaming service.

Table 5-1 – Ad hoc ecosystem functions and services

	Functions	Services
Regulating	Network access Routing data Monitoring data traffic	Quality of service (data and communication) Buffering against faults or device failures Controlling malware spreading
Supporting	Data and program storage Computational power Energy and battery management Display screen and sensor interfaces	Distributed data storage Sensing and distributed data gathering
Provisioning	Sending and receiving text messages Sending and receiving media files Generate images Produce documents	File sharing Multimedia data streaming Communication services Electronic Health care service Environmental Monitoring service

5.2.4 Security Attack Disturbances

5.2.4.1 Disturbance Regime vs Disturbance Events

Within the ad hoc network ecosystem it is necessary to distinguish between the natural disturbance regime and a single disturbance event (§2.2.1.4). The disturbance regime, on a par with natural ecosystems, shapes an ad hoc network environment over long time scales such as changes in technological advances, trends in user habits, business markets, and application areas which will contribute to evolving functions and services. A single disturbance is an event of intense stress occurring over a relatively short period of time potentially causing large changes to the dynamics of the ad hoc network. Security attacks such as malware can be thought of as single disturbance events creating destructive effects at varying speeds and severity depending upon the specific attack.

5.2.4.2 Malware Disturbance Events

As with natural ecosystems where diseases can spread quickly in monoculture populations (§1), so too can malware under similar conditions where there is wide spread use of identical software (§3). Examples of the effects of different malware on functions and services [243] [244] [245] [246] at different scales is shown in Table 5-2.

Table 5-2 - Malware effects on functions and services

	Effects on Function and Services											
	Individual Functional Effects								Community Functional Effects		Ecosystem Services Effects	
Malware Disturbance	Operation of device slows down	Application or protocol software stops working, or behaves incorrectly	Damage to stored data	Complete device shutdown / reboot	Increase in received messages / data	Increase in transmitted messages / data	Unable to transmit messages / data	Unauthorised collection of data	Reduced availability for network communication	Increased or bursts in network traffic	Service is not accessible	Service is slow (reduction in Quality of Service)
Virus	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Worm	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Trojan	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Spyware								✓				
Ransomware		✓	✓	✓			✓	✓				
Adware								✓				

Depending upon the type of malware and the motivation of the attack, effects at the individual scale can range from slowing down the operation of a device, to completely shutting down the device. At the community scale, malware such as some worms and viruses replicate and forward themselves as fast as possible creating bursts in network traffic or a reduction in network availability for communication. This may result in either a general reduction in quality of service at the ecosystem scale such as speed of retrieving data, or no service at all. Other types of malware such as Trojans, spyware, and adware are often installed by mistake, hidden within genuine programs and slowly extract personal data without affecting the functioning of the device or network. Ransomware can restrict access to data, software or general functionality of the device.

5.2.5 Measured Outputs

Productivity and stability are two important output measurements of natural ecosystems because they consider the effects on function, and the impact on resistance and resilience, of disturbance. When the spread of disease is considered, additional transmission characteristics are also analysed.

Within ad hoc networks, and networks in general, the term productivity is not discussed directly, instead the overall functional performance of a service, termed *Quality of Service* (QoS), is often used [129] [247] [248]. QoS can be considered through a number of functional outputs associated with performance such as throughput (amount of data successfully transferred within a fixed time period), bit error rate (number of transmission bit errors per unit time), and network delay (time taken for a bit of data to be transferred). It often depends upon the context as to which is used. Estimation of such characteristics can often be achieved through network simulators (§4.2.2). Additionally, it would be possible to analyse these functional outputs in response to malware so that the resistance and resilience of the network's quality of service could be inferred and is discussed further in section 6.4.1.

5.2.6 Natural Biodiversity Mechanisms

5.2.6.1 Software at the Individual Scale

In order to form analogies between natural diversity mechanisms and underlying ad hoc ecosystem functions a device is framed in terms of genetic software components. This is because genetic diversity can inform the diversity between individuals, species, or ecosystems (§2.3.1). Additionally, in order to devise biodiversity strategies based on local interactions and multiple layers of software, whilst incorporating multi-exploit malware, it is necessary to focus on the individual scale of a device and its structural composition. Figure 5-3 shows an ad hoc ecosystem with devices (Figure 5-3 (a)) comprised of individualised software stacks (Figure 5-3 (b)) generated from a pool of available software (Figure 5-3 (c) and (d)). This pool of software can be stored locally in whole or in part but is assumed to be separate from the realised software stack.

The *structural composition* of each device's individualised software stack (Figure 5-3 (a)) can be considered from a genetic perspective by representing this structural composition as a *genotype* (§2.2.2.1.4). The *pool of available software* (Figure 5-3 (c)) contains a bounded number of functions and variants with which to configure the genotype. The genotype is split into four layers representing the four general layers of the software stack (§3.2.1.3): applications (Layer A), application services (Layer B), core OS libraries (Layer C), and kernel (Layer D). Each layer is comprised of one or more software functions representing genes, termed *software gene functions* (F). Each function is situated at a specific locus (L) within the software stack.

Software gene functions for example may include web browsing, window management, graphics rendering, or hard disk interfacing. Each software gene function can be represented by one of a number of possible *software gene variants* (alleles in a biological systems), such as web browser type 1, or web browser type 2. Here a monoploid set of chromosomes is assumed (§2.2.2.1.4) so that only one variant is allowed at a locus within a single genotype at any given time.

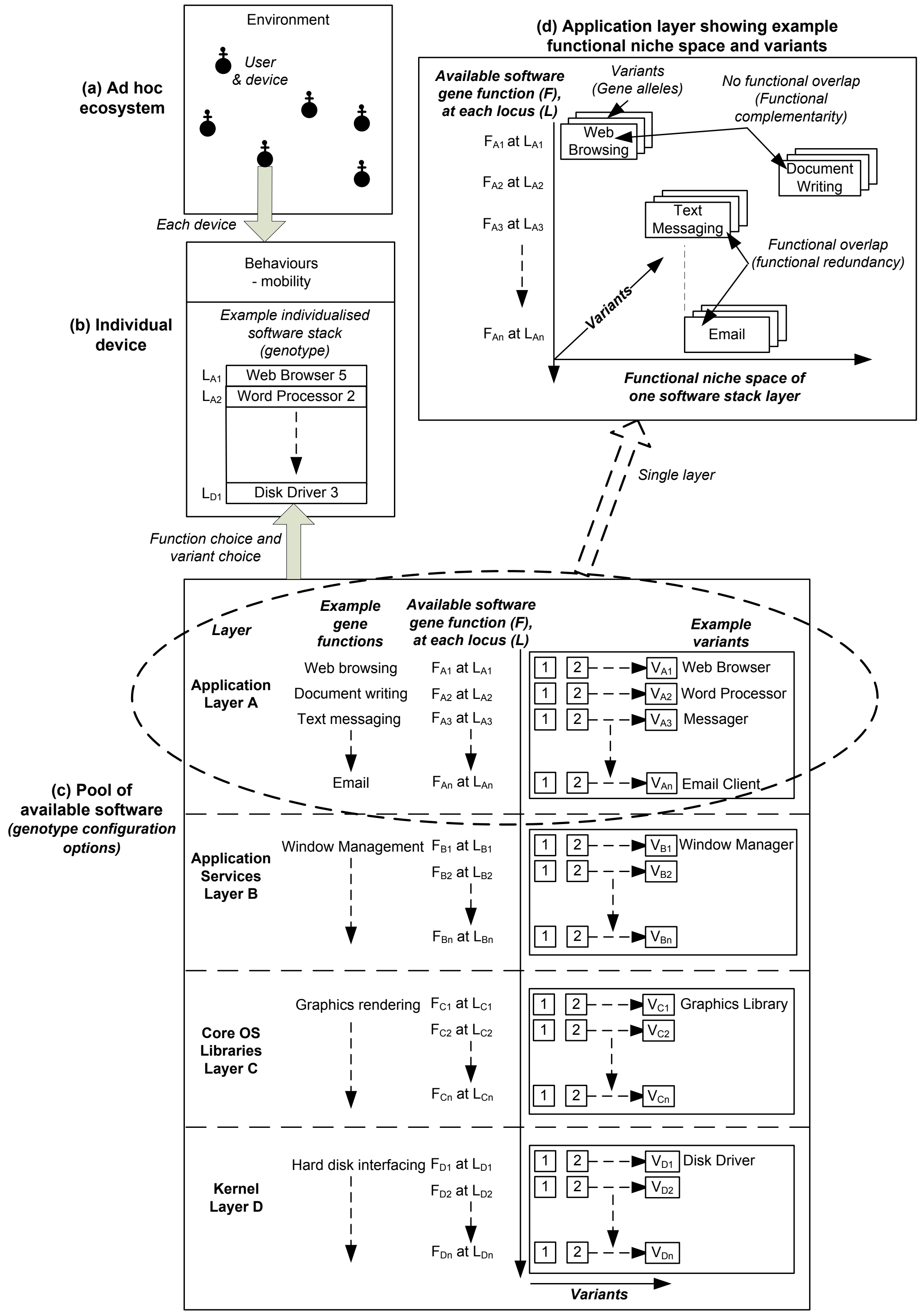


Figure 5-3 – Software at the individual scale

Software gene variants are defined from the perspective of propagating malware and are assumed to be sufficiently different, whilst remaining functionally equivalent, to warrant the necessity of different exploit code to penetrate the vulnerability.

In addition to this, each layer has a *bounded functional niche space* (§2.2.2.1.1) as shown in Figure 5-3 (d) by a third axis, where the variant axis has been rotated. Each software gene function has a position within the functional niche space representative of the functionality of that gene. There may be both overlapping and non-overlapping functions between loci. For example non-overlapping functions could be web browsing and document writing, whereas overlapping functions could be text messaging and email, both of which enable the sending and receiving of plain text communication. In principle, software genotypes can be of varying lengths encompassing different functions, allowing functional diversity to exist within the ad hoc network ecosystem.

Software is defined here in terms of genetics with two components of gene function and gene variant. However, as mentioned previously biodiversity mechanisms researched within ecology, predominantly associated with species, are also relevant at multiple scales, and are particularly relevant from the perspective of software functionality, and are described below.

5.2.6.2 Niche Differentiation

A niche for a particular software program is defined here by its functionality. As with natural ecosystems, when software overlaps in terms of functionality, sometimes competition or temporal conditions can reduce the software's fundamental niche to a realised niche (§2.2.2.1.1). For example text messaging and email overlap in sending text communication. When users have access to both, competition of usage and adoption by users results in each of the two mechanisms being better suited under different conditions leading to two different niches. It is quite possible that text messaging is used for sending short amounts of text because it is quick and instant, whereas email is used for sending larger amounts of text often in a more formal manner.

Ecological research suggests that both *perfect complementarity* (no functional overlap) and *functional redundancy* (functional overlap) greatly benefit ecosystems (§2.2.2.1.1). If software systems were designed with this in mind then perfect complementarity would generate greater functionality more rapidly as more software programs are developed. The downside would be a total dependence on a specific program to provide a certain function. Malware targeting a specific program type such as email could therefore cause loss of critical functionality, and hence redundancy is also needed. Within current software systems, where software for the user or application is the focus, both perfect complementarity and functional redundancy exist, but it is not evident from the literature if this has ever been analysed. Additionally, different software variants providing the same functionality exist, such as different web browser software or different email software applications. Functionality of software may not always remain static and could dynamically change during operation. Self-modifying code such as software reflection, where software is able to examine its own operation and modify its functionality at runtime, could potentially cause changes in the realised functional niche space at a particular locus (figure 5-3 (d)) over time (albeit that this represents only a part of the larger fundamental niche space to which it has access), which could also differ on different computers. One question is whether the changes could significantly impact the overall functional goal of the loci. Small changes may be beneficial for diversity as it could lead to slightly different approaches, different ordering of lower level commands and different memory locations of data, whilst still achieving the same goal. Significant changes however could mean that two variants at the same locus could no longer be considered as having the same functionality and would violate the concept of functionally equivalent software variants.

5.2.6.3 Facilitation

Software programs seldom operate in isolation of each other and *facilitation* (§2.2.2.1.2) is a natural process in software systems. Two or more pieces of software interacting together can cause a positive benefit for at least one of the software programs. An example would be of two software programs: a scanner driver software interfacing directly to the scanner hardware, and a software program to view and save the scanned image. Without the scanner driver, there would be no scanned image to view. Within ecology, increasing diversity in the presence of facilitation is thought to increase ecosystem function but the exact mechanisms and effects, particularly at the genetic level, are largely unknown (§2.2.2.1.2). However this type of arrangement is normal within software stacks where there are many dependencies between software functions at different layers.

5.2.6.4 *Trophic Levels*

The dependence between software functions at different layers of the software stack can also be viewed as being similar to the interaction between *trophic levels* of natural systems (§2.2.2.1.3). Similar to the lowest trophic level, the lowest software layer contains primary functions that interface to the outside world such as drivers for hardware and other low level functions (kernel). The next layer (core OS libraries) is built upon the kernel. The third layer (application services) is built upon the core OS libraries or sometimes the kernel as well. The top layer (applications) utilises the lower layers to provide functional software for the user. It is known in ecology that diversity at lower layers can increase the number of species at higher layers. In terms of software this would indicate that the more diversity in software functionality in the kernel, the more diversity there is, or can be at the application layer. This makes sense since devices with only a disk driver functionality in the kernel would have very limited application software. On the other hand if the kernel had drivers for a range of different sensors and actuators then a multitude of different application software would exist, and this is seen in practice with large volumes of 'Apps' available in 'App Stores' [249] [250]. As well as dependencies across layers there may be dependencies within layers at a finer level of granularity, for example between software programs and dynamic libraries. For the scheme proposed in Figure 5-3 this would mean dependencies between loci. The implication being that in order for particular loci to be operational, specific lower level loci would need to be present, limiting genotype configurations. However, a software program and the dynamic libraries it uses remain decoupled until the program actually runs. This is beneficial for diversity because patching and updating of the library can be conducted without recompiling or re-linking the software program, but more importantly different variants of the same library can be used on different computers.

5.2.6.5 Genetic Variation

Within ecology, genetic variation is the driving force behind functional differences between individuals (§2.2.2.1.4). Using the assumption that the software stack of a device can be represented as a genotype with multiple loci representative of multiple layers of software as shown in Figure 5-3, then the genetic variation of a group of ad hoc devices can be defined. The genetic variation is the number and frequency of different software variants across each locus and the number and frequency of different software stack genotypes. For one software gene function at a single locus, as the number of variants increases so too does the possible number of different genotypes. For example if there are five possible web browser variants, there are five possible genotypes. Although web browsers are inherently prone to being an initiating source of an attack (e.g. users unknowingly clicking onto malicious links), variants are considered to be sufficiently different with respect to propagating malware to warrant the necessity of different exploit code to penetrate the vulnerability (§5.2.6.1). As more gene functions are added, and hence more loci, the possible number of different genotypes increases according to Equation (2-12). Genetic variation of software variants determines the genetic diversity of the ad hoc network for which there are numerous measures used within ecology (§2.3.2). These methods could equally apply to the diversity of software composition across devices.

5.3 A Diverse System Model

5.3.1 The Requirements for a Diverse System Model

5.3.1.1 Requirements Overview

There is a large gap in understanding the benefits of diversity as a security mechanism from an ecological perspective (§3.4.5). Additionally there is no well defined metric for measuring diversity of computing systems. Ad hoc networks will feature more prominently in the future Internet (§3.2.3) and possess similar characteristics to natural ecosystems such as localised interactions, distributed architecture and the production of analogous functions and services to those of ecosystems. The spread of malware, similar to the spread of diseases, is rife in monoculture environments (§1.1.2), where it takes advantage of vulnerabilities at different layers of the software stack.

To investigate the benefits diversity brings against disturbances, such as malware spreading events within an ecosystem context, a *model of a diverse system* (§3.4.3) is required. It will need to simulate the injection of malware events whilst incorporating multiple layer exploits, diversity schemes based upon local interactions, mobility, and the peer-to-peer nature of ad hoc networks. It should also consider practical constraints such as user preferences (§3.4.5) and software compatibility, where there may be dependencies between specific variants at different loci (§3.2.1.3). Dependency between the presence of one locus and another, such as dependency on specific lower layer libraries, is not included (§5.2.6.4) however the model could be extended to allow different loci, and different numbers of loci on different computers. It will need to assess important outputs of an ecosystem such as the quality of service as a measure of productivity, resistance and resilience components of stability, as well as malware transmission characteristics. In addition to this, metrics for measuring diversity is required that captures the granularity of different functions and layers of a software stack and their distribution across devices in the network.

Most of the diverse system models developed in the literature (§3.4.3) treat each node as a complete entity. For example, treating nodes as different colours, or different single variants of software. As a result, and in general, malware modelling tends to simulate the targeting of single software variants as a complete entity on a device. The primary contribution of the model is to incorporate diversity whilst accounting for malware that uses multiple exploits targeting different vulnerabilities at different layers of a software stack, which is a growing concern within cyber security (§3.3). The model also allows the evaluation of different diversity strategies and is able to compare single and multiple exploit malware whilst using the same diversity strategy.

5.3.1.2 Model Constraints

The following assumptions and constraints have been applied as a starting point in modelling such a system, but the model could be extended at a later date to include further aspects.

- Within the natural biodiversity-ecosystem relationship, biodiversity can both affect the response to disturbance events, and can be affected by disturbance events (§2.2.2). This research only focuses on *the effect of biodiversity on the response to disturbance events* as a first step in modelling such complex relationships.

- Additionally this research focuses strictly on *disturbance events* over short time scales and ignores the effects of natural changes over long time scales caused by a disturbance regime (§2.2.1.4). This includes considering only closed networks where the number of nodes remains fixed so that there are no nodes entering or leaving the network.

- The disturbance event studied is constrained to *malware* since this is known to be rife in conditions where there is wide spread use of identical software and is on a par with disease spread in natural systems.

- The structural composition of each device's individualised software stack is considered from a genetic perspective by representing the *structural composition as a genotype*. Specific constraints are outlined in section 5.3.1.3 below. Malware and its exploits are also considered from a genetic perspective with the ability to target one or more software variants.

- Modelling of behaviours and node interactions remains at a high level of abstraction so that event based simulators, where data flow and detailed functionality of software is modelled, is not necessary.

5.3.1.3 Genotype Structure Constraints

To simplify modelling and to demonstrate the concepts of both diversity and malware targeting multiple layers, one function is included from each of the four layers of the defined software stack (§3.2.1.3) leading to genotypes with a fixed length of four loci. It is not necessary to model untargeted loci since they have no impact on malware propagation at an abstract level. This means that malware carrying exploits is limited to targeting at most four loci, one per layer, with any number of software variants being bounded only by the simulation parameters. This is a reasonable constraint to make since even the well known multi-exploit malware, Stuxnet (the first to use so many unknown cross-layer exploits §3.3.3), only targeted as many as four layers of the stack. It is additionally implied that at the time of initial infection the exploits are unknown and cannot be detected or blocked by anti-virus software as in the case of a zero day attack (§3.3.3). It demonstrates both the concept and the applicability of the current practical scenario of four exploits in the AND configuration. The OR configuration is applied to both cross layer and within layer and so the total number of exploits modelled can be far greater. Secondly, it is assumed that every device in the network has the same set of functionality (i.e four loci)" and this functionality does not change (i.e through self-modifying code) so that the niche space remains fixed throughout the simulations and the variants at each locus are considered to be functionally equivalent. Thirdly, it is assumed that there is no functional overlap between the four chosen loci. Fourthly, although facilitation and trophic levels are inherent in interacting software programs and layered software stacks, their interacting mechanisms are not included in the model.

These constraints allow the model to initially focus upon the effects of gene variation with respect to software variants whilst fixing the number of functions. This is because individual malware code predominantly targets specific variants with individualised exploits. With non-overlapping functions and a fixed number of loci across all genotypes, functional diversity is limited to the definition of the number of loci representing the different gene functions. However, the model could be extended at a later date to include the effects of functional variation such as varying the number of loci and functional overlap, making it additionally possible to categorise malware into types that target certain functions. Additionally the model could be extended to include relationships between layers impacting vulnerability using multi-stage Boolean logic. This will encompass dependencies between layers that contribute to software becoming vulnerable or not (see future work §9.3.1.3).

Figure 5-4 shows an example genotype with four loci. The first locus represents software relating to an application layer functionality such as web browsing. The second locus represents an application services layer functionality such as window management, which may be used by the web browser application to manage the style and position. The third locus represents a core OS library such as graphics rendering which may be used by the window manager to process 2D and 3D graphics. The fourth locus represents a kernel layer functionality such as low-level hard disk interfacing which may be used to manage downloaded files. For every genotype on every device, each locus can have one of a number of different variants, so for example locus 1 could have one of a limited number of different web browsers.

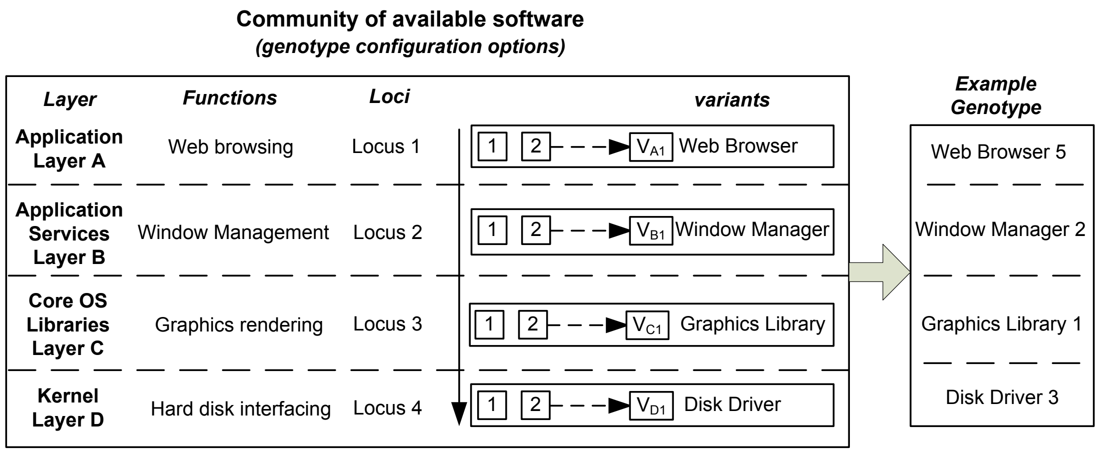


Figure 5-4 – Constrained genotype with four loci

The example shows a possible scenario, but equally represents any defined stack or partial stack at an abstract representation, where it is assumed that the granularity chosen and the functionality defined is in relation to the attacking malware. The variants at each locus can be automatically generated variants using diversity techniques or comprised of already available software (COTS), but are assumed to be sufficiently different from the perspective of the malware to warrant the necessity of different exploit code to penetrate the vulnerability (§5.2.6.1). For example, if the vulnerability lies within the source code implementation or design of an automated set of diverse binary files then the vulnerability may exist in all files, but the exact exploit code would need to be different for each variant. This is one of the fundamental benefits of diversity – to prevent vulnerabilities that exist from being exploited on a wide scale. If variants consist of different COTS software (e.g. Linux OS, Windows OS) the vulnerabilities are more likely to be different in the variants. However if the COTS variants were different versions of the same software (e.g. 10.12.5, 10.12.6) they could still be considered as different if different exploits are required, even if targeting the same vulnerability. The model takes into account the ability to infect different versions of the same software by specifying exploits within a locus (as opposed to across loci). The malware threat model and types are defined in §5.3.3.

5.3.2 Diversity Measures

There are many diversity measures in the literature for natural systems (§2.3), several of which have been used in isolation to define diversity in computing systems, or new multidimensional ones have been defined (§3.4.4). It is proposed here that diversity of computing systems is not defined by a single measure, but through several, all providing a different but necessary perspective (§2.3.2.4).

The diversity measures defined here along with the defined genetic composition of software described above captures the principles of all those proposed in the literature. For example multidimensional functions (§3.4.4.4) is captured in terms of software gene functions and functional overlap, where the partitioning of gene functions into different loci form the dimensions and are only limited by the granularity of the defined functions. The necessary measures are all currently used to assess genetic diversity within natural systems and are defined here (in terms of computing devices and software stack genotypes) for clarity which are used by the models (§6, §7). Additionally defined is the process for increasing diversity in relation to the metrics and the definition of maximum diversity.

5.3.2.1 Measures Definition

Genotypic Richness (G_R) – This is the number of different software stack genotypes currently in use across all devices within the ad hoc ecosystem (§2.3.2.1.1).

Genotypic Diversity (G_d) - This takes into account the frequency of all the different software stack genotypes across all devices and is calculated using Equation (2-2) (§2.3.2.1.3).

Number of Variants (v) - This is the number of software gene variants at a particular locus across all devices.

Variant Richness (V_R) - This is the average number of different software gene variants per locus across all devices. This is the same as the *allelic richness* (§2.3.2.2.1) and is calculated using Equation (2-4).

Variant Diversity (P_d) – This takes into account the frequency of software gene variants across all devices and can be calculated independently at each locus or averaged across loci. The *Nei Genetic Diversity* index is used as a measure of variant diversity as given by Equation (2-8) (§2.3.2.2.2).

5.3.2.2 Increasing Diversity

Diversity at a *single locus* can be increased in two ways by either:

1) Increasing the number of software gene variants, or
2) Equalising the distribution of variants across all devices

Diversity of *multiple loci* can be increased in three ways by either:

1) Increasing the number of loci (software gene functions / software stack layers)
2) Increasing the number of software gene variants at any locus, or
3) Equalising the distribution of either, or both variants and genotypes.

5.3.2.3 Maximum Diversity Definition

Maximum Number of Unique Genotypes: The number of variants at each locus dictates the maximum possible number of genotypes that could exist (Equation (2-12)).

Maximum Genotypic Diversity: This is the maximum diversity that can be achieved for a given set of genotypes, where they are evenly distributed across all devices. The given set of genotypes does not necessarily have to be the maximum number of unique genotypes (Equation (2-2)).

Maximum Variant Diversity: This is the maximum diversity of a given set of variants at a locus where all the available variants are evenly distributed. The Nei Genetic Diversity index (§2.3.2.2.2), (as well as the Shannon index), assess each locus independently and so maximum variant diversity at every locus may not necessarily need to utilise all of the possible unique genotypes. Figure 5-5 shows two examples where maximum variant diversity is achieved with three variants in each of the two loci (A1 to A3 and B1 to B3). In Figure 5-5 (a) even though only three genotypes are present in a network with nine devices, the three variants in each locus are evenly distributed, where each genotype appears three times. Figure 5-5 (b) shows an alternative solution where all the maximum number of nine unique genotypes are fully utilised and the three variants in each locus are also evenly distributed.

Device Genotype	
Locus 1	Locus 2
A1	B1
A2	B2
A3	B3
A1	B1
A2	B2
A3	B3
A1	B1
A2	B2
A3	B3

a) 3 Genotypes used to achieve maximum variant diversity at each locus

Device Genotype	
Locus 1	Locus 2
A1	B1
A1	B2
A1	B3
A2	B1
A2	B2
A2	B3
A3	B1
A3	B2
A3	B3

b) 9 Genotypes used to achieve maximum variant diversity at each locus

Figure 5-5 - Maximum variant diversity with nine devices and three variants

Absolute Maximum Diversity: To guarantee absolute maximum diversity within an ad hoc ecosystem there are four conditions that need to be fulfilled.

1) For a given number of variants and loci there exists a *maximum number of unique genotypes,* all of which need to be utilised across the devices of the ad hoc ecosystem.

2) The *maximum number of unique genotypes* needs to be evenly distributed so that *maximum Genotypic Diversity* occurs.

3) It follows that if the *maximum number of unique genotypes* are evenly distributed so too are the variants within each locus so that *maximum Variant Diversity* is also achieved.

4) To allow *absolute maximum diversity* to be achieved practically, the *minimum number of ad hoc devices* needs to be equal to the *maximum number of unique genotypes,* or a multiple of, to achieve an even distribution.

5.3.3 Malware Threat Model

The malware threat model is defined at an abstract level and depicts a theoretical representation regarding the way in which malware uses exploits to target different layers of the software stack to infect and propagate. The threat model is based upon the background research of malware (§3.3) applied to the previously defined software stack genotype (§5.2.6).

Within this threat model, malware is defined by three parameters:

1) The number of exploits targeting different software variants at each locus (e_1 to e_L).

2) The number of different loci targeted by the exploits (L).

3) The logic function defining the relationships of the exploits in order to carry out its malicious intention (AND, OR).

The malware representation is shown in Figure 5-6 showing exploits, loci and the logic function block. A single exploit is assumed to only be able to penetrate a single software variant. In practical terms this means that if an exploit is capable of penetrating two non-identical but similar software program stacks with common components, then they would be considered as being the same variant at the loci of the common components where the exploit is targeting. If different exploit code is needed then they would be considered as being different variants. It is possible for malware to carry multiple exploits at each loci to enable it to penetrate different variants of the same function. The number of loci defines the number of different software stack layer functions targeted by the malware in order to successfully carry out its malicious intention. The logic function defines the relationship of the exploits across the loci and is based upon *two types: the logical AND type* and *the logical OR type*. The AND and OR logical functions (together with inversion) form the basic blocks for which all other logical functions can be created and has therefore initially been limited to these two types.

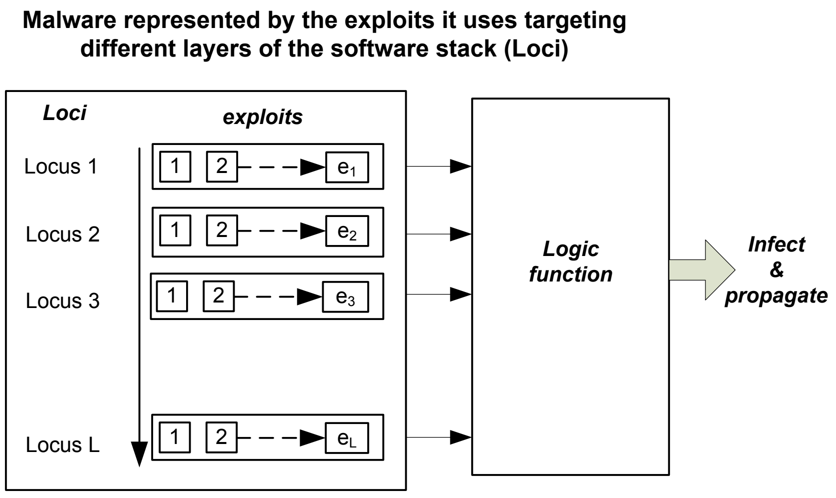

Malware represented by the exploits it uses targeting different layers of the software stack (Loci)
Loci
exploits
Locus 1
Locus 2
Locus 3
Locus L
1 2 e1
1 2 e2
1 2 e3
1 2 eL
Logic function
Infect & propagate

Figure 5-6 – Malware threat model

The *logical AND type*, is representative of malware that uses one or more exploits across loci to infect and propagate, and thus creating an AND relationship across these loci. In this case at least one of the variants in each of the loci targeted by the exploits must be present on a device to cause an infection. This means that the AND malware type only targets loci for which it has an exploit for. All other loci are not affected.

As an example the AND malware type is mapped to the Pegasus malware (§3.3.3) that targeted iPhone devices in 2016 and is shown in Figure 5-7. Here three exploits are used in an AND relationship across loci, where all three software vulnerabilities have to be present in order for the malware to be successful. Other multi-exploit AND malware exist, for example the self-propagating Stuxnet worm (§3.3.3) requires four separate exploits to infect and propagate. Although these malware do not have the capability to propagate over peer to peer wireless connections (e.g like the Cabir worm over Bluetooth [156], it does demonstrate the principles of using multiple exploits in an AND relationship. As the rise in mobile malware continues, multiple exploits are likely to become equally applicable to ad hoc networks with propagation over peer to peer wireless connections. Additionally, these malware examples only targeted one variant at each of the loci but it could have been theoretically possible to have used alternative exploits targeting different variants at the same locus if it was deemed worthwhile by the attackers, and suitable vulnerabilities were found. In 2017 for example at least eight vulnerabilities were identified across different OS implementations (potential variants) of the Bluetooth software stack [251] (at the Kernel layer) potentially leading to the automated spread of malware over peer to peer Bluetooth connections without being detected. It is just a matter of time before these are exploited by malware in unpatched versions of operating systems.

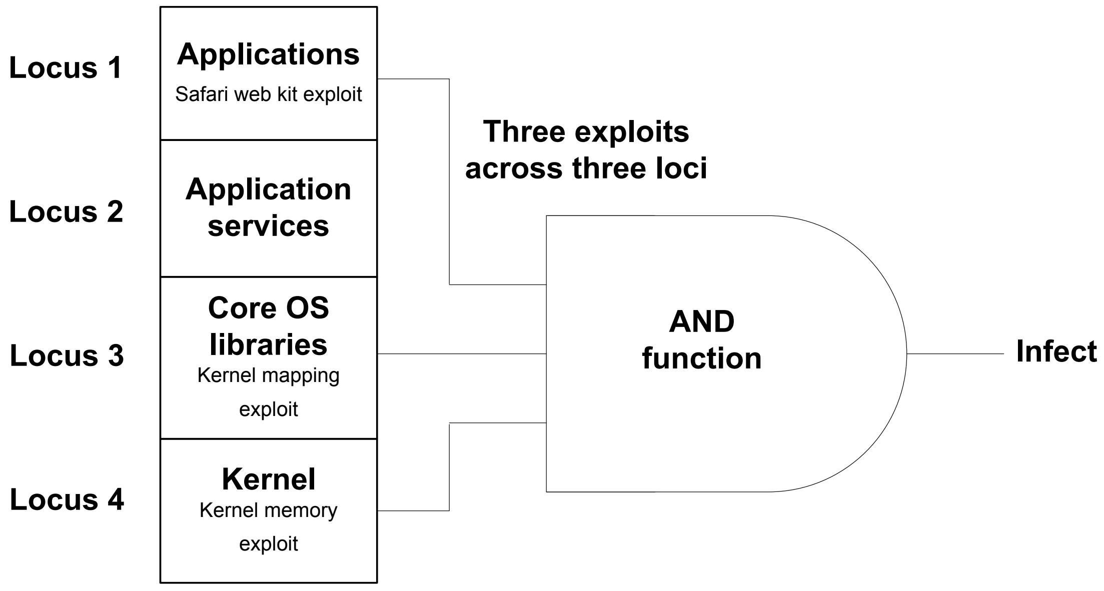


Figure 5-7 – Pegasus malware AND threat representation

Using alternative exploits to target old and new versions of software is becoming increasingly common as many users infrequently install updates or not at all. The CopyCat malware [252] for example iterates through six exploits, mostly at the kernel level, using several well-known Android vulnerabilities in order to gain access to root privileges of a device. When these alternative exploits target different functionality at different loci, an OR relationship is created. This can include malware with the ability to infect and propagate via alternative mechanisms. The *logical OR type*, is representative of malware that needs only one exploit to infect and propagate, but carries multiple exploits which are available for use. With the CopyCat malware for example, exploits targeted vulnerabilities in the camera driver, the IPV4 communications function, and user calls in the API library. Figure 5-8 shows an example based upon three exploits of the CopyCat malware showing a comparative OR threat representation to the Pegasus AND malware.

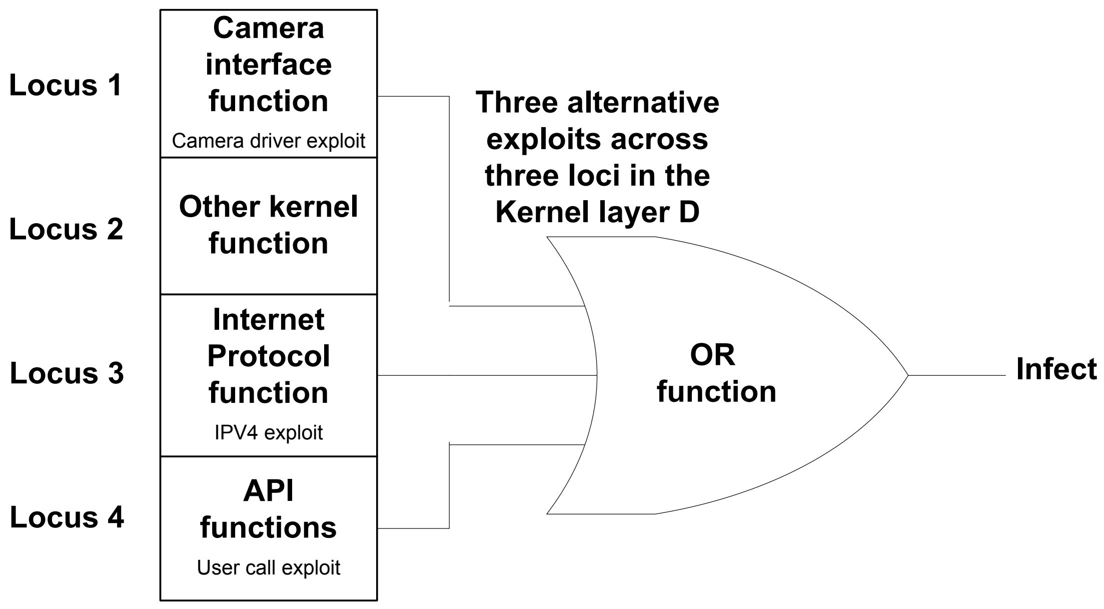


Figure 5-8 – OR threat representation example based upon the CopyCat malware

The reviewed epidemic models of computer security (§3.4.3.2) perceive malware as a single entity attacking a particular type of device. In comparison with the malware threat model this would be equivalent to a single locus without any logic function block, and is compared mathematically in chapter 6 (§6.3.2).

The threat model is currently scoped only for single stage logic using the two AND and OR functions, but could be extended to include multi stage logic to model more complex malware exploit functions. For example a first stage AND function of two loci, and a second stage OR function with a third locus. This could be used to represent a case where malware may propagate using two exploits at different loci, or using a single exploit at a third locus. Although not all of these combinations have been seen in practice, the threat model could allow scenarios to be portrayed that may happen in the future allowing their impact to be determined.

Additionally, the threat model is abstractly representative of any malware with exploits targeting software vulnerabilities and is applicable to any computer system that can be partitioned into layers of functions. However within this research it is applied only to the ad hoc network environment under the assumption that the malware is capable of propagating via peer to peer connections as shown by the malware data flow in Figure 5-9. This is representative of current and future real world scenarios such as those described in §3.2.2, where examples include moving inter vehicular communications, mobile sensor networks and other mobile devices.

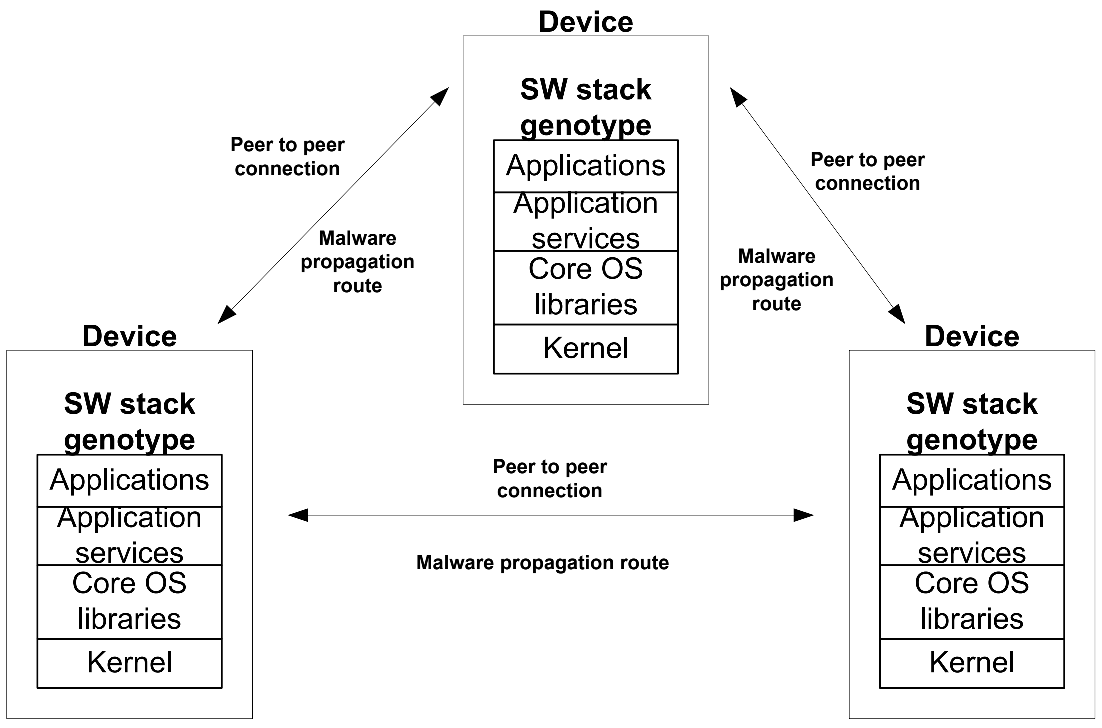


Figure 5-9 – Malware data flow in an ad hoc peer to peer environment

5.3.4 Diverse System Model Overview

5.3.4.1 General Overview

This section describes the general architecture at a high-level of two diverse system models that have been developed to incorporate software diversity and malware at the genetic level of an ad hoc network ecosystem. The intention of the overview is to highlight the key differences between the models and what each method offers. The mathematical content and simulation processes are detailed within the individual chapters for each model (§6,7). The model with the greatest flexibility to incorporate distributed dynamic diversity algorithms, realistic features and constraints follows an agent-based approach (§4.3.4). The model is predominantly simulation based following defined processes that are applied to each and every individual. Under certain constraints this model is comparable to an enhanced mathematical epidemic model, which has also been developed to incorporate software diversity and malware at the genetic level. The epidemic model is predominantly derived and calculated mathematically at a system level without knowledge or control of individual behaviour, and is a key difference between the two models. The epidemic model also provides a means for comparison against standard epidemic models (§4.3.1) as well as the agent model. The two models are outlined below with details of their design and implementation documented in chapter 6 for the epidemic based model and chapter 7 for the agent-based model.

5.3.4.2 *Constrained Diverse System Model: Mathematical Epidemic*

The mathematical based approach is comprised of a network model, a susceptibility model and a malware model as shown in Figure 5-10 and represents a system level view of the whole network.

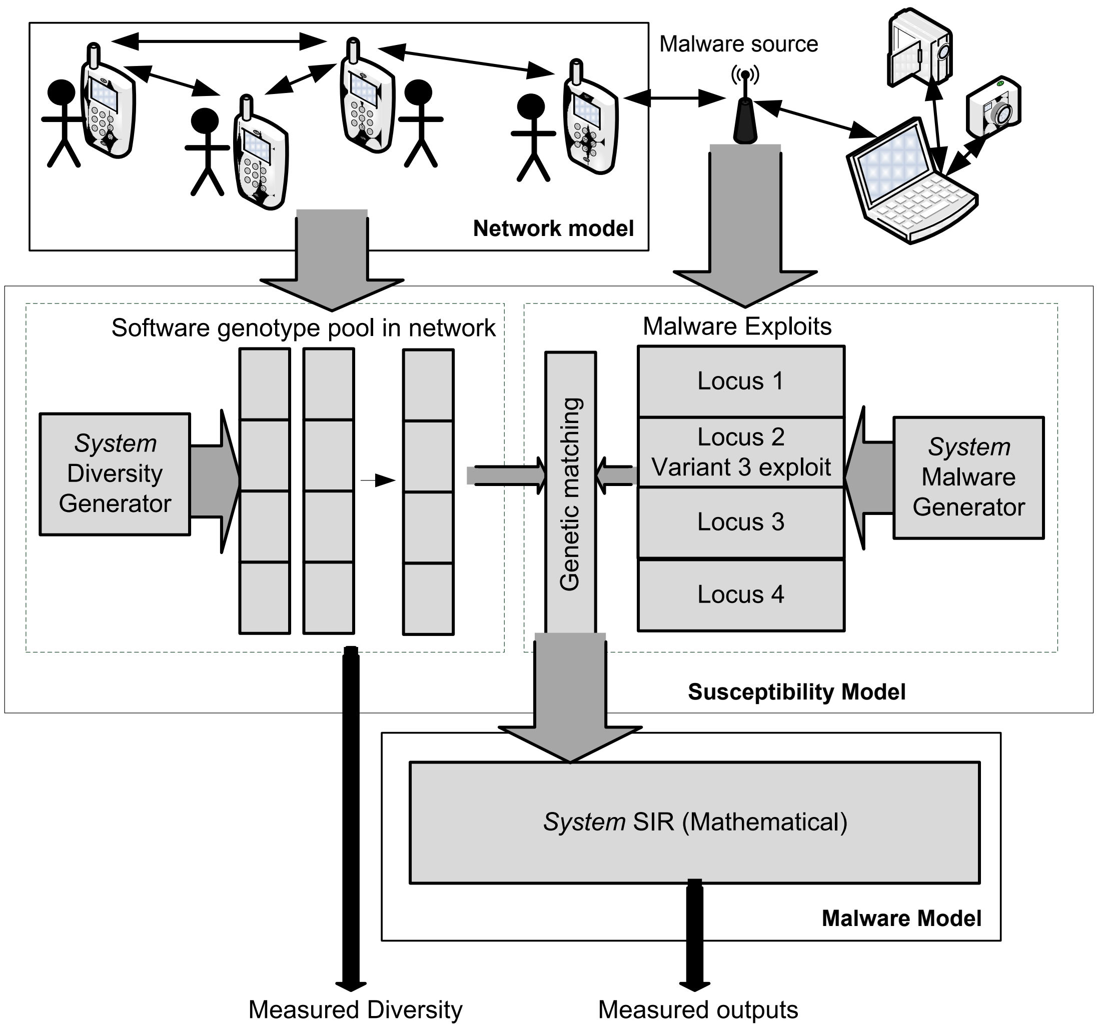


Figure 5-10 – Architecture of the epidemic based diverse system model

Network model: The network model assumes wireless communication protocols are employed utilising peer-to-peer communication with ad hoc devices that move around with their users. Devices can have the same or different software stack genotypes leading to diversity in the network. It is assumed the functionality of the devices combine to produce the ecosystems network and user services. At least one wireless access point is assumed to be present providing a connection to the Internet and an entry point for malware. The inherent nature of a mathematical approach assumes homogeneous mixing of devices and so does not offer the flexibility of the agent-based approach which can incorporate mobility modelling with geographic waypoint information and location based constraints. This highlights another key difference between the two models. However, with the epidemic approach, an analytical result can be achieved under the assumption of average system level conditions.

Susceptibility model: The susceptibility model mathematically derives the susceptibility of the network for a given diversity and malware configuration. The diversity of the network is set or derived at a system level (unlike the agent model where genotypes are individually set at each device). Malware with multi-locus exploits, as defined by the threat model (§5.3.3), is assumed to be capable of spreading within the network. The example malware in Figure 5-10 shows a single exploit targeting variant 3 at locus 2, but could consist of any number of exploits at different loci. The diversity and exploits are used to mathematically determine the proportion of devices that are susceptible to a pre-defined malware. The diversity of the network within a given time frame is constrained to being static so that once a diversity-malware configuration has been created it does not change throughout the dynamics of a simulation (unlike the agent model where the genotype configurations and hence diversity can change). This follows the assumptions made by currently proposed mathematical epidemic models involving diversity (§3.4.3.2).

The difference in this proposed model is that a genetic approach is taken to include different software functions at different layers of the stack together with malware utilising multiple exploits. Current methods assume each node comprises a single software variant and additionally malware targets a single software variant. The principles of genetic matching between exploits and genotypes is similar to the ideas used within infection genetic models (§4.4), but is matched through analytical calculations and is targeted specifically for the malware types defined by the threat model (§5.3.3). With static genotypes, the susceptibility of the network and the average rate of contact (§4.3) between those that are susceptible is pre-calculated before applying the malware model.

Malware model: Parameters generated from the susceptibility model are fed into the malware model to obtain simulated ecosystem outputs. The mathematical malware model can be either deterministically or stochastically based and currently supports either the SI or SIR compartmental models (§4.3). This approach is different to the agent model where individual devices keep track of their own health status which is determined by the dynamics of the individual simulation (rather than the mathematical equations).

The mathematics of the epidemic model is detailed in chapter 6.

The constrained epidemic model offers a method of:

(a) Comparing the proposed genotype structure which consists of software gene variants at different loci that can be targeted by multi-exploit malware with current epidemic models of diversity, where a genotype or node is considered as a complete entity. The model developed can additionally simulate non-maximally diverse scenarios (not considered by other diversity based epidemic malware models §3.4.3.2), allowing the diversity of current networks or networks with domineering software variants, to be analysed in response to malware and compared to the maximally diverse case.

(b) Verifying the agent-based model under homogeneous mixing and static diversity constraints.

(c) Comparing the mathematical model with the agent-based model, which can include additional features such as dynamic diversity, additional security mechanisms, geographic mobility and realistic constraints.

(d) Modelling abstract ecosystem outputs of resistance and resilience and maintaining functional performance (Quality of Service) in response to diversity and specific types of malware attacks.

5.3.4.3 Diverse System Model: Agent-Based

Unlike the epidemic approach where the general architecture (Figure 5-10) is representative of the whole network at a system level, the general architecture of the agent-based approach as shown in Figure 5-11 (of the diversity and malware interaction) represents a single device, and is the same for every device. The agent-based approach is comprised of a network model, a diversity model and a malware model.

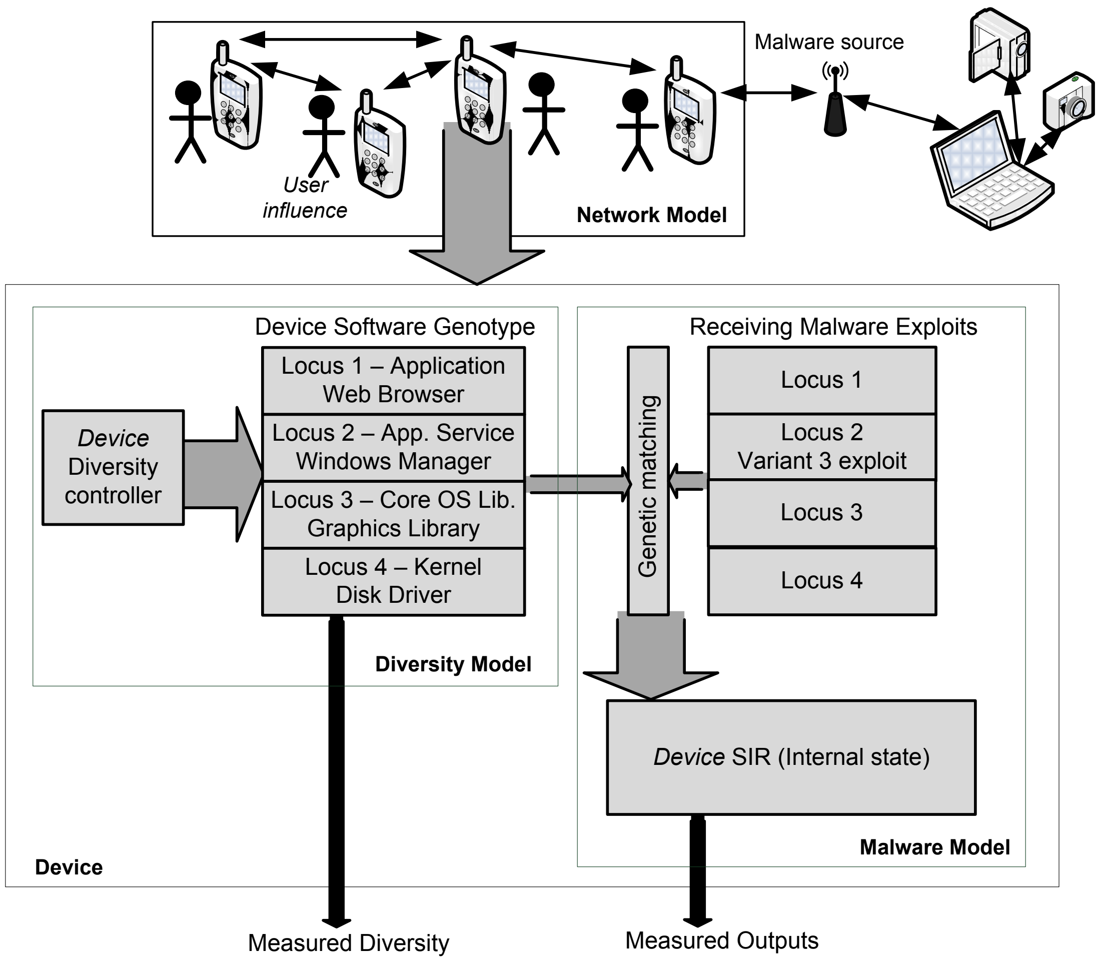

Malware source
User influence
Network Model
Device Software Genotype
Locus 1 – Application Web Browser
Locus 2 – App. Service Windows Manager
Locus 3 – Core OS Lib. Graphics Library
Locus 4 – Kernel Disk Driver
Device Diversity controller
Diversity Model
Receiving Malware Exploits
Genetic matching
Locus 1
Locus 2 Variant 3 exploit
Locus 3
Locus 4
Device SIR (Internal state)
Malware Model
Device
Measured Diversity
Measured Outputs

Figure 5-11 – Architecture of the agent-based diverse system model

Network model: Each agent represents an ad hoc device with software variants performing functions that contribute to the ecosystem's network and user services. As with the epidemic approach, the network model assumes wireless peer-to-peer communication with at least one entry point for malware. With the agent approach however, location based mobility can be modelled. An additional feature, that is not possible with the epidemic approach, is that during local encounters (contact between devices) it is possible to exchange with the contact both genotype information and malware, if it is present. Users may also influence the mobility pattern of devices, or place constraints on the use of certain software variants. The agent-based approach can be additionally constrained to model homogeneous mixing of devices so that verification and comparisons can be made to the epidemic model (§7).

Diversity model: The diversity model uses the genotype structure outlined in section 5.3.1.3 to represent the software composition of a device. The diversity of the network within a given time frame can be either described as static like the epidemic model, where the genotypes remain fixed in each device, or dynamic, where the genotypes may change based upon device level decisions from information obtained during local encounters and is described further in §7. The software genotype on each device is self determined by the diversity model. The diversity controller within each device has its own perspective on the diversity of the network based upon its local encounters, and in response, determines what software genotype should be chosen in order to maximise diversity, subject to any constraints. Practical constraints can be applied together with additional security mechanisms to explore the effectiveness of diversity as an integrated security approach. Whilst the concept of static diversity assignment and dynamic diversity assignment are similar to ideas proposed by colouring algorithms for diversity (§3.4.3.1), the colouring algorithms have been fixed to 3 or 4 colours (software programs) and are simulated on networks with fixed communication links. The algorithms developed here are for ad hoc networks with continuously changing communication links, are multi-layered, and are unbounded in the number of potential software programs.

Malware model: Devices keep track of their own health status which may be susceptible, infected or recovered following the basic SIR epidemic compartments

(§4.3.4), but does not follow the mathematical equations like the epidemic model. Successful genetic matching between the device's own genotype and exploits only occurs if malware is received by the device and the exploits match to vulnerable software components. The malware model uses computational genetic matching between the device's own genotype and a propagating piece of malware, which also takes into account the malware's logical function as defined by the threat model (§5.3.3). If a match occurs the device is deemed to have become infected, and this in turn can change the internal state of the device from susceptible (S) to infected (I). An aggregation of the states of the individual devices provides the system level perspective. These computational matching methods are similar to those used within infection genetic models (§4.4), but as with the epidemic model, is tailored specifically for the defined malware types (§5.3.3). As a result of the interaction of the devices, incorporating genotypes and malware, diversity is measured along with ecosystem outputs such as resistance and resilience.

The agent model offers a method of:

(a) Exploring diversity and malware beyond the limitations of the epidemic approach, through dynamic diversity based on local interactions, user influence and constraints, additional security mechanisms, and geographic mobility.

5.3.4.4 Modelling Environment

The modelling environment used for both models is Mathworks Matlab since it provides a computational environment for modelling the high level abstraction of device behaviours, as well as matrix manipulation for performing simultaneous device operations. Its ability to aid in the generation of GUI's is useful for creating fast modelling interfaces for retrieving, generating and saving settings and simulation data. The built-in libraries help reduce the need to spend time debugging low level code which could otherwise be needed. Disadvantages however are the simulation times and memory usage when simulating large networks. To improve this, manipulation of data types and controlled saving of data during simulations is required.

5.4 Summary

This chapter has presented an ecosystem model for an ad hoc network, making analogies between natural biodiversity mechanisms relating to functionality within ecosystems, and natural diversity mechanisms relating to functionality within ad hoc networks. Malware can be thought of as destructive disturbance events affecting the function and stability of the ad hoc environment. Although the focus here is on ad hoc networks, many of the principles described are also applicable to computer networks in general. In an ad hoc network ecosystem, functionality is predominantly generated by underlying software and hardware components which can be captured in terms of genetics at the individual scale of devices. There are two key components of software at the genetic level affecting ecosystem functionality: That is software gene function and software gene variant. The analogous relationships described imply that the fundamental enabling mechanisms for enhancing diversity already exist within the current structure of software and ad hoc networks. There are methods for measuring diversity at the genetic level that could equally apply to the diversity of software composition across devices.

A diverse system model is required to simulate these mechanisms where individualised software stacks can be represented as genotypes. Some constraints have been applied such as limiting the number of loci to four, with one non-overlapping function being represented from each of the four layers of the software stack. This allows the model to focus upon the effects of gene variation with respect to software variants and malware targeting specific variants with individualised exploits. Single measures of diversity in computing systems have been defined in the literature; however it is proposed here that several metrics are necessary to define computing diversity at the genetic level, all of which provide a different but necessary perspective. A threat model has been defined, focussing upon two types of malware; the logical AND and the logical OR which are representative of malware using multiple exploits to gain entry and propagate. Two system models have been proposed: the mathematical epidemic model, which is detailed within chapter 6, and the agent model which is detailed within chapter 7.

Chapter 6

Constrained Diverse System Model: Epidemic Based

6.1 Introduction

This chapter details the architecture and mathematical derivation of the constrained diverse system model. The constraints as a result of using an epidemic model are described, together with some fundamental questions that the model can address under these constraints. The main aspect is the susceptibility model which generates both diversity and malware and subsequent analytical genetic matching. Two types of malware as defined in the previous chapter have been incorporated, each with varying numbers of exploits. Susceptibility equations are derived for the two types of malware for software stack genotypes having up to four loci. Outputs of the model are defined including resistance and resilience components of stability, along with input constraints so that an optimum diversity can be calculated to either tolerate or prevent a specific type of malware attack.

6.2 System Model Overview

6.2.1 Overview and Constraints

The constrained diverse system model builds upon the basic mathematical SIR epidemic model (§4.3.1) to investigate diversity and malware propagation at the genetic level. The mathematical approach is constrained by four key aspects:

1) Homogeneous mixing, where the system is the average of the individual devices (§4.2.5).

2) Static diversity, where the genotypes present on each device remain fixed throughout a malware epidemic.

3) Software functions are assumed to be compatible with each other so there are no constraints regarding genotype configurations.

4) Individual users have no influence over the choice of genotypes which are predetermined by a centralised source.

Despite these constraints some key mathematical results have been established to answer the following questions under the given constraints:

1) What security protection or mitigation is offered by biodiversity?

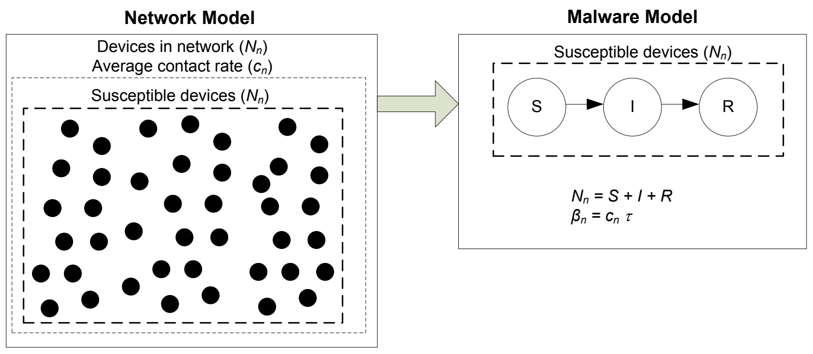
2) How much biodiversity is needed to overcome specific attacks and is there an optimum biodiversity level?

A key feature of this mathematical approach is the susceptibility model which defines the diversity and the malware, and subsequently the susceptibility. Additionally, to incorporate diversity into the *SIR* an equivalent malware model is defined and is described below.

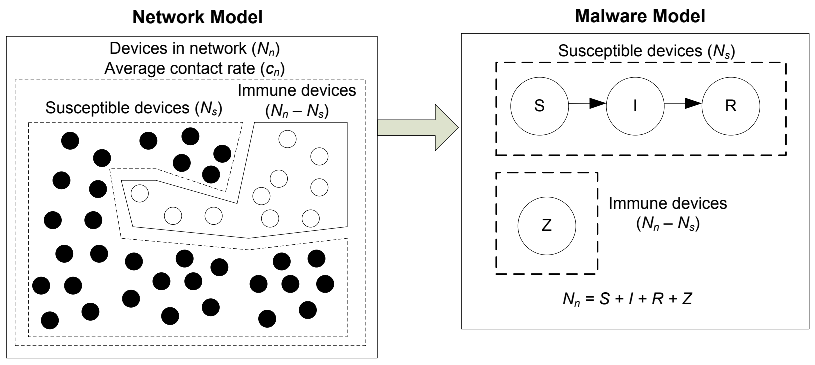
6.2.2 Equivalent Epidemic Model with Diversity

6.2.2.1 Without Diversity

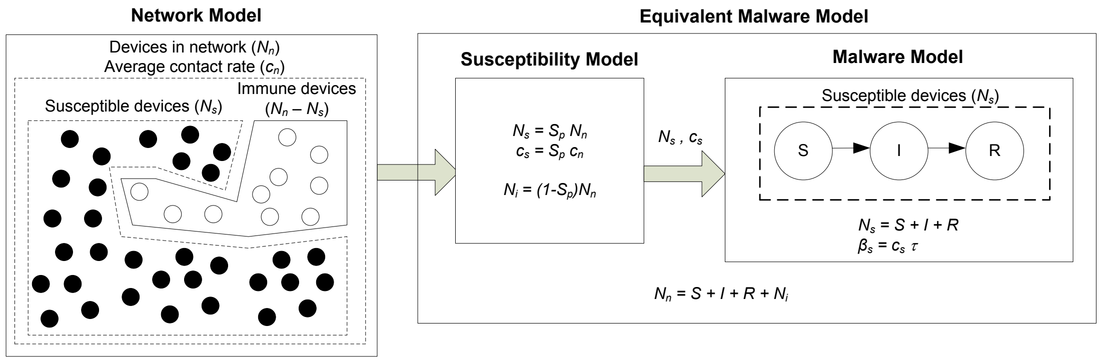
Using the *SIR* as the underlying model (either deterministic or stochastic), it is assumed that a large number of ad hoc devices (N_n) exist, and the devices mix homogeneously where they make wireless contact with each other at an average rate (c_n), as shown in Figure 6-1 (a).


Network Model
Devices in network (N_n)
Average contact rate (c_n)
Susceptible devices (N_n)
Malware Model
Susceptible devices (N_n)
S
I
R
$N_n = S + I + R$
$\beta_n = c_n\ \tau$

a) No diversity - Basic SIR epidemic malware model


Network Model
Devices in network (N_n)
Average contact rate (c_n)
Susceptible devices (N_s)
Immune devices
($N_n – N_s$)
Malware Model
Susceptible devices (N_s)
S
I
R
Z
Immune devices
($N_n – N_s$)
$N_n = S + I + R + Z$

b) Static diversity - SIR malware model with extra immune compartment



c) Static diversity - SIR malware model with an added susceptibility model, where only a proportion of the network is susceptible and the infection rate is also proportionally reduced

Figure 6-1 - Equivalent epidemic model

For a basic *SIR* model as shown in Figure 6-1 (a), where there is no diversity between devices, the entire network is assumed to be susceptible to the malware so that the number of devices (N_n) within the network equates to the number of susceptible devices. During an epidemic simulation, which is modelled by the *SIR* equations (§4.3.2.1), those susceptible may become infected, before recovering through various mechanisms, if they are available, such as malware detection and removal after antivirus updates. The sum of the devices within the S, I and R compartments equate to the number of devices within the network (N_n) which remains fixed throughout an epidemic simulation. The rate of infection (β_n) is the product of the contact rate (c_n) and the probability of transmission (τ).

6.2.2.2 *With Static Diversity*

6.2.2.2.1 *Extra Immune Compartment*

For a network where there is static diversity, only a proportion of the network (N_s) is susceptible, since only those genotypes with exploit matched vulnerable software variants can ever become infected. The remaining devices are considered immune as shown in Figure 6-1 (b). The malware model could be extended so that those devices that are immune could be given another compartment labelled as 'Z', where the sum of the devices within the S, I, R and Z compartments equate to the number of devices within the network (N_s). However, since the rate of entering or leaving the compartment is zero for each specific malware attack, the Z compartment is fully detached from the *SIR* compartments. This results in those that are immune not participating in the dynamics of the epidemic spread, and leaving only those susceptible (S_p) being included.

6.2.2.2.2 *Equivalent Model*

An equivalent malware model as shown in Figure 6-1 (c) can be defined with the inclusion of a susceptibility model to calculate the proportion of devices that are susceptible (S_p) given a specific type of malware attack and diversity scenario. This can then be used to identify the number of susceptible devices (N_s) participating in the spreading dynamics of a known model. The number of susceptible devices within the network (N_s) can be defined as:

$$N_s = S_p N_n \tag{6-1}$$

And those immune (N_i) as:

$$N_i = (1 - S_p) N_n \tag{6-2}$$

Additionally, assuming that the density of the network, and hence the contact rate of the network (c_n) remains unchanged, it follows that the average rate of contact between only those that are susceptible (c_s), is also a proportion (S_p), but of the total network contact rate (c_n) so that:

$$c_s = S_p c_n \tag{6-3}$$

Resulting in a modified infection rate (β_s):

$$\beta_s = c_s \tau \tag{6-4}$$

These results can then be fed into the standard *SIR* model to simulate the output dynamics.

6.3 Susceptibility Model

6.3.1 Overview

Susceptibility is described here as either the proportion (S_p) or number of devices (N_s) which could potentially become infected by a particular malware. The susceptibility model calculates the susceptibility for a given diversity and malware scenario. It is assumed there is a statically diverse network so that pre-computing the susceptibility in this way is valid for the constrained diverse system model. The susceptibility, as shown in Figure 6-2, will depend upon both the diversity of the software gene pool generated by the *system diversity generator*, and the malware generated by the *system malware generator*. The term *system* is prefixed here to signify that the diversity and malware are generated and controlled at the system level for the constrained epidemic model. The diversity of the software gene pool depends upon several parameters including the number of loci, which has been limited to a maximum of four (§5.3.1.3), the number and frequency of software gene variants at each locus, and hence the number of possible unique genotypes (§2.3.2.3). The specific type of malware attack generated depends upon the number of exploits, the targeted loci and variants, and one of two types of malware which are defined in the malware threat model(§5.3.3). The example malware in Figure 6-2 shows a single exploit targeting variant three at locus two, but could consist of any number of exploits at different loci. Genetic matching between these two aspects determines the susceptibility, for which an analytical result has been derived (§6.3.2).

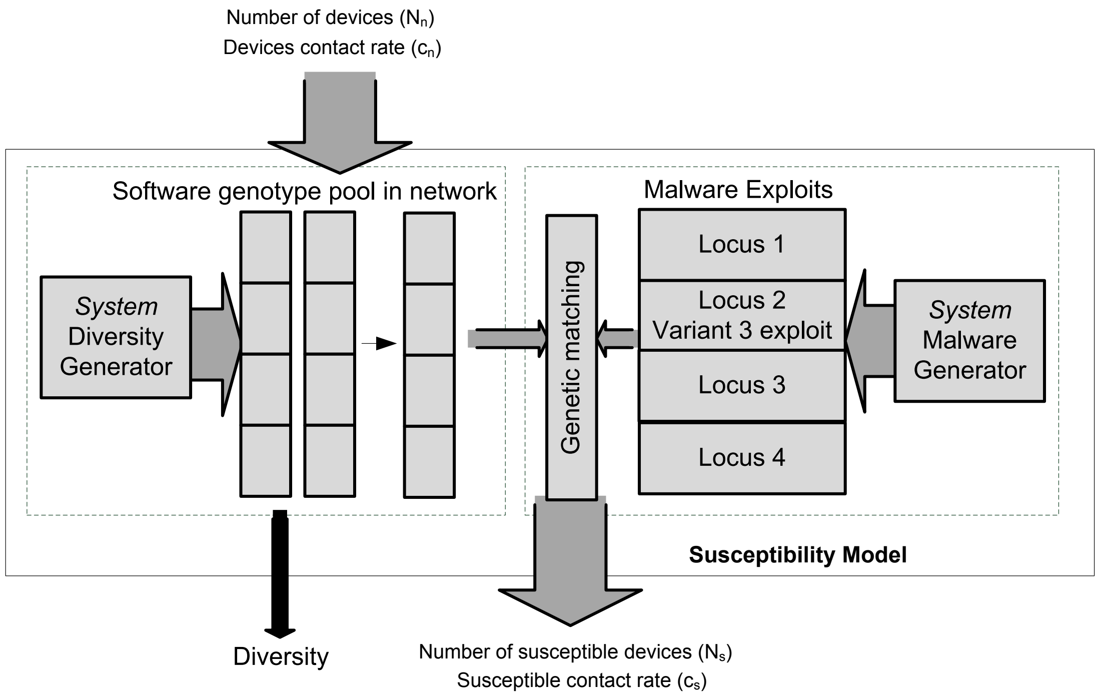

Number of devices (N_n)
Devices contact rate (c_n)
Software genotype pool in network
System Diversity Generator
Malware Exploits
Genetic matching
Locus 1
Locus 2 Variant 3 exploit
Locus 3
Locus 4
System Malware Generator
Susceptibility Model
Diversity
Number of susceptible devices (N_s)
Susceptible contact rate (c_s)

Figure 6-2 – Susceptibility model

6.3.2 Analytical Result of Susceptibility

The remainder of this section describes an analytical derivation of the susceptibility of the network based upon the previously described diversity (§5.3.2) and malware threat model definitions (§5.3.3).

6.3.2.1 One Locus Model

When software stack genotypes are comprised of a single locus as shown in Figure 6-3, the maximum number of unique genotypes available is equivalent to the number of possible software variants at that single locus. All the reviewed epidemic based diversity models of security (§3.4.3.2) perceive diversity as a single dimension in this way such as the number of possible software packages or node types (L) so that for a single exploit (see Figure 5-6 Malware threat model) or malware there is only ever one susceptible genotype (software package or node type). The total number of susceptible devices is then equivalent to the number of times the susceptible genotype (software package or node type) occurs within the network (N). Additionally the security models either assume each device has a unique genotype, or the genotypes are equally distributed so that maximum diversity is assumed, thus equating the susceptibility with one exploit to a value of N/L.

The definition used by Lively [43] for a non-computing genetic diversity epidemic model defines the number susceptible in terms of susceptible genotype frequencies so that maximum diversity is not necessarily assumed. Using this definition the number of devices susceptible in the network (N_s) for a single locus and a single exploit can instead be defined as:

$$N_s = g_i N_n \qquad (6\text{-}5)$$

Where g_i is the frequency of the ith genotype that is susceptible and N_n is the total number of devices in the network. The single layer models will subsequently be referred to as the *'one locus model'* since they are equivalent to a software stack genotype model with one locus.

For software stack genotypes with multiple loci, together with viruses using multiple exploits (within or across loci for the AND and OR types) the above equation will not hold since more than one genotype may become susceptible.

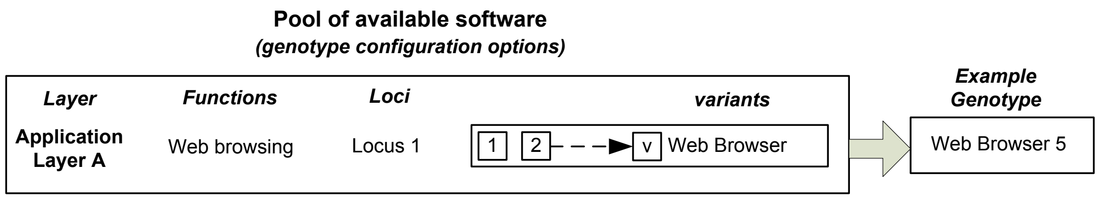


Figure 6-3 - Single locus genotypes – one locus model

6.3.2.2 *Multiple Loci and Multiple Exploits (Genotypic Perspective)*

The remainder of this analytical result describes new work that has been developed. To consider multiple loci and multiple exploits, firstly consider susceptibility from a purely genotypic perspective. Figure 6-4 shows genotypes with multiple loci (up to a maximum of four) with upper bounds on the number of software gene variants at each locus. The number of genotypes (G) is the product of the number of software variants at each locus (§2.3.2.3), which are all assumed to be used in the network.

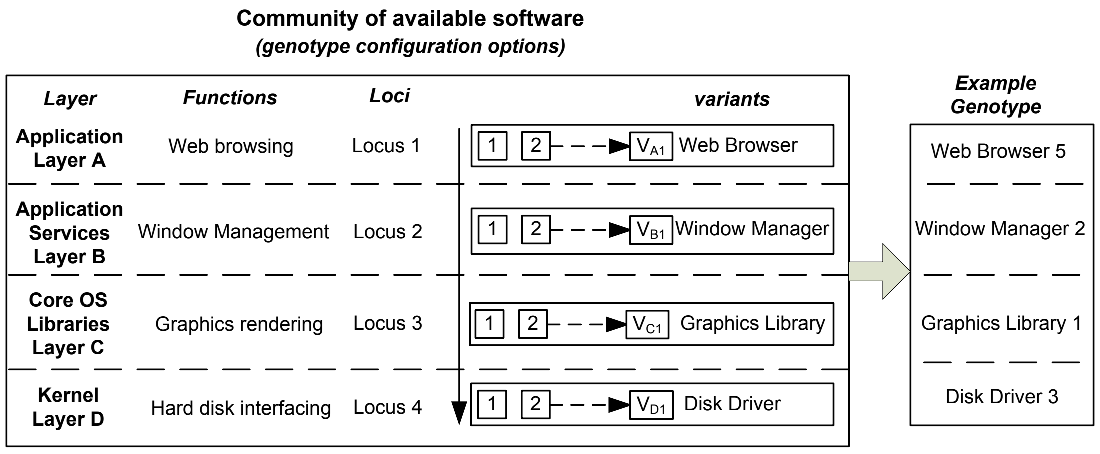


Figure 6-4 - Multiple locus genotypes

The single locus Equation (6-5) can be rewritten for malware with multiple exploits in a multiple locus network. The number of susceptible devices N_s is now the sum of the frequencies of all the susceptible genotypes h (genotypes that match to an exploit) multiplied by the total number of devices in the network. Note that h, which is the number of susceptible genotypes and determined by the number and targeting location of exploits, should not be confused with G, the total number of possible genotypes which can be derived from v and the number of loci.

$$General\ case: \quad N_s = \sum_{i=1}^{i=h} g_i N_n \tag{6-6}$$

In a static network that is at *absolute maximum diversity* (§5.3.2), the frequency of all the genotypes will be equal and the equation simplifies to:

$$Maximum\ diversity\ case: \quad N_s = \frac{h}{G} N_n \tag{6-7}$$

Where h is the number of susceptible genotypes, and G is the *maximum number of unique genotypes.*

Both Equations (6-6) and (6-7) follow the general Equation (6-1) of the susceptibility model where

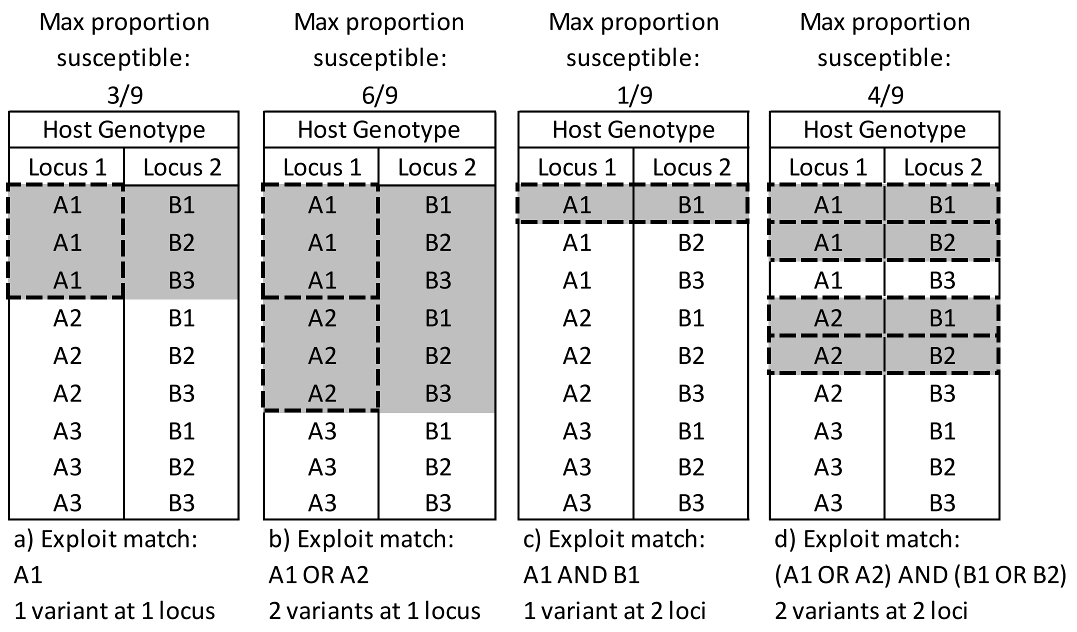
$$N_s = S_p N_n \tag{6-8}$$

$$where\ S_p = \sum_{i=1}^{i=h} g_i\ (general\ case),\ \ or\ S_p = \frac{h}{G}\ (maximally\ diverse\ case)$$

This gives a general result for the proportion susceptible S_p in terms of susceptible genotypes where S_p is the sum of the frequencies of the susceptible genotypes in the general case or the ratio of $\frac{h}{G}$ for the maximally diverse case. The proportion susceptible S_p however can be defined more specifically in terms of loci, variants, and exploits so that for a given malware and diversity scenario the susceptibility can be calculated. The equation for S_p will also change depending upon which of the two, logical AND, or logical OR, malware types is being considered. These equations are derived as follows and forms the analytical method of genetic matching.

6.3.2.3 *S_p for Multiple Loci and Multiple Exploits (Logical AND type)*

Figure 6-5 gives examples of susceptibility for the AND case when there are three software variant choices at each of two loci (A1 to A3 and B1 to B3). In all examples there are nine (3 x 3) possible genotypes. As shown in Figure 6-5 (a), when an exploit targets one software variant on one locus (A1), the proportion of nodes that become susceptible is the frequency (a_1) of A1. Under maximum diversity this equates to 3/9ths (1/3) since it is assumed that the frequency of all genotypes is equal. When two variants are targeted by two exploits (A1 or A2) at the same locus (either of the two variants need to be present in the genotype, equating to both being susceptible), as shown in Figure 6-5 (b), the susceptibility increases to $(a_1 + a_2)$ or 6/9ths (2/3). However, when one variant is targeted on each of the two loci (A1 and B1), as shown in Figure 6-5 (c), the susceptibility changes to $(a_1 b_1)$ or 1/9th (1/3 X 1/3), since both variants *must* be present to become susceptible. As shown in Figure 6-5 (d) when either of two variants on both loci are targeted, the susceptibility increases to $(a_1 + a_2)(b_1 + b_2)$, or 4/9ths (2/3 X 2/3).



Max proportion susceptible: 3/9

Host Genotype	
Locus 1	Locus 2
A1	B1
A1	B2
A1	B3
A2	B1
A2	B2
A2	B3
A3	B1
A3	B2
A3	B3

a) Exploit match:
A1
1 variant at 1 locus

Max proportion susceptible: 6/9

Host Genotype	
Locus 1	Locus 2
A1	B1
A1	B2
A1	B3
A2	B1
A2	B2
A2	B3
A3	B1
A3	B2
A3	B3

b) Exploit match:
A1 OR A2
2 variants at 1 locus

Max proportion susceptible: 1/9

Host Genotype	
Locus 1	Locus 2
A1	B1
A1	B2
A1	B3
A2	B1
A2	B2
A2	B3
A3	B1
A3	B2
A3	B3

c) Exploit match:
A1 AND B1
1 variant at 2 loci

Max proportion susceptible: 4/9

Host Genotype	
Locus 1	Locus 2
A1	B1
A1	B2
A1	B3
A2	B1
A2	B2
A2	B3
A3	B1
A3	B2
A3	B3

d) Exploit match:
(A1 OR A2) AND (B1 OR B2)
2 variants at 2 loci

Figure 6-5 - Examples of susceptible genotypes for the AND type

This methodology of exploit matching to genotypes is similar to gene matching algorithms used in ecology (§4.4) where parasite genotypes are matched to host genotypes. However there is a difference in the matching pattern for computer malware. In the ecology algorithms the parasite genotype is limited so that it can only have one exploiting allele (variant) choice per locus to match the host genotype. Specifically, for the *matching alleles algorithm* [239], this is a limited version of the AND case described here and would equate to Figure 6-5 (c) where the exploit is matched to one variant on each of the two loci. A single malware however could potentially use a different exploit on a different device, especially if it is targeting a similar vulnerability when the variants are closely related.

One Locus: The general equation for the AND case can be derived using probability theory [253] since each locus is independent of the others and therefore whenever one or more exploits target a locus, this can be considered as an independent event. Additionally the proportion susceptible at each locus for a given exploit scenario defines the probability of those susceptible.

The probability (P) of one independent event occurring on one locus j is therefore given by

$General\ case$:
$$P(j) = \sum_{k=1}^{k=x_j} a_k \tag{6-9}$$

$Maximum\ diversity\ case$:
$$P(j)_{max} = \frac{x_j}{v_j} \tag{6-10}$$

Where a_k is the frequency of variant k of those susceptible, x_j is the number of exploits targeting locus j, and v_j is the number of variants in locus j.

Multiple Loci: For multiple independent events occurring (multiple loci targeted by exploits) the probability AND rule (multiplication rule) given in Equation (6-11) can be applied

$$\begin{aligned} P(and, j = 1 to J) &= P(j = 1)\, P(j = 2) \ldots .P(j = J) \\ &= \prod_{j=1}^{j=J} P(j) \end{aligned} \tag{6-11}$$

Where J is the number of loci targeted by an exploit.

This equation defines S_p for the AND type and holds for any number of loci.

Logical AND type

Multiple Loci:
$$S_p = \prod_{j=1}^{j=J} P(j) \tag{6-12}$$

6.3.2.4 *S_p for Multiple Loci and Multiple Exploits (Logical OR type)*

Figure 6-6 give examples of susceptibility for the OR case. When an exploit targets one or more software variants on one locus, the OR case is identical to the AND case as shown on Figure 6-6 (a) and (b). For multiple loci the two cases become different. When an exploit targets one variant on either of the two loci (A1 or B1) as shown in Figure 6-6 (c) the susceptibility becomes $(a_1 + b_1 - a_1 b_1)$, where the subtraction accounts for the genotype that is double accounted for in the summation. Maximum diversity is 5/9ths (1/3 + 1/3 – 1/9). As shown in Figure 6-6 (d) the susceptibility increases to $(a_1 + a_2) + (b_1 + b_2) - (a_1 + a_2)(b_1 + b_2)$, or 8/9ths (2/3 + 2/3 -4/9) when two variants on either of the loci are targeted (A1 or A2, or B1 or B2).

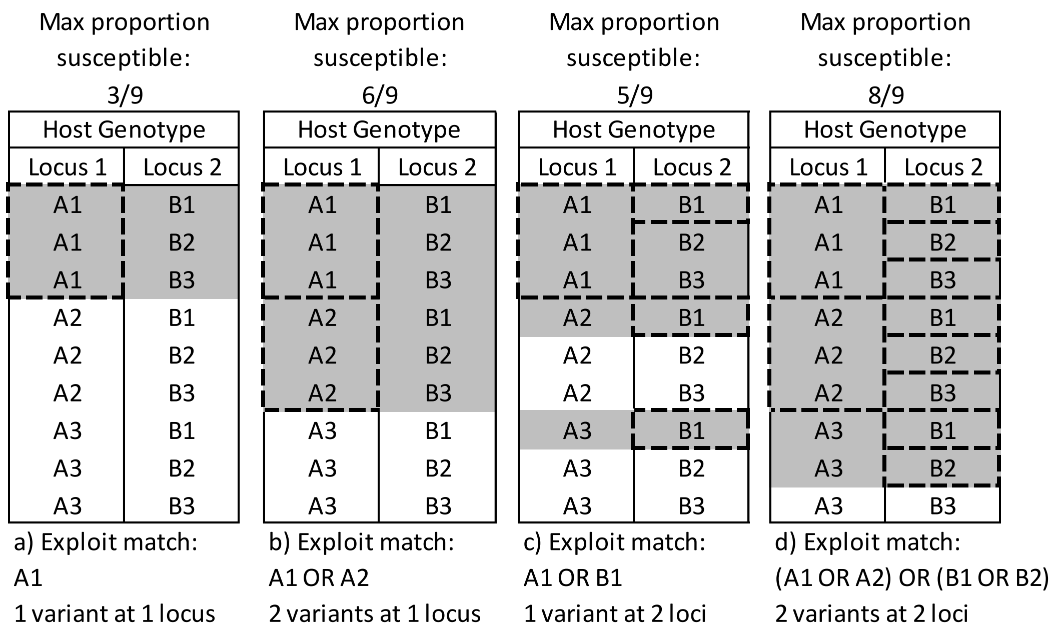

Max proportion susceptible: 3/9
Max proportion susceptible: 6/9
Max proportion susceptible: 5/9
Max proportion susceptible: 8/9
Host Genotype
Locus 1
Locus 2
A1 B1
A1 B2
A1 B3
A2 B1
A2 B2
A2 B3
A3 B1
A3 B2
A3 B3
a) Exploit match: A1 1 variant at 1 locus
b) Exploit match: A1 OR A2 2 variants at 1 locus
c) Exploit match: A1 OR B1 1 variant at 2 loci
d) Exploit match: (A1 OR A2) OR (B1 OR B2) 2 variants at 2 loci

Figure 6-6 - Examples of susceptible genotypes for the OR type

Unlike the logical AND type where equation (6-12) holds for any number of loci, the analytical derivation of the OR type results in different equations for different number of loci.

One Locus: The OR case can also be derived using probability theory. For one locus the OR case is identical to the AND case.

The probability (P) of one independent event occurring on one locus j is given by

$General\ case$:
$$P(j) = \sum_{k=1}^{k=x_j} a_k \qquad (6\text{-}13)$$

$Maximum\ diversity\ case$:
$$P(j)_{max} = \frac{x_j}{v_j} \qquad (6\text{-}14)$$

Where a_k is the frequency of variant k, x_j is the number of exploits targeting locus j, and v_j is the number of variants in locus j.

Logical OR type:

(one locus)
$$S_p = P(j) \qquad (6\text{-}15)$$

Two Loci: For two independent events occurring (two loci targeted by exploits) the probability OR rule (General Addition Rule) can be applied as given in Equation (6-16).

$$P(j = 1\ or\ j = 2) = P(j = 1) + P(j = 2) - P(j = 1)P(j = 2) \qquad (6\text{-}16)$$

And therefore the proportion susceptible S_p for a two locus network becomes

Logical OR type:

(two loci)
$$S_p = \sum_{j=1}^{j=2} P(j) - \prod_{j=1}^{j=2} P(j) \qquad (6\text{-}17)$$

For multiple independent events (multiple loci targeted by exploits) the OR rule can be generally applied iteratively based on the number of loci. The three and four locus derivations are documented in Appendix A (A.1) and (A.2). The result for the four locus is used extensively and referenced within the results (chapter 8) since the underlying model is based upon a software stack genotype with four loci.

6.4 Outputs

6.4.1 Outputs of Current Epidemic Models

Whilst mathematical models of epidemics focus on transmission characteristics and epidemic thresholds, they rarely consider or link these to ecological productivity and stability directly. Also, terminology used within the literature in reference to ecosystems such as stability and resistance has different meanings for epidemic models. For example *Stability analysis* of epidemic models investigates the reaction of the system to small perturbations around equilibrium points (fixed points) determined from the actual equations of the system model [254]. The analysis determines if the points are stable (system moves towards the point) or unstable (system moves away from the point). The term *resistance* either means *drug resistance* which develops when micro-organisms no longer respond to a drug to which they were previously susceptible [255], or *host resistance* which describes how susceptible a particular host is to a particular disease or pathogen [256] [257]. These two aspects are both incorporated within the equations or design of epidemic models so that transmission characteristics and resultant effects can be analysed at the system level.

However, it is possible to infer ecological outputs from epidemic models in some cases. For example the *final size of an epidemic* (§4.3.2.6) describes the system level perspective of the total proportion of individuals that were infected [234]. If instead, the total proportion of individuals that were *not* infected is considered, then this can be used as a measure of system level *resistance*. Some models inherently incorporate recovery parameters such as the classic SIR model (§4.3.1) for which system recovery times, usually discussed as the *duration of the epidemic* [234] or the extinction time of the epidemic [258], can be simulated under different conditions. The duration of the epidemic can be used to infer *engineering resilience* since it indicates how quickly the system can recover from a given scenario. An advantage is that these stability parameters can be measured at a high level of abstraction without the need to simulate user data flow to measure functional performance.

6.4.2 Outputs Overview of the Developed Model

This section defines the outputs from the model using the high level abstraction described above for the two key components of ecosystem stability: resistance and resilience (§2.2.2.2.2, §5.2.5). An overview of the outputs is shown in Figure 6-7. The resistance component (M_R) can be calculated and constrained analytically, and compared to the simulated output. The resilience component (M_L) is determined from simulation since it has no analytical solution (§4.3.2.4), and is described further in §6.4.5.2. However when there are no recovery mechanisms for the malware, the peak infection time (T_1) can be calculated for the deterministic *SI* case. Additionally, an optimum diversity can be determined for a specific type of malware attack given one of two constraints. The first constraint relates to the maintenance of ecosystem function when faced with malware so that an acceptable Quality of Service (QoS) is still guaranteed (§5.2.5). A QoS Tolerance (Q_T) is used to determine a required malware resistance from which an optimum diversity can be determined. The second constraint which is only applicable when recovery mechanisms are in place is to determine the optimum diversity to prevent a major outbreak of the malware. This occurs when the reproduction number is below the critical threshold $(R_0 = 1)$ (§4.3.2.5). The calculated optimum diversity necessary to tolerate or overcome a specific malware attack also leads to the quantification of the minimum number of devices required to uphold this optimum diversity requirement.

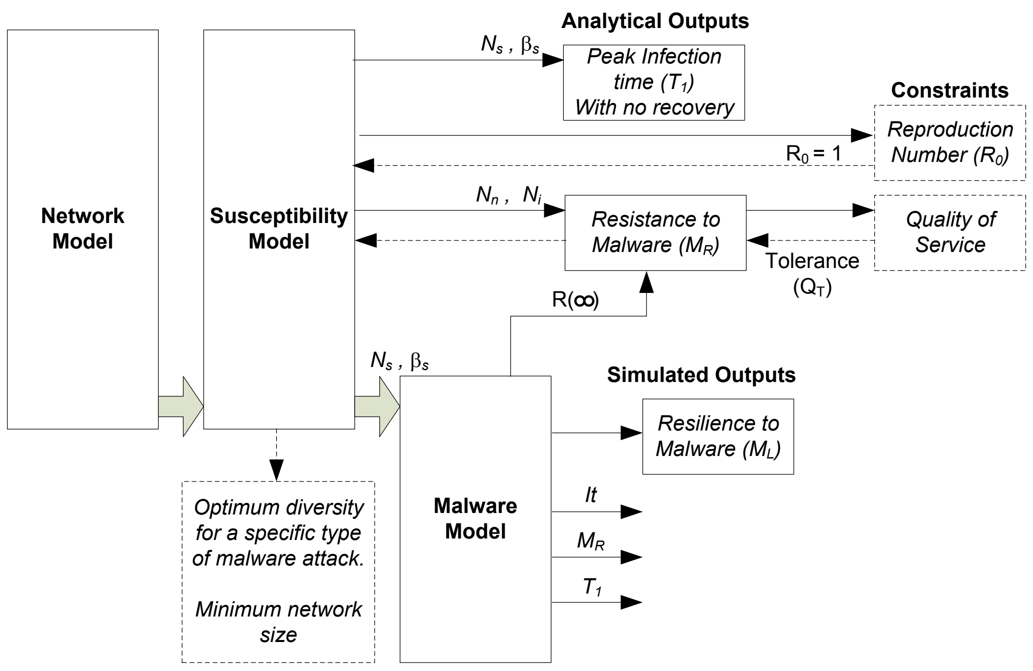


Figure 6-7 - Overview of outputs, constraints and optimum diversity

6.4.2.1 Malware with No Recovery (SI)

Malware can spread in a short space of time relative to the period of recovery. This may be particularly true with zero day exploits where at the time of the attack there is no known patch or even detection mechanism. When there is no chance of recovery in the timescale of the epidemic, the SIR model reduces to an SI model (§4.3.1). The resistance (M_R) of the ad hoc ecosystem to the malware becomes critical, where there may be a tolerance (Q_T), below which ecosystem functions and services become severely degraded (§2.2.2.2.2). The time taken to reach the maximum degradation in services can also be calculated (T_1).

6.4.2.2 *Malware with Recovery (SIR)*

When recovery mechanisms for malware are available within the timescales of the epidemic such as software patching or antivirus signature detection to remove and recover the infected devices, resilience (M_L) as well as resistance (M_R) becomes important. As well as using diversity to maintain QoS, it can also be used to maintain the reproduction number below the critical threshold $(R_0 = 1)$ and prevent a major epidemic outbreak.

The analytical calculations and optimisation methods to determine the necessary diversity to tolerate or mitigate an attack are described in the remainder of this section.

6.4.3 Resistance to Malware (M$_R$)

6.4.3.1 *Resistance to Malware (M$_R$) with no recovery (SI)*

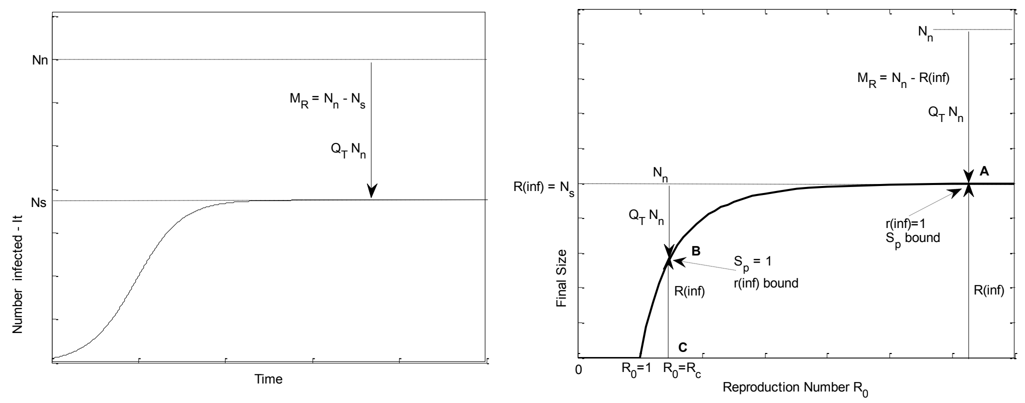
As shown in Figure 6-8 (a), without recovery (SI model), all of the susceptible devices (N_s) will eventually become infected over time and the resistance (M_R) is the number that do not become infected out of a total number (N_n), or the number immune (N_i), so that for a deterministic or stochastic SI model the malware resistance is defined as:

$$M_R = N_n - N_s = N_i \qquad (6\text{-}18)$$

6.4.3.2 Resistance to Malware (MR) with recovery (SIR)

Resistance to malware with recovery is defined in the same way as no recovery: as the number that do not become infected. However with recovery mechanisms in place, resistance is not just attributed to by those immune. The rate of infection and the rate of recovery that make up the reproduction number (R_0), have an effect on the final size of the epidemic (§4.3.2.6 , §4.3.3.6), and hence those that do not become infected. This reproduction number relationship (note the difference in the axis between Figure 6-8 (a) and (b)), as pictured in Figure 6-8 (b) is shown in relation to the final size of an epidemic $R(\infty)$ and the total network size N_n. The malware resistance for a deterministic model and approximated for the stochastic model is defined as:

$$M_R = N_n - R(\infty) \qquad (6\text{-}19)$$



a) Resistance with no recovery (SI) b) Resistance with recovery (SIR)

Figure 6-8 - Resistance to malware (MR) with and without recovery

Sections 6.4.3.3 and 6.4.4 further describe the mathematics of Figure 6-8.

6.4.3.3 Quality of Service Tolerance (Q_T)

As discussed within section 5.2.5 the overall functional performance of services generated by an ad hoc network ecosystem is termed Quality of Service (QoS). In the absence of network simulators to measure specific functional outputs associated with QoS, the best the epidemic model can do is to assume that an infected device has a defined amount of impact on functional performance which in turn degrades the overall QoS. This can be considered particularly true for malware such as viruses, worms, and Trojans (§5.2.4). In the simplest case it can be assumed that infected devices contribute nothing to the overall QoS, whereas uninfected devices contribute fully. This means that once a device has become infected it looses normal functionality, and is only left with the ability to re-transmit the malware to other devices. An infected device degrades QoS in proportion to the number of devices in the network. If only a single device is infected there will only be a small impact, and if all devices in the network are infected the QoS becomes zero. Under this assumption the QoS is represented by the network level output of resistance to malware. However the model could be extended to incorporate the general case where the contribution to QoS is dependent upon the defined specifics of the malware, including both the damage caused, and the mechanisms by which propagation occurs (§5.2.5). The models have not been designed to depict behaviour of the network fabric, however improved techniques for measuring QoS using network simulators is discussed in §9.3.1.2.

The Quality of Service Tolerance (Q_T) is defined here as the required resistance level in order to maintain an acceptable QoS for an ad hoc ecosystem when faced with a malware attack. The QoS, or functional performance of the ad hoc ecosystem may become severely impacted below this resistance level. For example, a Q_T of 0.8 would imply that the ad hoc ecosystem needs to be at least 80% resistant to the malware to maintain an acceptable QoS.

Q_T is the proportional tolerance level of resistance as shown in Figure 6-8 so that when constrained:

$$M_R = Q_T N_n \qquad (6\text{-}20)$$

6.4.4 Optimum Diversity for a Specific Malware Attack

For a specific malware attack it is possible to determine the optimum diversity in terms of the number of software gene variants required at each locus for either a specified Quality of Service Tolerance level (Q_T), or the reproduction number threshold ($R_0 < 1$) to prevent a major malware outbreak. Firstly it is required to determine the resultant constrained susceptibility (S_p), given a specified Q_T, which can subsequently be used with the AND type S_p or the OR type S_p equations (§6.3.2) to find the optimum diversity (§6.4.4).

6.4.4.1 Constrained S_p With no Recovery (SI) and specified Q_T

For a constrained Q_T with no recovery the constrained susceptibility S_p can be determined as follows.

Using the constraint (6-21)

$$M_R = Q_T N_n = N_n - N_s$$

Substitute in $N_s = S_p N_n$

$$Q_T N_n = N_n - S_p N_n$$

Divide through by $N_n$and rearrange for S_p

$$S_p = 1 - Q_T$$

6.4.4.2 Constrained S_p With Recovery (SIR) and specified Q_T

For a constrained Q_T with recovery the constrained S_p can also be determined.

Using the constraint (6-22)

$$M_R = Q_T N_n = N_n - R(\infty)$$

Substitute in $R(\infty) = r(\infty) N_s = r(\infty) S_p N_n$ (from Equation (4-25))

$$Q_T N_n = N_n - r(\infty) S_p N_n$$

Divide through by N_n

$$Q_T = 1 - r(\infty) S_p$$

Rearrange for $r(\infty)$

$$r(\infty) = \frac{1 - Q_T}{S_p}$$

Substitute $r(\infty)$ into the final size Equation (4-25): (6-23)

$$1 - r(\infty) = \big(1 - i(0)\big) e^{-R_0 r(\infty)}$$

And substitute $R_0 = \frac{\beta_s}{\gamma} = \frac{S_p\beta_n}{\gamma}$

$$1 - \left(\frac{1 - Q_T}{S_p}\right) = \big(1 - i(0)\big)e^{-\left(\frac{S_p\beta_n}{\gamma}\right)\left(\frac{1-Q_T}{S_p}\right)}$$

Rearranging for S_p

$$S_p = \frac{(1 - Q_T)}{1 - \left[\big(1 - i(0)\big)e^{-\left(\frac{\beta_n(1-Q_T)}{\gamma}\right)}\right]}, \qquad assuming\ i(0) \approx 0$$

This gives the required proportion susceptible S_p for a specified Q_T and β_n/γ .

There are however three bounds as shown in Figure 6-8 (b). The asymptotic bound (A) occurs for high values of R_0 where the final size of the epidemic approaches all those devices in the network that are susceptible where $r(\infty) = 1$ in relation to N_s. This will happen when the recovery rate is very small relative to the infection rate and resembles the SI model where there is no recovery.

Rearranging Equation (6-22) for S_p

Bound A as shown in Figure 6-8 (b) (6-24)

$$S_p = \frac{1 - Q_T}{r(\infty)}$$

$$= 1 - Q_T\ when\ r(\infty) = 1$$

At this point S_p is defined as its lowest possible value to maintain the specified tolerance. The resultant diversity required to maintain this S_p will be at its highest.

The bound (B) occurs when $S_p = 1$. At this point the rate of recovery is so high relative to the infection rate that all of the devices can be susceptible and the required tolerance can still be achieved. The resultant diversity will be at its lowest for a specific type of attack. Using Equation (6-22)

Bound B as shown in Figure 6-8 (b) (6-25)

$$r(\infty) = \frac{1 - Q_T}{S_p}$$

$$= 1 - Q_T, \qquad when\, S_p = 1$$

The bound (C) is a critical value of R_0 (R_c), which must not be exceeded when all devices are susceptible to maintain at least a specified QoS to keep within the bounds of A and B for a specified quality of service. Using the final size Equation (4-25) with the approximation that $I(0) \approx 0$, and substituting in the bound B for $r(\infty)$:

Bound C - R_c as shown in Figure 6-8 (b) (6-26)

$$1 - r(\infty) = e^{-R_0 r(\infty)}, \qquad where\, r(\infty) = 1 - Q_T$$

Rearranging for R_0

$$1 - (1 - Q_T\,) = e^{-R_0(1-Q_T)}$$

$$Q_T = e^{-R_0(1-Q_T)}$$

$$R_c = R_0 = \frac{-\ln{(Q_T)}}{(1 - Q_T)}$$

Additionally $when\, S_p = 1$ (6-27)

$$R_0 = \frac{S_p \beta_n}{\gamma}, \;\; where\, S_p = 1$$

Therefore

$$\frac{\beta_n}{\gamma} \leq R_c$$

It is also worth noting that when all devices are susceptible the model reduces to a standard SIR model so that when R_0 is reduced further beyond R_c to below a value of 1 a major malware epidemic will be prevented. For a specified Q_T therefore there is a trade off between the speed of recovery and diversity. The faster the recovery (lower R_0 value), the higher the tolerated susceptibility and hence less diversity is required. At the bounds B/C (the critical value of R_0 (R_c)) the minimum amount of diversity is required, whilst at the bound A, the maximum amount of diversity is required. Figure 6-9 shows the relationship between Q_T and R_c. When Q_T is 100%, maximum resistance is specified which can only be achieved when either the susceptibility (S_p) is zero, or when $R_0 < 1$ to prevent a major malware epidemic (§6.4.4.3).

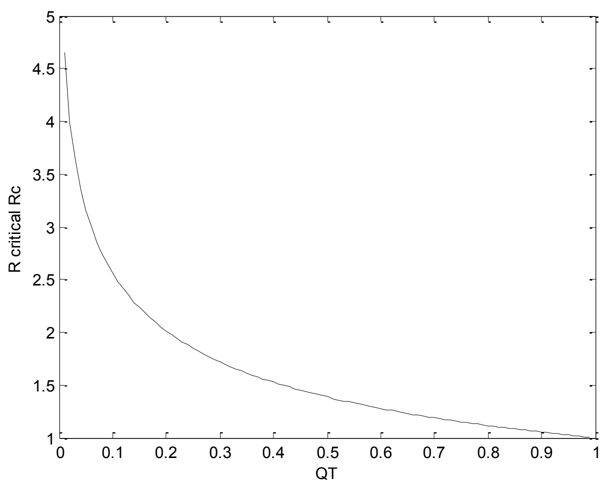


Figure 6-9 - Critical value R_c for a specified Quality of Service Tolerance

6.4.4.3 Constrained S_p With Recovery (SIR) and specified $R_0<1$

To mitigate a specific malware attack by preventing a major outbreak the reproduction number must be less than 1. Again this can be used to constrain the susceptibility S_p and optimise the diversity.

Using (6-28)

$$R_0 = \frac{S_p \beta_n}{\gamma} < 1$$

Rearranging for S_p

$$S_p < \frac{\gamma}{\beta_n}$$

6.4.4.4 Optimisation of Diversity

To optimise diversity three assumptions are made:

1) Variants are evenly distributed amongst devices for all loci so that in order to achieve the Quality of Service Tolerance level or reproduction threshold it is assumed *absolute maximum diversity* can be achieved with the calculated optimum number of variants.

2) The number of variants at each locus has a minimum bound such that the number must be equal to or greater than the number of exploits, since without the least number of variants present the exploit would not exist.

3) The optimum number of variants is defined as the minimum number needed such that *variant richness* is minimised:

$$minimise\ (V_R = \frac{1}{J}\sum_{j=1}^{j=J} v_j) \tag{6-29}$$

The resultant number of genotypes necessary is

$$G = \prod_{j=1}^{j=J} v_j \tag{6-30}$$

6.4.4.4.1 *For the AND malware type (General)*

Assuming maximum diversity can be achieved for a given number of loci and variants then using Equations (6-10) and (6-12) where

$$S_p \le \prod_{j=1}^{j=J} P(j)_{max} = \frac{x_1}{v_1}\frac{x_2}{v_2} \ldots\ldots \frac{x_J}{v_J} \tag{6-31}$$

Rearranging, and assuming S_p is constrained

$$v_1\, v_2 \ldots.\, v_J \ge \frac{x_1\, x_2 \ldots.\, x_J}{S_p},\ \ where\ v_1 \ge x_1,\ v_2 \ge x_2, \ldots.\ v_J \ge x_J \tag{6-32}$$

The product of the number of variants at each locus equates to the *maximum number of unique genotypes* (or genotypic richness) (§2.3.2.3, §5.3.2). The number of variants at each locus that satisfies this number of genotypes and minimum bounds, can have multiple solutions (§2.3.2.3). The optimum solutions for a given malware and QoS tolerance, can be defined from the minimisation of the variant richness (V_R).

Worked example for two loci AND, with no recovery:

$$Suppose\ x_1 = 2, x_2 = 4, Q_T = 0.8,$$

$$so\ that\ v_1 v_2 \geq \frac{x_1\, x_2}{1 - Q_T} = \ 40\ genotypes, and\ v_1 \geq 2, v_2 \geq 4$$

The potential solutions for v1 and v2 are given in Table 6-1 and Figure 6-10 (a) showing the bounds of the solutions. The optimum solutions are shown by the shading in Table 6-1 with a value of 6.5 and in Figure 6-10 (b) by the minimum of the curve of the variant richness solutions.

Table 6-1 - Worked example for the two locus AND type

V_1	V_2	≥ 40 genotypes	V_R
2 (min)	20	40	11.0
3	14	42	8.5
4	10	40	7.0
5	8	40	6.5
6	7	42	6.5
7	6	42	6.5
8	5	40	6.5
9	5	45	7.0
10	4 (min)	40	7.0

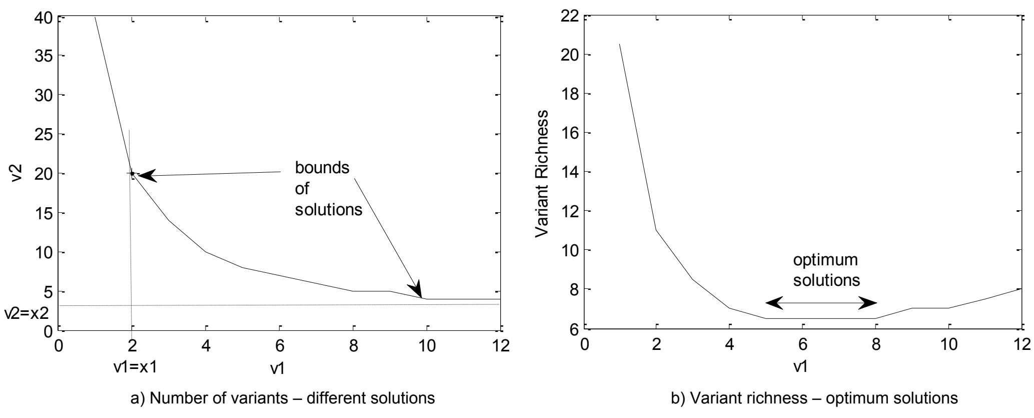


Figure 6-10 - Diversity optimisation example for the two locus AND type

The computational result of an example where there are no malware recovery mechanisms with two loci is given below (results for four loci are given in §8.2.2). Malware is specified with two exploits at locus one, and four at locus 2, with a Quality of Service tolerance of 80%. The result indicates a number of possible optimum solutions with an average of 6.5 variants required. With 40 genotypes, the minimum number of ad hoc devices participating in the network would also need to be 40 in order to satisfy absolute maximum diversity.

6.4.4.4.2 *For the AND malware type (Average)*

For a practical system it may not be possible to specify the exact number of exploits at each individual locus that the network must tolerate or mitigate, instead it may be possible to specify an average number of exploits to obtain an average number of variants at each locus (variant richness). Assuming the number of exploits and variants are the same in each locus, so that $x_1 = x_2 \ldots = x_J, and\ v_1 = v_2 \ldots = v_J$, then Equation (6-32) can be simplified to:

Variant Richness:

$$V_R = v \geq \sqrt[J]{\frac{x^J}{S_p}} \tag{6-33}$$

For the average equation there is only one solution and so minimisation is not required.

6.4.4.4.3 *For the OR malware type (General and Average)*

The OR malware type has been defined for up to four loci. Assuming the absolute maximum diversity can be achieved for a given number of loci and variants then the following optimisations can be defined.

Single locus:

The result for the single locus case is identical to the AND malware type but has been included here for clarity and completeness. Using Equation (6-15) for one locus where

$$S_p \leq P(j)_{max} = \frac{x_1}{v_1} \tag{6-34}$$

Then by rearranging for $v_1$and assuming S_p is constrained the number of variants required can be easily solved for a fixed number of exploits and tolerance.

$$v_1 \geq \frac{x_1}{S_p}, \qquad where\ v_1 \geq x_1 \tag{6-35}$$

The result gives one possible solution for v_1.

Two Loci:

Using Equation (6-17) for 2 loci where

$$S_p \leq \sum_{j=1}^{j=2} P(j)_{max} - \prod_{j=1}^{j=2} P(j)_{max} = \left[\frac{x_1}{v_1} + \frac{x_2}{v_2}\right] - \left[\frac{x_1 x_2}{v_1 v_2}\right] \tag{6-36}$$

Rearranging the equation for v_2 and assuming S_p is constrained. The *general* equation is:

$$v_2 \geq \frac{\left(\frac{x_1 x_2}{v_1} - x_2\right)}{\left(\frac{x_1}{v_1} - S_p\right)}, where\ v_1 \geq x_1, and\ v_2 \geq x_2 \tag{6-37}$$

This will give different solutions for v_2 when v_1 is varied above the minimum of x_1, with the optimum solutions satisfying the minimum *variant richness* (V_R).

The computational result of an example with no malware recovery mechanisms with two loci is given below (results for four loci are given in §8.2.2). Malware is specified with two exploits at locus one, and four at locus 2, with a Quality of Service tolerance of 80%. As with the two locus AND example, the result has a number of possible optimum solutions with an average of 28 variants required generating 720 to 780 genotypes.

Worked example for the two locus OR:

$$Suppose\ x_1 = 2, x_2 = 4, Q_T = 0.8,$$

$$so\ that\ \ v_1 \geq 2, v_2 \geq 4$$

$$v_2 \geq \frac{\left(\frac{x_1 x_2}{v_1} - x_2\right)}{\left(\frac{x_1}{v_1} - S_p\right)}$$

The minimum bounds for the number of variants are shown in Figure 6-11 (a) leaving only valid solutions in the top right quadrant, with optimum solutions being the minimum variant richness as shown in Figure 6-11 (b).

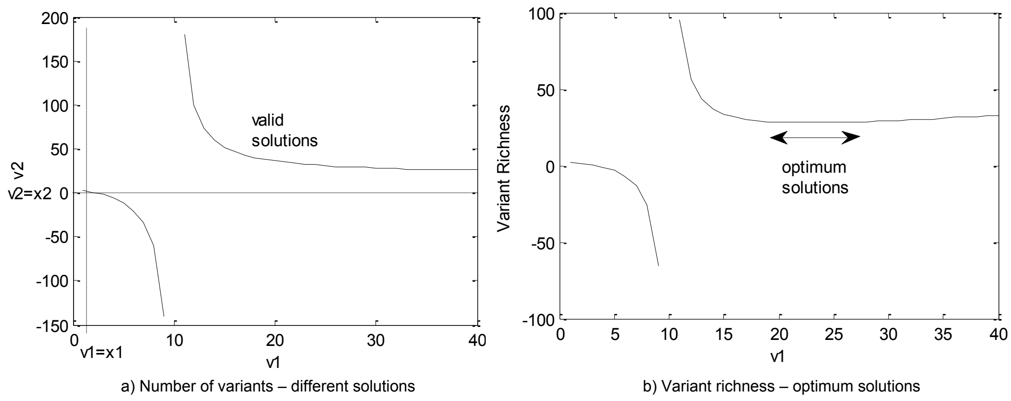


Figure 6-11 - Diversity optimisation example for the two locus OR type

Variant Richness (average number of variants for an average number of exploits): As with the AND type, when an average number of exploits are specified, the number of exploits and variants are assumed to be the same in each locus, so that $x_1 = x_2, and\ v_1 = v_2$, then equation (6-36) can be reduced giving the *average* equation:

$$S_p \leq \frac{2x}{v} - \frac{x^2}{v^2} \tag{6-38}$$

Which can be solved numerically for v.

A similar derivation for the general and average equations can be shown for three and four loci, and is documented within Appendix A (A.3) and (A.4). The determination of the exact number of variants can be solved computationally, results of which are documented in chapter 8 (§8.2.2).

6.4.4.4.4 *For the OR malware type (Approximation)*

When the number of variants (v) at each locus becomes large relative to the number of exploits (x) at each locus, then the summation term dominates the resultant susceptibility (S_p). So that for relatively *large values of* v or correspondingly *small values of* S_p, the OR susceptibility equations can be approximated by:

$$S_{p(approx)} \leq \sum_{j=1}^{j=J} P(j)_{max} = \left[\frac{x_1}{v_1} + \frac{x_2}{v_2} + \cdots \frac{x_J}{v_J}\right], \tag{6-39}$$

$$where\ v_1 \gg x_1,\ v_2 \gg x_2, \ldots .\ v_J \gg x_J$$

For given values of $x_1, x_2, \ldots x_J$ and S_p, v_J can be computed for a range of $v_1 to\ v_{J-1}$ values to find the valid solutions satisfying the minimum *variant richness* (V_R).

Variant Richness (average number of variants for an average number of exploits): If the number of exploits and variants are assumed to be the same in each locus, Equation (6-39) can be simplified to:

$$S_p \leq \frac{Jx}{v} \tag{6-40}$$

Giving:

$$V_R \leq \frac{Jx}{S_p}$$

Approximations and exact solutions for up to four loci are compared in Figure 6-12 for one exploit per locus (Figure 6-12 (a)) and eight exploits per locus (Figure 6-12 (b)) when all loci have the same number of variants. The graphs show the example when a specified QoS tolerance of 80% with no recovery equates to a small value of susceptibility with a large variant richness. In this region the exact solutions are close to the approximated solutions (dashed lines).

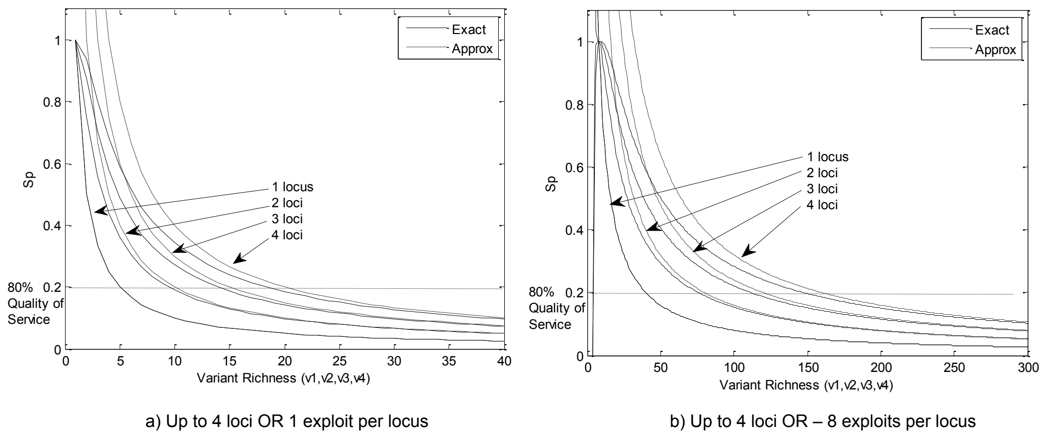


Figure 6-12 - Exact and approximation curves for the OR malware type

Minimum Network Size: When the summation term dominates the resultant susceptibility, all genotypes accounted for more than once in the equations are ignored (Equation (6-39)), thereby eliminating any overlapping relations between loci. Subsequently an even distribution of genotypes no longer matters, only an even distribution of variants to achieve maximum diversity (maximum variant diversity). The required minimum network size to achieve the required constraint is therefore no longer governed by the number of genotypes, but the variant richness so that a smaller number of devices are needed to achieve the required constraint such as QoS Tolerance.

6.4.5 Resilience to Malware (M_L)

For a malware model with no recovery, there is no resilience component; however the time at which the peak infection occurs can be calculated. For a malware model with recovery the resilience can be measured under differing scenarios.

6.4.5.1 Peak Infection Time T_1 of Malware with No Recovery (SI)

For an SI deterministic model the time T_1 at which the infection peaks has an exact solution, when the number infected gets to within 1 of its final value. Equation (4-14) in section 4.3.2.4 defines T_1 as:

$$T_1 = \frac{1}{\beta} \ln\left(\frac{(N-1)(N-I(0))}{I(0)}\right) \tag{6-41}$$

For a partially susceptible network with static diversity, the equation becomes:

$$T_1 = \frac{1}{\beta_s} \ln\left(\frac{(N_s-1)(N_s-I(0))}{I(0)}\right) \tag{6-42}$$

$$substitute\ for\ N_s = S_p N_n, and\ \beta_s = c_s\tau = S_p c_n \tau$$

$$T_1 = \frac{1}{S_p c_n \tau} \ln\left(\frac{(S_p N_n - 1)(S_p N_n - I(0))}{I(0)}\right)$$

6.4.5.2 Resilience to Malware with Recovery (SIR)

For an SIR model, the reciprocal of the time at which the epidemic ends and all devices have recovered to their original operational state signifies the system level rate of recovery, or resilience M_L. However, there is no analytical solution for the end time (§4.3.2.4), but can be measured from the SIR simulation when the number recovered is within 1 of its final value. Equation (6-43) defines M_L as:

$$M_L = \frac{1}{T(R = R(\infty) - 1) - T(0)} \tag{6-43}$$

6.5 Summary

This chapter describes a mathematical epidemic approach to a diverse system model that has been developed to incorporate software diversity and malware at the genetic level of an ad hoc network ecosystem. The mathematical approach is constrained by four key aspects: 1) homogeneous mixing, 2) static diversity, 3) compatible software functions, and 4) non–influential users. Despite these constraints some key mathematical equations and methods have been established to investigate the security protection or mitigation offered by diversity and how much diversity is needed to tolerate or overcome specific attacks under these constraints. The mathematical approach has been developed with the standard SI/SIR epidemic models and can be used with both deterministic and stochastic methods.

A key feature of this mathematical approach is the susceptibility model which defines the diversity and the malware, and subsequently the susceptibility. Two types of malware have been incorporated; the logical AND and the logical OR which are representative of malware using multiple exploits to gain entry and propagate. Equations have been derived using probability theory for the susceptibility of both the AND and OR types for multiple exploits, loci, and software gene variants. The AND and OR methodology of matching exploits to genotypes is different from the standard gene matching algorithms used in ecology since the method used here is more appropriate to malware and the different types. It is more generalised but can also be constrained for specific malware and diversity scenarios. Under any specific malware and diversity scenario the resulting epidemic can be simulated. Current epidemic based malware models of diversity have been referred to as the *'one locus model'* since they are equivalent in this model to a software stack genotype with one locus.

Outputs from the model developed by this work include the two key components of ecosystem stability: resistance and resilience. An optimum diversity can be determined to either tolerate a specific malware attack given a specified QoS tolerance, or overcome an attack when recovery mechanisms such as software patching and antivirus detection are in place. For a specified QoS tolerance there is a trade off between the speed of recovery and diversity. The faster the recovery, the higher the tolerated susceptibility and hence less diversity is required. Under certain constraints approximations can be used to simplify, yet still determine, the optimum diversity required. Worked examples are included showing how diversity optimisation can be computationally determined. The calculated optimum diversity necessary to tolerate or overcome a specific malware attack informs upon the minimum number of devices required to uphold this optimum diversity requirement.

In relation to the hypothesis (§1.2), the development of the epidemic model gives some insight into how incorporating biodiversity concepts into computer networks, specifically ad hoc networks, can make them more resistant to cyber attacks. The model can inform the amount of security protection offered by biodiversity in the form of either tolerance to a specific type of attack, or mitigation to a specific type of attack when recovery mechanisms are available. Under such scenarios the optimum level of diversity necessary to provide the required security protection can be determined.

Chapter 7

Diverse System Model: Agent-Based

7.1 Introduction

This chapter describes the architecture of the agent-based diverse system model. Firstly an *overview of the system* is given, highlighting which aspects are comparable to the epidemic model. Each of its three components; *network model, diversity model, and malware model* are then described over several sections detailing their design and modes of operation. A section on *outputs* describes a number of measured properties including resistance and resilience and how an optimum diversity can be measured for a given scenario. Finally a description of the *implementation* framework is given.

7.2 System Model Overview

The agent-based diverse system model has been designed with greater flexibility than the epidemic based method (§6) with the inclusion of dynamic genotype configuration, geographical location, integration with some existing security mechanisms, and realistic constraints associated with software configuration limitations as a result of users, hardware, or compatibility. However, it is also capable of simulating the same conditions as the epidemic model to allow the comparison of results between the diversity methods and for the comparison of results where the inclusion of additional mechanisms is made.

The agent model has several modes of operation as shown in Figure 7-1 which can be described by the selection of a *network model*, a *diversity model*, and a *malware model*. Selection of those circled by a dashed line indicates the modes that are comparable to the epidemic approach.

As with the epidemic approach, the network model assumes that ad hoc devices move around with their users. Devices have the same or different software stack genotypes that, in this agent-based approach, are either fixed for a period of time or are dynamically changed in response to new information. Additionally in the agent approach it is possible for devices to exchange genotype information upon contact with each other (In the sense of observation of software configuration on the contacted device). A malware source initially infects one device which can then lead to malware propagating within the ad hoc network.

In the agent-based approach there are two choices of *network model*. The first uses the homogeneous mixing assumption (§4.2.5) where *random encounters* are generated between devices. The second uses the *random waypoint* (§4.2.1) approach to inform the physical locations of devices to determine which, when, and for how long nearby devices are in range. The network model is further described in section 7.3.

There are two types of *diversity model*, *static diversity* where the genotypes present on each device remain fixed throughout a malware epidemic simulation, and *dynamic diversity* where the genotypes may change in response to information. The two models are described further in sections 7.4 to 7.6.

The *malware model* is responsible for the generation of malware exploits, genetic matching, and the monitoring of health states. It has two modes of operation, one in which it is assumed there are no recovery mechanisms in place for the malware. In this case each device has two health states of susceptible or infected (SI), which is comparable with the *SI* compartments of the epidemic model (§6.4.2.1). The second mode of operation is when there are recovery mechanisms in place where each device has an additional recovered (R) state, which is comparable with the *SIR* compartments of the epidemic model (§6.4.2.2). The inclusion of additional security mechanisms alters the dynamics between states and is described further in section 7.7. Outputs from the model are covered in section 7.8.

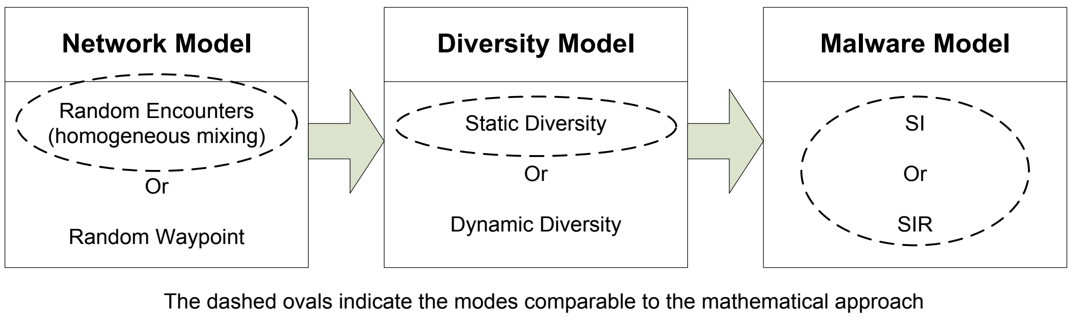

Network Model
Random Encounters (homogeneous mixing)
Or
Random Waypoint
Diversity Model
Static Diversity
Or
Dynamic Diversity
Malware Model
SI
Or
SIR
The dashed ovals indicate the modes comparable to the mathematical approach

Figure 7-1 – Agent-based diverse system model showing modes of operation

7.3 Network Model

The network model has two modes of operation: *random encounters* and *random waypoint*, both of which result in a non deterministic pattern of encounters and are described as follows.

7.3.1 Random Encounters

The random encounters method of homogeneous mixing uses randomisation of encounters between devices that is comparable to the stochastic epidemic model. This network model has been implemented to enable direct comparisons between the epidemic approach and the agent approach with the same input conditions. Additionally, at the same time it incorporates an element of realism that stochastic models try to represent (§4.3.3). The flow chart is given in Figure 7-2. The encounters occur stochastically with an average rate of contact c to model the Poisson process (§4.3.3.1). A random number is chosen for each device from a uniform distribution with a value between 0 and 1. This is used within an inequality equation to validate an encounter against the contact rate c at each time step. The binary result determines whether an encounter has occurred (*encounter flag*). The number of the randomly encountered device is selected from a uniform distribution of those devices in the network. The randomly chosen device is selected such that it cannot be itself. On a successful encounter it is assumed that both genotype information and malware, if it is present, are transmitted, so that the probability of transmission $\tau = 1$. With the random encounter method there is no consideration to the locality of devices, or the length of time taken to transmit the genotype or malware data. This is inherent in the specified average rate of successful contact.

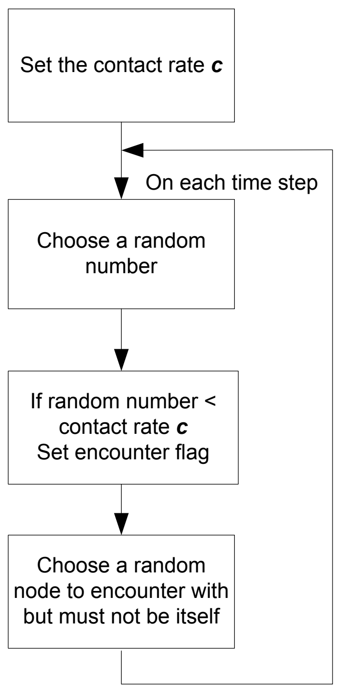

Set the contact rate ***c***
On each time step
Choose a random number
If random number < contact rate ***c*** Set encounter flag
Choose a random node to encounter with but must not be itself

Figure 7-2 - Random Encounters implementation

7.3.2 Random Waypoint

7.3.2.1 Calculation of Waypoints

The random waypoint algorithm has been implemented as described in section 4.2.1 to model the mobility of devices within a confined rectangular area where the selected destination, speed and a stationary time period (pause) of each device is chosen randomly from a uniform distribution. The result is a set of waypoints (x and y coordinates) defining the location of every device at every time step of the simulation. The flow chart of the implementation is given in Figure 7-3. During the calculation process there are small differences between the randomly selected destinations and the actual destinations during each segment of a devices travel path. This is due to integer rounding of the incremental x and y coordinates (delta x and delta y) and therefore the true destination, distance and angle are recalculated.

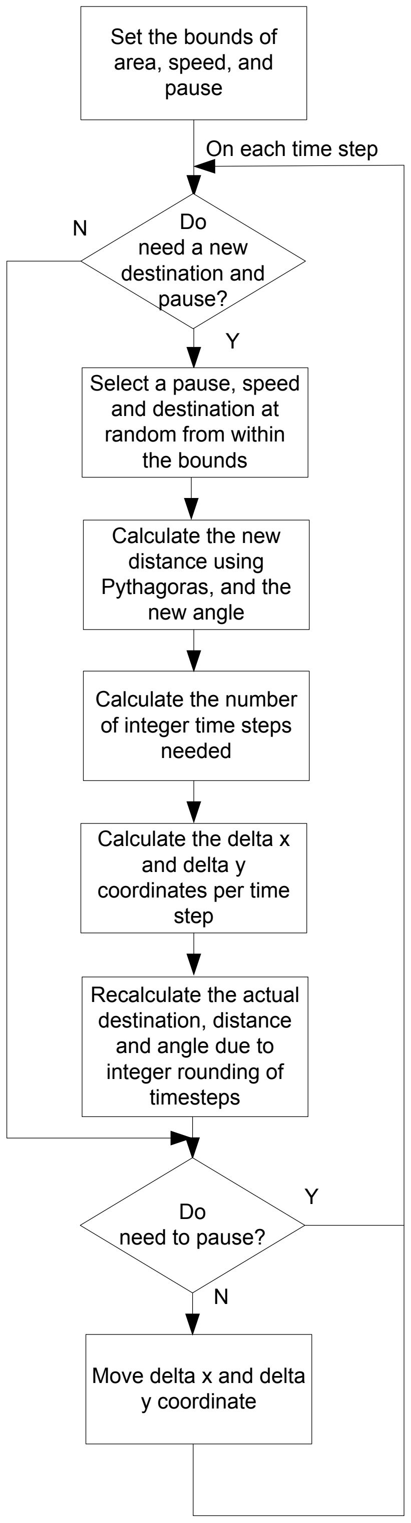

Set the bounds of area, speed, and pause
On each time step
Do need a new destination and pause?
N
Y
Select a pause, speed and destination at random from within the bounds
Calculate the new distance using Pythagoras, and the new angle
Calculate the number of integer time steps needed
Calculate the delta x and delta y coordinates per time step
Recalculate the actual destination, distance and angle due to integer rounding of timesteps
Do need to pause?
Y
N
Move delta x and delta y coordinate

Figure 7-3 - Random Waypoint implementation

The random waypoint model has the flexibility to be extended to include other geographically shaped areas of interest or modified to incorporate non-random waypoints. Additionally the random waypoint algorithm could be replaced with real waypoint data of mobility patterns and is therefore a first step toward modelling geographical location of devices.

7.3.2.2 Selection of Devices in Range

Communication between devices in practical ad hoc networks is controlled through routing protocols (§3.2.3), which form part of a larger network protocol stack (§3.2.1.3). There may be several factors that determine which devices exchange data, and when this occurs, including the requirement to be in range, availability to provide the necessary bandwidth, link strength and link duration. Routing protocols store routing information in routing tables to instruct where particular data is to be sent. In the absence of a simulator (§4.2.2) to model realistic network traffic and routing a more abstract approach is taken based upon several relevant factors as described below.

To model a successful encounter between devices they must be within communication *range*, a parameter which can be configured for the simulation run, for a period of time long enough to transmit genotype information and malware. The method for selection of devices in range has been modelled as one of the following (pictured in Figure 7-4).

1. Nearest in range
2. Random in range
3. Available in range

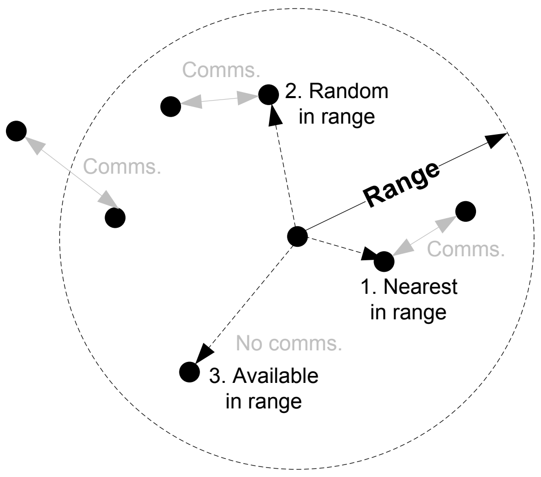


Figure 7-4 – Selection of devices in range

The first two selection methods account for those cases where a device can communicate with another device, if it is in range, at any point in time. This results in devices receiving data from only one other device at a time, but allows them to send data (and potentially malware) to more than one device at a time. This is directly physically possible in a multi-user (MU) communication system such as the Multi-User Multiple Input Multiple Output (MU-MIMO) method [259] where there can be separate communication channels between devices. Even protocols with single user (SU) communications can appear to communicate simultaneously by utilising multiplexing techniques, although the transfer speeds of genotype and malware data would be comparably slower. The random in range selection method allows for comparison with the random encounter model representative of a homogeneous mixing system (§4.2.5) where a device can be selected randomly by more than one other device at any point in time. The nearest in range selection method is an alternative where the likelihood of better link strength and link duration is favoured by the model for the transmission of data.

The third selection method only selects devices that are both in range and are available to provide the maximum link bandwidth since they are not currently communicating with any other device. This could represent a routing protocol where the availability of a device is a favoured factor.

7.3.2.3 Successful Data Transmission

Successful data transmission between a pair of devices can only be achieved if the communication link can be maintained long enough. In this model it is assumed that the genotype data and malware are transmitted in an order consistent with a hierarchical network protocol stack (§3.2.1.3), where the malware cannot be sent prior to link establishment; moreover it is assumed that the entirety of the packetised genotype or malware data must be communicated for successful receipt. The transmission of data between devices is modelled using a tagging system where the selected device in range is tagged to signify the start of data transmission. The tag is released if the device goes out of range or if the data is transferred successfully. The implementation method that determines which devices are in range and when they are tagged is given in the flowchart of Figure 7-5 (a). Euclidian distances between devices are calculated using their x and y coordinate positions to determine if they are in range. The tagging process is described using the state diagram pictured in Figure 7-5 (b). The time required to successfully transmit genotype data and the time required to successfully transmit the malware exploit are two parameters that can be set in the model. When the device is tagged the end times of successful transmissions of genotype (g*enotype end time*) and exploit data (e*xploit end time*) are calculated. If there is no malware then the tag is released at the end of the genotype data transmission (*state 3: Transmit genotype*). If the tagged device goes out of range before one of the end times then the tag is released and the transmission of related data is unsuccessful. Since it is assumed that the genotype data is transmitted before the malware, it may be possible to successfully transmit genotype data, but not the malware.

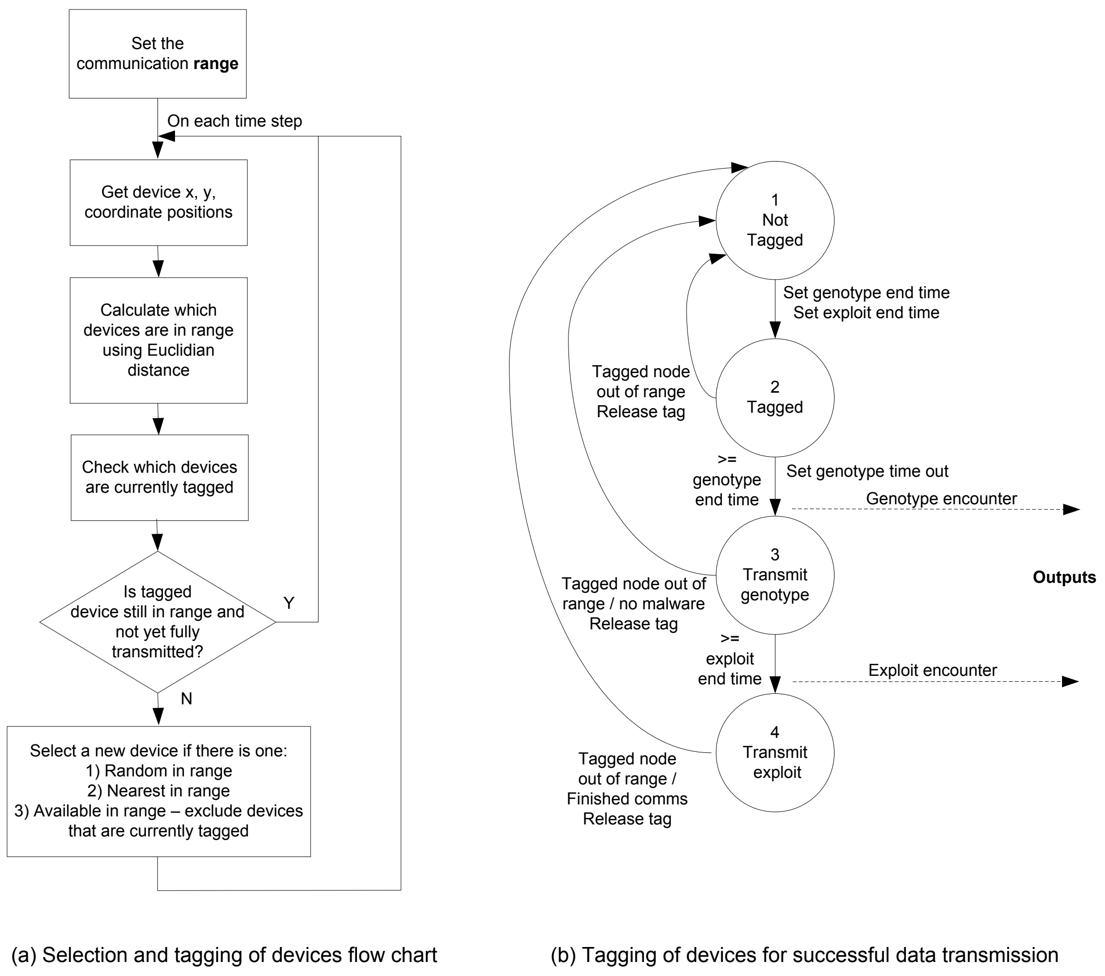

Set the communication **range**
On each time step
Get device x, y, coordinate positions
Calculate which devices are in range using Euclidian distance
Check which devices are currently tagged
Is tagged device still in range and not yet fully transmitted?
Y
N
Select a new device if there is one:
1) Random in range
2) Nearest in range
3) Available in range – exclude devices that are currently tagged
1 Not Tagged
Set genotype end time
Set exploit end time
2 Tagged
Tagged node out of range Release tag
>= genotype end time
Set genotype time out
Genotype encounter
3 Transmit genotype
Tagged node out of range / no malware Release tag
Outputs
>= exploit end time
Exploit encounter
4 Transmit exploit
Tagged node out of range / Finished comms Release tag
(a) Selection and tagging of devices flow chart
(b) Tagging of devices for successful data transmission

Figure 7-5 - Successful data transmission implementation

The resultant output of the Matlab function to the wider model is two single flags per time-step, where one determines if genotype data is successfully transmitted (*Genotype encounter*), and the other determines if the exploit data is successfully transmitted (*Exploit encounter*). A genotype time out period is defined to allow a minimum time before genotype information is captured again from the same device (*genotype time out*). This prevents repeated data capture of the same genotype information in a slow moving scenario. However, if a repeatedly encountered device becomes newly infected, the infection is transmitted assuming it remains in range for long enough.

7.4 Diversity Model - Measuring and Calculating Diversity

7.4.1 Achieving Maximum Diversity in a Practical Ad hoc Network

It is recognised that the more software gene variants and loci there are within a system, the greater the number of possible genotype configurations there are, which could become large (§2.3.2.3, Figure 2-10). It quickly becomes impractical to model network sizes capable of achieving *absolute maximum diversity* for one instance of a network. Additionally, network sizes in practice may range from being very small to very large. When simulating malware propagation, it is sufficient to only include enough genotypes to adequately model the proportion susceptible. Since it is only those susceptible that can ever become infected (§7.5.1).A condition of maximum diversity is that all variants are distributed evenly at each locus (§5.3.2.1). The measured distribution of variants is incorporated into the Nei diversity index calculation (§2.3.2.2.2) and is referred to as *variant diversity* within this research (§5.3.2.1). To achieve *maximum variant diversity* at a single locus there needs to be sufficient devices (N_n) to represent all of the available variants (v). When the number of devices is large enough that all possible genotypes can be evenly represented as well then the *absolute maximum diversity* condition can also be achieved. The agent-based model therefore aims to maximise diversity by maximising variant diversity at each locus independently, regardless of the network size.

7.4.2 Calculating the Maximum Obtainable Variant Diversity

The maximum value of the Nei Genetic Diversity for the monoploid genotype case (§5.6.2.1) occurs at one locus when the number of devices (N_n) is the same as the number of available variants (v) such that every variant is only used once. This leads to the frequency of each variant being $\frac{1}{N_n}$. The maximum diversity index of 1 (§2.3.2.2.2) however is only achievable for large network sizes (and a correspondingly large number of variants), where, using the substitution for the number of different alleles (a) with the number of variants (v) in equation (2-7) of the Nei diversity Index gives:

$$P_d = 1 - \sum_{i=1}^{i=v} (f_i)^2, \tag{7-1}$$

$$where\ f_i = \frac{1}{N_n} at\ maximum\ diversity$$

$$P_d = 1 - \left[v \times \left(\frac{1}{N_n} \right)^2 \right]$$

Substitute in $v = N_n$, to give the Nei diversity as

$$P_d = 1 - \left[\frac{1}{N_n} \right], \qquad where\ N_n \gg 0\ for\ P_d \to 1$$

However, for the ad hoc network being modelled, the number of devices is likely to be greater than the number of variants available at a locus, which may be few in number for practical reasons. In this case the *maximum obtainable diversity* will be less than the maximum diversity (given a fixed network size) and less than 1. At the maximum obtainable diversity the variants are evenly distributed across the devices, so that the frequency of each variant is $1/v$ resulting in the probability (Nei diversity) at a single locus as:

$$P_{dmax} = 1 - \left[v \times \left(\frac{1}{v}\right)^2\right] \qquad (7\text{-}2)$$

$$= 1 - \left[\frac{1}{v}\right], \qquad where\ N_n \geq v$$

For example when the number of variants is only two, the maximum obtainable variant diversity is 0.5, regardless of how many devices (subject to a minimum of two) are present in the network which means that half of the devices will have one variant and the other half will have another variant. When $v = N_n$, equations (7-2) and (7-1) are equivalent. This calculable value provides a reference as to the level of diversity that can be achieved given a finite number of variants under ideal conditions.

7.4.3 Practical Constraints Limiting Variant Diversity

In an ideal scenario it is assumed that all devices are able, and all users are willing, to use any of the different versions of software available at every locus. Additionally, the ideal scenario assumes that software across loci is compatible, such that any genotype can be possible given a fixed number of variants at each locus. In practice however, generating the ideal scenario may be difficult. For example if diversity is achieved through readily available versions of software programs providing the same functionality (§3.2.1.3) they may not be compatible across loci (e.g. operational only with a specific operating system), they may differ in terms of quality and efficiency, and both user desirabilities and hardware limitations may have an influence over which ones are chosen for a specific device. Additionally user desirabilities may differ, depending upon whether they are imposed at a community scale, such as from an IT department where groups of devices may be constrained, or at an individual scale through personal preferences. Additionally, producing automated diverse versions of software or binary files is still in its infancy (§3.2.1.1), and even when implemented there may still be problems interfacing between the different products. These practical constraints could lead to variant or genotype configurations that are unusable or unfavourable, which will influence diversity patterns in the network. The inclusion of constraints in the agent model is detailed in sections 7.6.1.1, 7.6.2.2, and 7.6.2.4.

7.5 Diversity Model - Static Diversity

7.5.1 Distribution of Software Gene Variants

The diversity model has two modes of operation, the first of which is static diversity. Static diversity can either be fixed for all runs of a simulation with a pre-computed data set to achieve a specific distribution of variants and hence genotypes, or the variants can be assigned randomly at the start of every run. The random assignment of variants is used to achieve the maximum diversity possible for a given number of devices, loci and variants. At the start of each simulation run, each device chooses a software variant from the available pool at each locus using a uniform random distribution so that on average the software variants are distributed evenly across devices independently at each locus. This method of random assignment is comparable to the epidemic model of static diversity under maximum conditions (§6.3.2.2). Figure 7-6 (a) shows the measured variant diversity from simulation against the calculated maximum obtainable under ideal conditions (§7.4.2) for a given variant richness with four loci and 1000 devices averaged over 10 runs. The maximum genotypic richness (number of genotypes) (§5.3.2) and genotypic diversity (§5.3.2) that can be achieved is 1000, as shown in Figure 7-6 (b), which is as expected with 1000 devices.

The maximum number of unique genotypes (§5.3.2) surpasses this when $V_R = 5$, however when simulated with enough devices, or averaged over a sufficient number of runs, all unique genotypes will be utilised with equal probability. For a large variant richness a sufficient number of devices or runs become impractical to simulate as described in section 7.4.1, so that a proportion of genotypes will not be represented. However when simulating malware propagation, it is sufficient to only include enough genotypes to adequately model the proportion susceptible. The proportion susceptible is determined by the malware defined. If 50% of the genotypes are susceptible then a network size capable of adequately simulating this could be smaller than if only 1% are susceptible. If only 1% are susceptible the network would need to be relatively bigger (or more runs would be needed) so that on average the susceptibility is adequately represented. In practice for such a scheme it may be both true that there are more genotypes available than currently being used in the network of interest, or less genotypes available than the number of devices. For example ten variants at each of the four loci would generate 10,000 genotypes. For a small network it may be true that only a subset of these are realised at any given time. For static diversity simulated over a number of runs (or different network instances, where genotypes are assigned randomly) would average out so that all genotypes are equally used, even though in a single instance not all would be used.

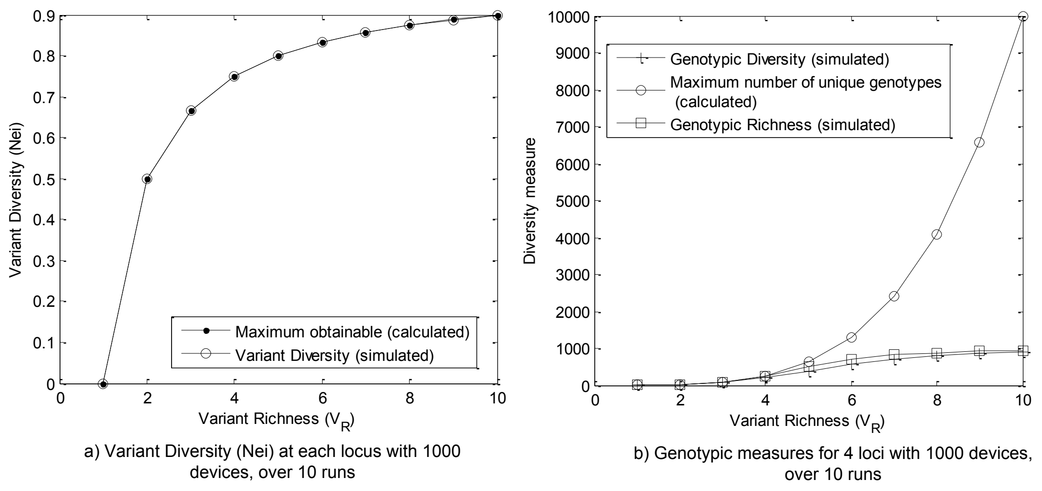


Figure 7-6 - Diversity measures in static diversity mode

7.5.2 Susceptibility

For the agent-based model the health status of every device is initially set to susceptible (§7.7.1) at the start of a simulation run regardless of genotype configuration. However the true susceptibility of the network can be measured under static diversity conditions by matching the generated genotypes against a specific malware attack type, and summing all those that match. Whilst the susceptibility should closely match the result of the epidemic model, under the same diversity and malware conditions, the dynamics of the infection and specific parameters will depend upon the network model. For the random encounter network model, the dynamics of the infection should closely match the epidemic model with the same input parameters.

7.6 Diversity Model – Dynamic Diversity

This section describes two dynamic diversity algorithms, both of which aim to maximise variant diversity given a fixed number of variants, devices and loci. The first is based upon the random selection of available variants and is an extension to the static case that forms a baseline in which to make comparisons. The second algorithm allows individual devices to select variants based upon information obtained during local encounters with other devices and incorporates the geographical locality of devices. The flow charts for the two algorithms are given in Figure 7-7, with the differences highlighted by the double-lined boxes.

The literature suggests that a degree of dynamism in a diversity scheme can be beneficial to confuse a targeted attacker (§3.4.3.1 and §3.4.3.4), however in a moving network where communication links are continuously changing, and in a future Internet where software and malware can rapidly evolve, together with access to vast quantities of data affecting local decisions, the need to be real-time dynamic may be essential.

The algorithms can optionally incorporate two additional mechanisms, as indicated by the dashed lines, of current technology to explore the benefits of a dynamic scheme when integrated with existing security mechanisms such as vulnerability data and virtualisation (VM update). These are in addition to the standard recovery mechanisms included by the epidemic model (§6.4.2.2) and implemented by the agent model (§7.7.3.2). In addition, practical constraints, as indicated by the dotted lines, such as software compatibility and user influence can be switched on to see the effects a realistic scenario may have. The algorithms and the additional mechanisms are further described below.

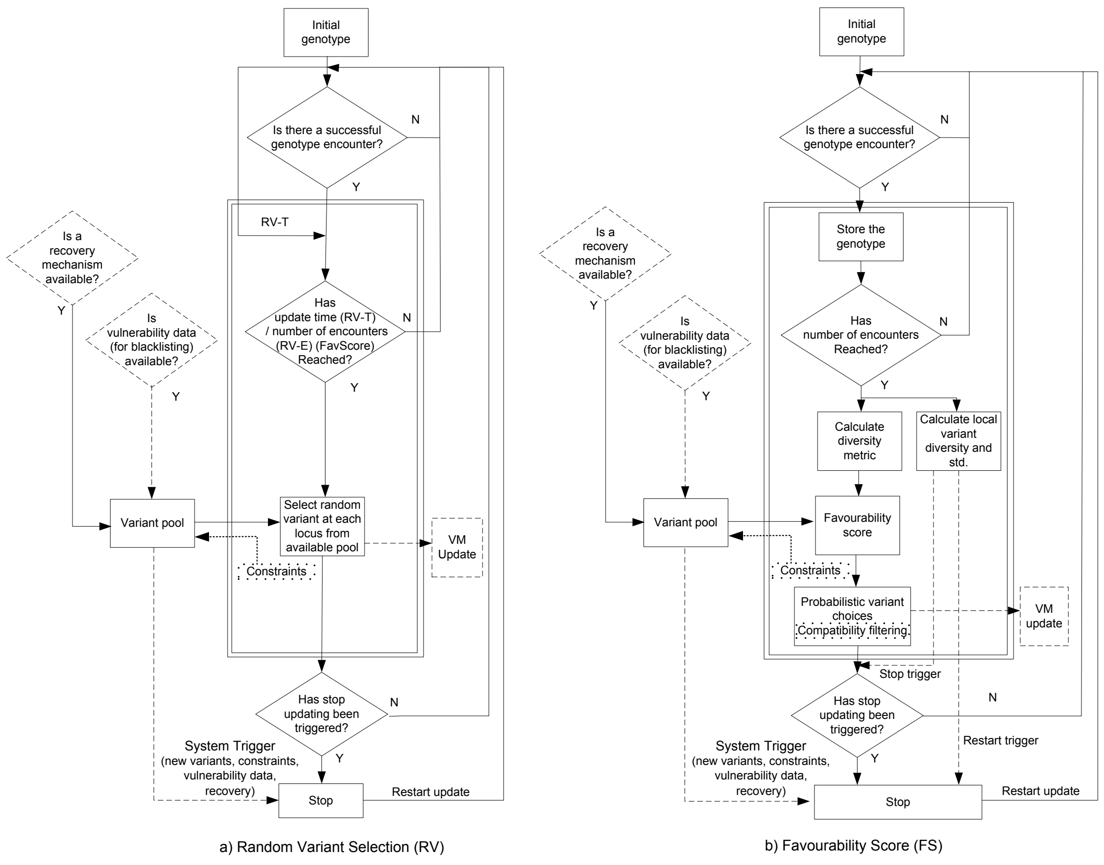

Initial genotype
Is there a successful genotype encounter?
N
Y
RV-T
Is a recovery mechanism available?
Y
Is vulnerability data (for blacklisting) available?
Y
Has update time (RV-T) / number of encounters (RV-E) (FavScore) Reached?
Variant pool
Select random variant at each locus from available pool
Constraints
VM Update
Has stop updating been triggered?
System Trigger (new variants, constraints, vulnerability data, recovery)
Stop
Restart update
a) Random Variant Selection (RV)
Store the genotype
Has number of encounters Reached?
Calculate diversity metric
Calculate local variant diversity and std.
Favourability score
Probabilistic variant choices
Compatibility filtering
VM update
Stop trigger
Restart trigger
b) Favourability Score (FS)

Figure 7-7 - Dynamic diversity algorithms

7.6.1 Dynamic Diversity Algorithm– Random Selection of Variants (RV)

The most simplistic dynamic algorithm, the random variant (RV), attempts to maximise variant diversity by randomly selecting a variant for each locus. This is similar to the static case, except that genotypes on each device can be reselected in response to input triggers such as time (RV-T), encounters with other devices (RV-E), or other system triggers. The flow chart for the random variant algorithm is given in Figure 7-7 (a). Only constraints (§7.6.1.1) or vulnerability data (§7.6.3) can vary the restriction of variant choices at each device.

7.6.1.1 Constraints – User Influence

User constraints can be applied to both algorithms, but differ in the flexibility of the constraints that can be applied. For the RV algorithm, variants are constrained such that they are either available or not available for selection. This coincides with the way in which variants are selected randomly from those available without any bias. For example individual constraints such as physical hardware restrictions or user software preferences may limit those available for selection. When a device is constrained, such that only one variant can be selected, the result is a device with static diversity. Incorporating constraints into the dynamic model will limit the achievable variant diversity (§7.4.3) that the ad hoc ecosystem can achieve when maximisation is being sought.

7.6.2 Dynamic Diversity Algorithm– Favourability Score (FS)

The favourability score algorithm (FS) attempts to maximise diversity by allowing each device to make variant choices based upon local encounters with other devices and exchanging genotype information. Each device independently maintains its own perspective on the local distribution of software variants and variant diversity, and adjusts its own genotype accordingly. The agent model in dynamic diversity mode assumes that the genotype information of encountered devices is visible. For the scheme to become practically viable both a discovery protocol and a trust model would need to be developed to provide the necessary reliable information. There are many discovery protocols in existence for the automatic detection of devices, their services and parameters to connect them. The Bluetooth service discovery protocol for example determines which Bluetooth profiles are supported to determine compatibility [260]. A discovery protocol for the exchange of genotype information would discover software variants rather than profiles. Additionally a trust model would be needed to establish trust relationships between devices to both authenticate the validity of the genotype information, and maintain privacy between trustees. Trust models for ad hoc networks determine trustworthiness of other devices without central authorities [261] allowing devices to participate in various protocols, for example determining trustworthy routes for forwarding data packets [262]. The work in the thesis does not develop a discovery protocol or evaluate trust models for the exchange of genotype information; instead the research firstly considers whether allowing software variant information to be visible upon contact would be of benefit to both diversity and security against existing forms of malware propagation. It also considers whether there are advantages of dynamic diversity using this methodology over static diversity or random assignments of variants.

The flow chart for the FS algorithm is shown in Figure 7-7 (b). During the diversity maximisation period, as each device encounters another device successfully, the genotypes of the encountered devices are recorded. When enough encounters have been made, a parameter which can be set, a diversity metric (§7.6.2.1), is calculated based upon the genotypes of the encounters. If necessary the device will adjust the genotype to a different software variant configuration in an attempt to improve diversity within the network (§7.6.2.3, §7.6.2.4). If restrictions are set such as through vulnerability data (§7.6.3), software compatibility (§7.6.2.4), or user constraints (§7.6.2.2), this will affect the chosen genotype.

7.6.2.1 *Calculating the Diversity Metric*

The diversity metric (dm) is formed independently by every device and indicates which variants could be chosen to improve diversity. Firstly, genotype data is stored by every device in a running buffer with a first in – first out (FIFO) arrangement, the depth of which can be set. The frequency of each recorded software variant (f_v) at every locus is calculated by summing the occurrence of each variant stored in the buffer and dividing by the number of encounters. This indicates how many of each variant is being used locally. To obtain a metric indicating which variants could be chosen to even out their distribution, the frequency is subtracted from unity and then normalized across each locus to 1. This results in variants used frequently being assigned a low metric value and variants used infrequently being assigned a high metric value up to a maximum of 1.

$$dm_v = 1 - f_v, \qquad 0 \leq dm_v \leq 1 \tag{7-3}$$

It is assumed that each device only stores the most recent genotype information so that the diversity metric is calculated from only those stored in the buffer. If all genotypes were stored and used, the data would not be representative of the current local network since both network and software stack configurations will change over time.

7.6.2.2 *Constraints - Individual and Community User Desirabilities*

The application of user constraints is more flexible in the FS algorithm, than the RV algorithm, but can also be limited to match the RV case. The constraints are based around two aspects that limit the maximum obtainable variant diversity

(§7.4.3); a) community scale desirabilities, and b) individual scale desirabilities. The levels assigned are used to influence the software choices of the algorithm when selecting genotype configurations. Unlike the binary constraints of the RV algorithm, here the constraints are applied in the range 0 to 100. Table 7-1 shows an example of how these two aspects could be initialized.

Community scale desirabilities (D_c) – Each software variant has a community scale desirability level, which could be based on the specification of an IT department, or accommodate realistic data for a network such as 70% of users prefer, in an indirect sense, the Google Android core OS library. A number is assigned for each software variant in the range 0 to 100, where the sum of these desirabilities totals 100 for each locus. The community scale desirabilities impose a system level constraint without attributing software to specific individuals. This means that the variant diversity level will be maintained, even though devices are making individual and local decisions.

Individual scale desirabilities (D_I) – The individual scale desirabilities *are* attributed to specific individuals. There is no difference in dynamic variant diversity levels from the community scale desirabilities when the same aggregated percentage of software is set. However, differences in the dynamics of the malware propagation will occur when specific individuals are constrained, for example, to never select the vulnerable software variants. Additionally, differences will occur when either location based constraints are imposed on specific individuals or devices move in non-random mobility patterns. Each software package has an individual desirability level based on the current user's desirability for the software. For example the user desirability may be to use the software represented by variant 1 of locus 1 most of the time, but with a willingness to switch to variant 2, 3, or 4 if necessary. Additionally users may have specific requirements relating to specialised software in order to efficiently perform their responsibilities. In this case it is possible to constrain specific individuals to use a fixed (static) software variant or variants, whilst the remainder of the network tries to maximise diversity. It is important that the diversity scheme does not negatively impact the user experience and so there may be the need to maintain a proportion of specific individual desirabilities whilst maximising diversity of the ad hoc ecosystem. The desirability value is also a number between 0 and 100 for each software package, but may be different within each device. The sum of these desirabilities totals 100 for each locus.

The constraints of user and community desirability data cannot be applied in the same way to the RV algorithm as the FS algorithm due to the random selection of variants. A configuration comparable to the RV constraints would be to apply a desirability value of zero to those variants that are unwanted and apply equal values to those variants wanted. When a device is constrained in the FS algorithm such that only one variant can be selected, the result, as with the RV algorithm, is a device with static diversity.

Table 7-1 - Setting desirability values example

	Variant Number	Community Desirabilities Dc	Individual Desirabilities Di1	...	DiN
Locus 1	1	20	80		80
	2	20	10		15
	3	20	5		3
	4	20	3		1
	5	20	2		1
Locus 2	1	10	20		25
	2	20	20		25
	3	10	20		10
	4	10	20		30
	5	50	20		10
Locus 3	1	70	70		10
	2	25	20		10
	3	2	5		70
	4	2	3		5
	5	1	2		5
Locus 4	1	20	50		5
	2	20	20		5
	3	20	10		50
	4	20	10		20
	5	20	10		20

7.6.2.3 Favourability Score

The favourability score (F) combines the diversity metric (§7.6.2.1), the constraints (§7.6.2.2), and the vulnerability data (§7.6.3). The equation given in (7-4) is termed the favourability score because it 'favours' rather than determines particular software choices.

$$F = \left(w_{dm}dm + w_{D_c}D_c + w_{D_I}D_I\right)B \qquad (7\text{-}4)$$

Where dm is the diversity metric, D_c and D_I are constraints (§7.6.2.2), and w_{dm}, w_{D_c}, w_{D_I}, are weighting factors to weight the importance or inclusion of each summation term. For example to perform diversity maximisation only, both w_{D_C} and w_{D_I} are set to zero. B is a binary matrix of vulnerability data (further described in §7.6.3) with a '0' representing variants that should be blacklisted, and a '1' representing variants that are deemed low risk and safe to use. The binary matrix is used to completely mask out unsafe variants from the list. The model currently assumes that each device has knowledge of the variants available to it to make an informed choice (§5.2.6.1). In practice however different devices may have knowledge of different variants, depending upon how they are generated and stored. For example all devices may have knowledge of COTS variants such as alternative commercial software or open source software modules (§3.2.1.3) as they would be readily accessible, however variants generated via automated code diversification techniques (§3.4.2) may not necessarily be widely available.

7.6.2.4 Probabilistic Variant Choice and Compatibility Filtering

The chosen variant at each locus of the genotype is selected probabilistically and independently based upon the favourability score where a higher score results in a higher probability that it will be chosen. This prevents all devices from choosing the same solution if there is a 'best' option. In an ideal scenario where all variants are compatible across loci this selection method is capable of always choosing operational genotypes. In a realistic scenario however, not all configurations of software may be compatible and so the option of compatibility filtering can also be included. The four steps involved in the decision process for updating a genotype with compatibility filtering are shown in Figure 7-8. In general it is often the operating system that dictates compatibility (§3.2.1.3) and so the core OS library is used as the reference locus in this example, however it could be applied to any locus. Firstly the software variant choices are split into subsets where a mask is created for every OS core library variant to identify compatible software. In the second step an OS core library variant is chosen probabilistically from the OS core library locus using the favourability score. The third step applies the mask of the chosen OS to obtain a filtered favourability score. In the fourth step the remaining locus variants are chosen from the filtered favourability score using the same probabilistic approach.

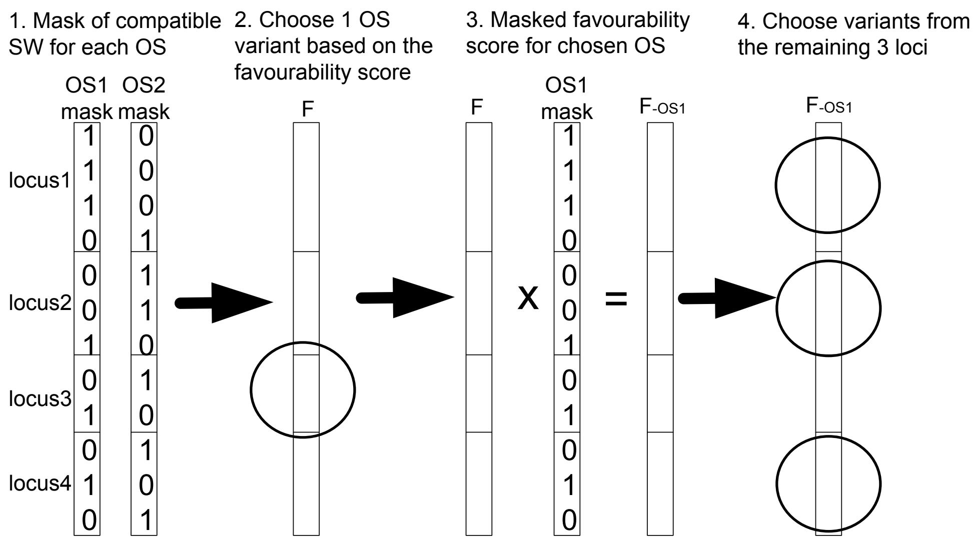

1. Mask of compatible SW for each OS
2. Choose 1 OS variant based on the favourability score
3. Masked favourability score for chosen OS
4. Choose variants from the remaining 3 loci
OS1 mask
OS2 mask
F
F
OS1 mask
F-OS1
F-OS1
locus1
locus2
locus3
locus4
1 1 1 0 0 0 1 0 1 0 1 0
0 0 0 1 1 1 0 1 0 1 0 1
1 1 1 0 0 0 1 0 1 0 1 0
x
=

Figure 7-8 - Updating a genotype with software compatibility filtering

7.6.3 Using Vulnerability Data - Blacklisting of Vulnerable Variants

Using diversity as a stand-alone security mechanism is probably unlikely in practice, and so making use of already available, but untapped security data as part of an integrated approach could better support the benefits of a *dynamic* diversity scheme. When vulnerabilities and corresponding exploits are first publically declared there is often a race between cyber attackers to further exploit the vulnerability and antivirus/software companies to produce antivirus signatures and software patches (§3.3.3). The level of threat perceived by the disclosed vulnerability dictates the amount of time, effort and speed in which antivirus signatures and software patches are released. Even the fastest developed patches may not be enough to prevent a surge in attacks which can occur within a few hours of disclosure (§3.3.3). With dynamic diversity it is possible to temporarily prevent software variants perceived as a security risk due to exposed vulnerabilities from being chosen as a valid variant solution. The term *blacklisting* is used here to denote the mechanism of preventing specific vulnerable variants from being chosen. Software vulnerability information is currently stored in publically accessible databases such as the NIST National Vulnerability Database (§3.4.4.6), or the CVE database (§1.1.2). The automated dissemination of vulnerability information could be released as soon as it becomes available and this would be a lot sooner than the corresponding antivirus signature and software patch, and more importantly, potentially faster than the response from cyber attackers (assuming users allow the diversity scheme to act upon the vulnerability data).

Blacklisting is introduced into the model stochastically with an average rate k at which the blacklisting information is disseminated. This can be set as an independent rate, or a rate dependent upon the contact rate between devices. For example it can be set such that an average of 1 in 10 contacts made are with an access point capable of providing updated blacklisting information. Blacklisting of software variants within the model is undertaken by all devices in the network, however it is acknowledged that in practice some users may wish to avoid changing configurations, for example due to compatibility implications. As shown in Figure 7-7, for both algorithms, as soon as new vulnerability data becomes available it is applied immediately by constraining the choice of variants and re-updating (§7.6.4) the internal genotype. For those that are still susceptible, blacklisting provides a temporary immunity until new antivirus signatures or patching is applied. The effects of blacklisting on the malware model are detailed in section 7.7.3.

7.6.4 Stopping and Starting the Genotype Update Process

The genotype update process is required to make an intelligent genotype selection in order to maximise the variant diversity of the network, subject to the available variants and applied constraints. Whilst frequent changes of variants at some loci may be hidden and go unnoticed by the user, others may disrupt the user experience. Additionally if there is malware already propagating in the network, the act of switching to a vulnerable genotype could spread the malware even further. Therefore once diversity is maximised it may then be beneficial to update genotypes less often, such as only updating when there are new constraints, variants, or other information as shown in the flow chart of Figure 7-7. The start-stop state diagrams for the update process of the RV algorithm and the FS algorithm are shown in Figure 7-9 (a) and (b) respectively.

For the RV algorithm, the local genotype information is not collected and so the only triggers for stopping the update process as shown in Figure 7-9 (a) is either *time* based, or *encounter* based. For example it can be possible to select the genotype once in a single update cycle and then remain static by not updating until a *system trigger* occurs for the device such as the availability of new variants, different constraints, or vulnerability data allowing temporary blacklisting. For the FS algorithm the decision to stop updating is based upon each device's individual perception of the optimisation of the locally measured variant diversity. The local measurements are calculated using the Nei Index equation (2-8) from the recorded genotypes stored in the buffer as shown in the flow chart of Figure 7-7 (b). The running standard deviations of these measurements are calculated over a number of samples to determine how much the diversity level is changing; when optimised there is very little change. As shown in Figure 7-9 (b) when there is at least a *sufficient number of samples* and the standard deviation has progressed below a *minimum threshold, a local minimum* is found by comparing the previous standard deviation value to the current value before stopping the update process. If the standard deviation goes above *a maximum threshold* due to changes in the network by other devices, or a *system trigger* occurs, then the device restarts updating its own genotype again.

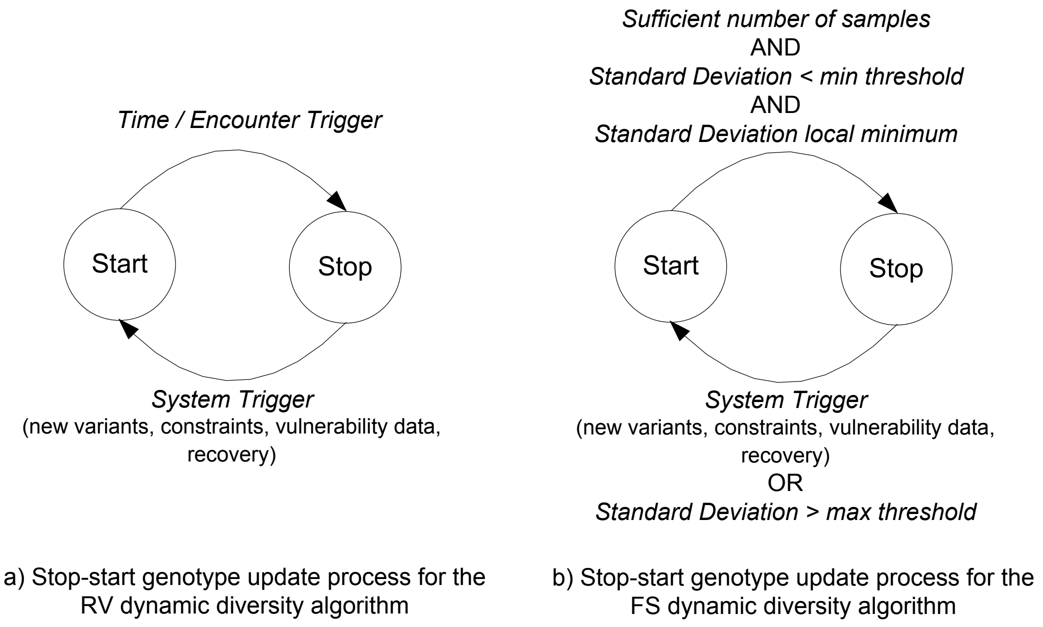

Time / Encounter Trigger
Start
Stop
System Trigger
(new variants, constraints, vulnerability data, recovery)
a) Stop-start genotype update process for the RV dynamic diversity algorithm
Sufficient number of samples
AND
Standard Deviation < min threshold
AND
Standard Deviation local minimum
Start
Stop
System Trigger
(new variants, constraints, vulnerability data, recovery)
OR
Standard Deviation > max threshold
b) Stop-start genotype update process for the FS dynamic diversity algorithm

Figure 7-9 - Stop-start update states for the RV and FS algorithms

The time evolution of a stop-start update sequence from one device using the FS algorithm is shown in Figure 7-10. The variant choice for a single locus, the standard deviation of the locally measured diversity (Std.) and relevant trigger signals are shown. Firstly the data buffer fills with variant diversity measurements (Buffer full) which are used to calculate the running standard deviation. During this period variant choices are being selected as part of the update process. When the local minimum is found below the minimum threshold the device is triggered to stop updating (1. Stop updating). At this point variant number one is chosen. Sometime later an encounter is made with a device with vulnerability data which triggers the blacklisting of vulnerable variants. This also provides a system trigger to re-start the update process, which then halts when the stop conditions are true (2. Blacklisting and update). In this scenario variant one is blacklisted and therefore becomes unselected. When a new patch for the vulnerable variant is installed (Patch download), the device stops blacklisting and re-starts the update process where any of the variants can be selected (3. Patch and update). In this scenario the device had become infected prior to blacklisting and subsequently moves to the recovered state (Recovery).

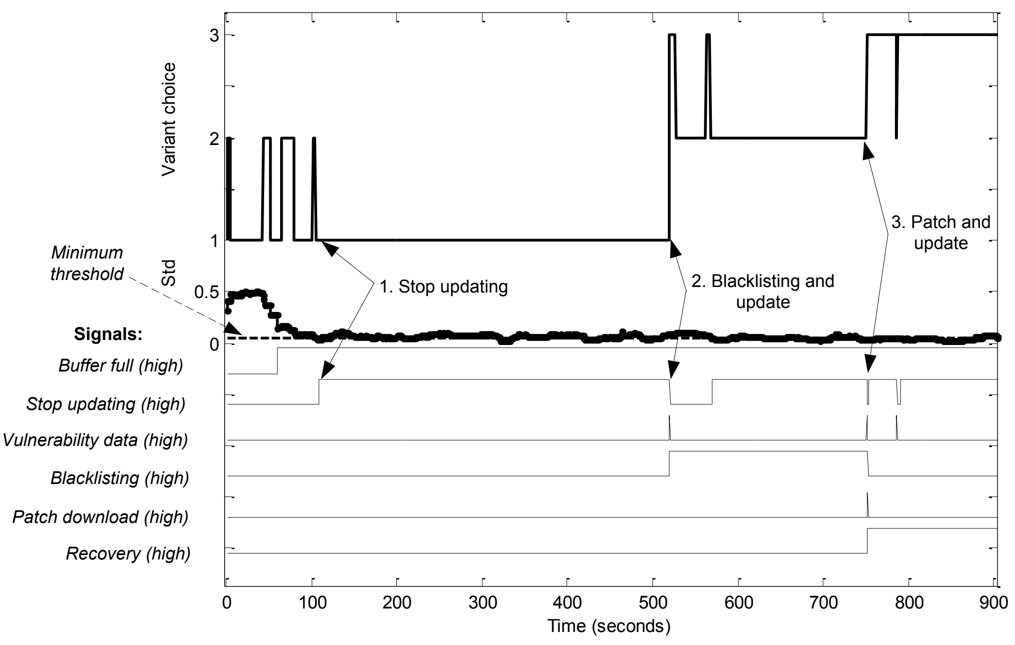


Figure 7-10 - Time evolution of a stop-start update sequence from one device

7.6.5 Virtualisation – Deleting Infected Genotypes

Virtualisation is seen as one of the key enabling technologies for the future Internet (§3.2.1.2). It also has the potential to become a practical platform for the realisation of dynamic diversity, particularly through the use of virtual machines (VMs). Its two key attributes are: firstly the ability to isolate a full or partial software stack from the rest of the device's computer system and data memory; secondly, it is able to run more than one isolated software stack at a time. These two aspects together mean that VMs can be used to configure software genotypes dynamically in the background before swapping with the current configuration. Because each VM is treated as a separate isolated entity, the destruction of an out-of-date VM will also destroy (most) malware since it will be contained within the VM. This is also based on the assumption that new VMs are always created from known *malware free* sources. This is currently the case since known good versions are usually stored by the virtualisation tools which can be reloaded at any specified time (§3.2.1.2). Using virtualisation to swap small chunks of functionally equivalent software has been practically tested (§3.4.3.4) by one research group proving that the concept is feasible, however using virtualisation to manage complete software stacks seamlessly would require further development of the technology and is discussed in future work (§9.3.2.4).

On the assumption that VM technology could provide a practical platform, the dynamic model also incorporates an optional element which, when enabled, models that every time a genotype is updated, any malware present in the current configuration is also deleted. Unlike blacklisting, this does not make the device, temporarily immune: It could still become infected in the future with the same malware. The effects of virtualisation on the malware model are detailed in section 7.7.3.

7.7 Malware Model

The malware model generates the exploits, genetically matches them to genotypes and monitors the health state of each device at every time step of the simulation using the SIR compartments. The health states are aggregated and used to assess the malware attack as it propagates through the network. Unlike the epidemic model, in the agent model the malware can be introduced at any time in the simulation, so that it may be introduced before, during, or after specific mechanisms have been introduced.

7.7.1 Susceptibility

In the agent model *all devices* are assumed to be *initially susceptible*. This is different from the epidemic model which requires that the susceptibility is pre-computed (which remains fixed) in order to assess the dynamics of the malware. This is because within a dynamically diverse network all of the devices are potentially susceptible to any specific malware attack, since any device is capable of adopting any possible genotype. However, in the agent model the time window in which each device is *truly susceptible* will be variable and dependent upon local information, individual decisions and the constrained choice of different variants. It is not necessary to compute this true instantaneous susceptibility to progress the dynamics of the simulation since the mechanism of genetic matching accounts for this by only allowing those instantaneously susceptible (those currently with a vulnerable genotype) to become infected given a successful malware encounter.

7.7.2 Contact Rate and Probability of Infection

For the agent-based model the contact rate c_n is determined by the network model. For the random encounters approach (§7.3.1) the contact rate can be set and is comparable to the epidemic model. It is assumed that once an encounter has been made with an infected device there is a successful transmission of the infection so that $\tau = 1$. For the random waypoint approach there are several parameters that determine the contact dynamics and successful transmission of the malware (§7.3.2). Regardless of the network model, it is assumed that once the propagating malware has successfully entered into a device and has been matched to a device's genotype, the infection is transmitted. This can happen if there are either no antivirus mechanisms in place to detect or block the malware, or the malware is unknown such as in the case of a Zero-day attack.

7.7.3 SIR Compartments

Each device is accountable for its own SIR state. The transitions between the states on any particular device are determined by the interaction of the devices in the network, the diversity of the genotypes, the malware exploit data, and the instantaneous rate values. The flow sequence of the SIR state machine for each device is shown in Figure 7-11. There are four mechanisms that determine the movement between states. The first two mechanisms of genetic matching and recovery are present in the epidemic model. In the agent model however, genetic matching can be influenced by more than just the exploits and contact rate. Additional mechanisms such as changing genotypes, data transmission times, and device locality will also cause an effect, and increasing the realism of the model. The dotted lines represent additional flow mechanisms that are not present within the epidemic model and include the effects from both blacklisting and virtualisation when in dynamic diversity mode. The effects of these four mechanisms are further described below.

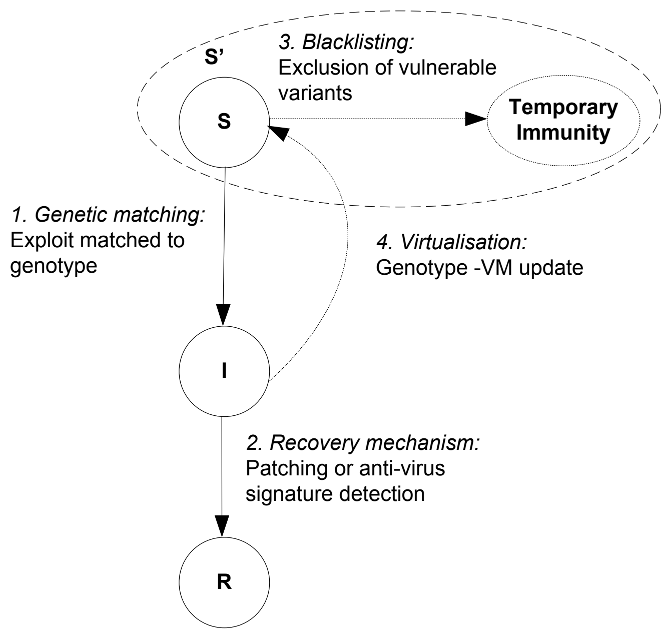


Figure 7-11 - Flow sequence of the SIR compartments in the agent-based model

7.7.3.1 Genetic Matching and Malware Exploit Data

The malware exploit data is constructed in a similar arrangement to the software stack genotype (§5.3.1.3) and is comprised of loci that map to the corresponding loci of the software genotype. The type of malware can either be a logical AND or a logical OR, both of which were fully defined in the malware threat model (§5.3.3). As shown in the flow chart of Figure 7-12, genetic matching between incoming malware and the software stack genotype determines if the device can become infected. For the random encounter network model (§7.3.1), a successful malware encounter happens at the same time as a successful genotype encounter since both data are assumed to be transmitted instantaneously upon a single successful encounter. With the random waypoint model (§7.3.2) it is possible to independently define the time taken to transmit the genotype and the time taken to transmit the exploit data so that a successful malware encounter only occurs if the encountered device is infected and has been in range for the defined length of time.

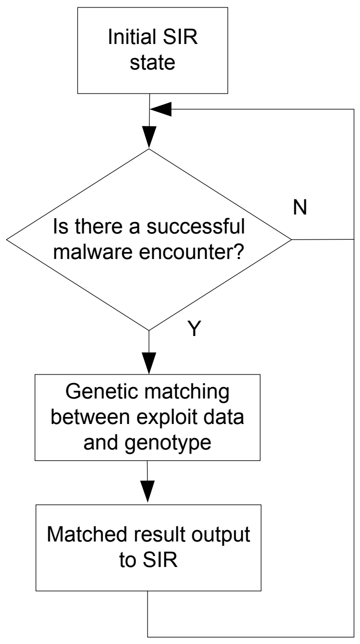

Initial SIR state
Is there a successful malware encounter?
N
Y
Genetic matching between exploit data and genotype
Matched result output to SIR

Figure 7-12 - Genetic matching flow chart

7.7.3.2 Recovery Mechanisms

As with the epidemic model recovery mechanisms are assumed to occur at a *recovery rate* after infection. Such mechanisms include patching or antivirus signature detection to remove malware that was previously unknown at the time of the initial outbreak. When no recovery mechanisms are included for the duration of the epidemic, only the susceptible (S) and infected (I) states can be reached. When recovery mechanisms are included a recovery rate γ can be set for each device. This type of recovery mimics the epidemic model when they are all set to the same value and initialised at the start of the initial malware outbreak. Similar to the generation of the contact rate in the Random Encounter model, the recovery of the devices occurs stochastically with an average rate of recovery γ to model the Poisson process. A random number is chosen for each device from a uniform distribution with a value between '0' and '1' and is used within an inequality equation to validate the recovery against the recovery rate γ at each time step.

7.7.3.3 The Effects of Blacklisting

Blacklisting is applied to any device regardless of which state it is in (since the device itself cannot detect an infection in this model §9.3.1.2), however it is only of use when devices are still susceptible and would be equivalent to an extra *temporary immunity* state as that pictured in Figure 7-11, where the device is prevented from becoming infected. The S' state denotes that the actual effect of blacklisting for those that become temporarily immune is to remain in the susceptible state. The start of blacklisting information dissemination may be at the point of initial malware infection if the vulnerability has just been disclosed but could equally have been at some point in the past if prior knowledge of a potential threat was received, or further in the future if it models a zero day attack where the vulnerability and knowledge of an exploit is still unknown. For those already infected blacklisting offers no protection and it is assumed these devices remain infectious, even though the vulnerable variant is no longer used. When the signatures or software patches are released to detect and remove the malware, the infected devices can recover and the temporary blacklisting can be removed.

7.7.3.4 The Effects of Virtualisation and Deleting Infected Genotypes

Like blacklisting, the use of VM's to create and destroy genotypes, is applied to every device regardless of what state it is in, but is only of use when devices are actually infected since it removes undetected malware. During a genotype - VM update cycle, if the device is infected it will move back to the susceptible state as shown in Figure 7-11. Simulations and analysis involving virtualisation are conducted in conjunction with blacklisting (§8.3.4).

7.8 Outputs

This section defines the outputs from the agent model as shown in Figure 7-13, most of which are comparable to the epidemic model. Only two analytical outputs are defined for the agent model, all other outputs are determined from simulation including the two key components of stability: resistance and resilience. A key benefit of the epidemic model is its analytical solutions. This enables an optimum diversity to be calculated in terms of the number of variants at each locus to tolerate or mitigate a specific type of attack for a given scenario. In constrained mode, the agent model is comparable to the epidemic model and therefore simulated outputs are compared to those calculated. Beyond the bounds of the epidemic model, with the inclusion of dynamic diversity, device geographical location, blacklisting, virtualisation and practical constraints, the results are analysed purely through simulation and comparisons with the epidemic based results. The outputs are summarised as follows.

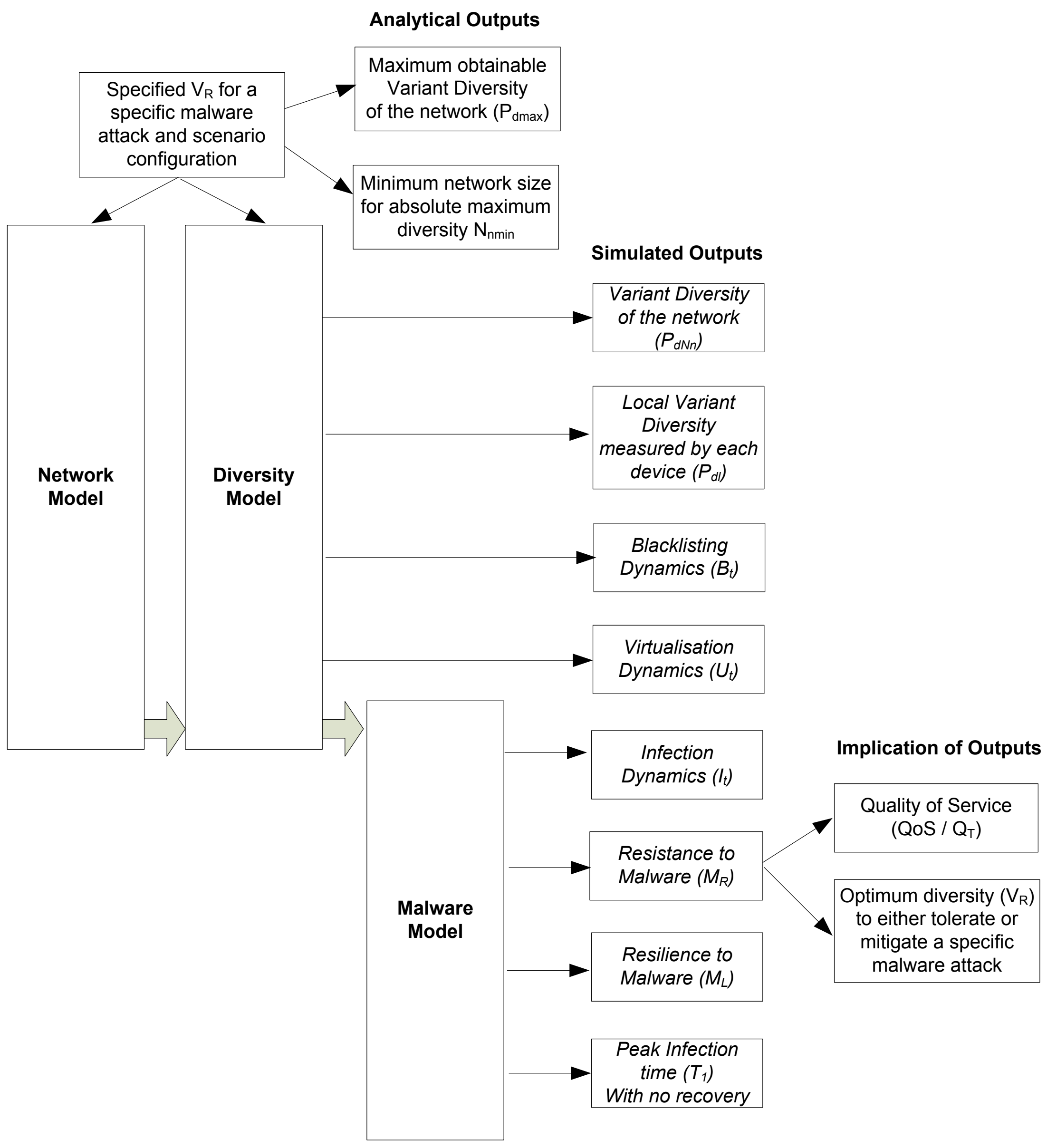

Analytical Outputs
Specified V_R for a specific malware attack and scenario configuration
Maximum obtainable Variant Diversity of the network (P_{dmax})
Minimum network size for absolute maximum diversity N_{nmin}
Simulated Outputs
Network Model
Diversity Model
Variant Diversity of the network (P_{dNn})
Local Variant Diversity measured by each device (P_{dl})
Blacklisting Dynamics (B_t)
Virtualisation Dynamics (U_t)
Malware Model
Infection Dynamics (I_t)
Resistance to Malware (M_R)
Resilience to Malware (M_L)
Peak Infection time (T_1) With no recovery
Implication of Outputs
Quality of Service (QoS / Q_T)
Optimum diversity (V_R) to either tolerate or mitigate a specific malware attack

Figure 7-13 - Outputs and optimum diversity for the agent-based model

7.8.1 Analytical Outputs

The two analytical outputs are defined as:

Maximum obtainable variant diversity (p_{dmax}) (§7.4.2): This calculated value is the maximum that can be achieved for the whole ad hoc ecosystem when there are no constraints on variant selection such as through user desirabilities, blacklisting, and software compatibility. Its calculation is defined in equation (7-2).

The minimum network size (minimum number of ad hoc devices) to achieve absolute maximum diversity (N_{nmin}) (§5.3.2.3): This is the minimum number of ad hoc devices necessary to utilise every possible genotype configuration for one run instance of a simulation and is calculated using equation (2-12).

7.8.2 Simulated Outputs

Variant Diversity of the network (P_{dN_n})*:* This is the measured instantaneous variant diversity from the simulation across all devices in the network calculated using the Nei diversity index equation (2-8).

Local Variant Diversity measured by each device (P_{dl})*:* This is the measured instantaneous variant diversity from the simulation across all local encounters stored in the genotype buffer calculated using the Nei diversity index equation (2-8).

Blacklisting Dynamics (B_t)*:* This is the instantaneously measured number of devices that are currently blacklisting variants.

Virtualisation Dynamics (U_t) *:* This is the instantaneously measured number of devices performing VM updates.

Infection Dynamics (I_t)*:* This is the instantaneously measured number of devices infected.

Resistance to Malware (M_R)*:* As per the epidemic model, this is the measured number of devices that do not become infected at the end of the epidemic. For the agent model, with or without recovery mechanisms, this equates to the number of devices in the S (or S′) state at the end of the epidemic (major outbreak) simulation since all devices initially start in the S (or S′) state.

$$\begin{aligned} M_R &= S(\infty) \qquad (7\text{-}5) \\ &= N_n - I(\infty), \quad no\ recovery \\ &= N_n - R(\infty), \quad with\ recovery \end{aligned}$$

Resilience to Malware (M_L)*:* This is the reciprocal of the measured time when the number recovered is within 1 of its final value.

Peak Infection Time $(T1)$ *(no recovery):* This is the time at which the number infected is within 1 of its final value.

7.8.3 Implication of Outputs

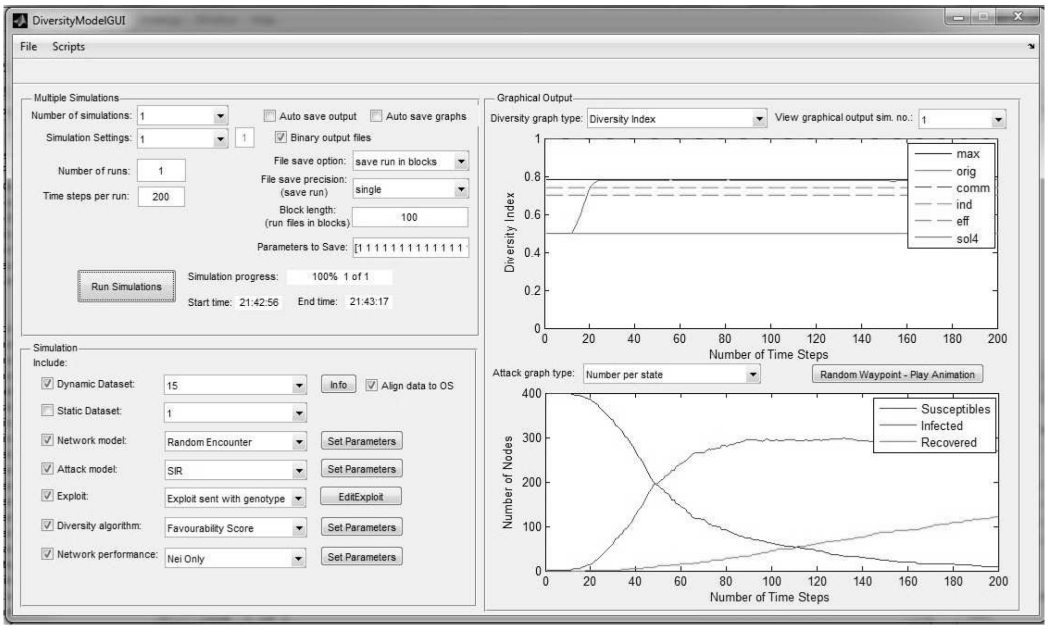
Optimisation of Diversity: For the agent model optimisation of diversity is measured through simulation across a parameter range to either tolerate or mitigate a specific attack. Unlike the epidemic model where optimisation can be calculated, for the agent model the point of desired tolerance, or mitigation is the measured point of the optimised diversity.

Quality of Service Tolerance (Q_T)*:* This is the measured resistance that will inform the Quality of Service for a particular scenario and malware type. When resistance is measured over a varying parameter, the value of the parameter at the required Q_T can be found.

7.9 Matlab Implementation

This section provides a brief overview of the Matlab implementation of the agent-based diverse system model.

The initial implementation of the agent model was created as a distributed architecture in a modular structure. A frame work has been developed with a user interface to allow further network models, diversity algorithms and attack models to be added. The model can be run from the user interface where settings files can be created, saved and run, multiple simulations can be run sequentially, and output files can be saved or loaded into the display. The model can also be run in *batch mode* where a sequence of tests is left to run in order, each providing input to the main GUI window, which is pictured in Figure 7-14.


DiversityModelGUI

Multiple Simulations
Number of simulations: 1
Simulation Settings: 1
Number of runs: 1
Time steps per run: 200
Auto save output ☐ Auto save graphs ☐
Binary output files ✓
File save option: save run in blocks
File save precision: (save run) single
Block length: (run files in blocks) 100
Parameters to Save: [1 1 1 1 1 1 1 1 1 1 1 1 1
Simulation progress: 100% 1 of 1
Start time: 21:42:56 End time: 21:43:17

Simulation
Include:
✓ Dynamic Dataset: 15 — Align data to OS ✓
☐ Static Dataset: 1
✓ Network model: Random Encounter
✓ Attack model: SIR
✓ Exploit: Exploit sent with genotype
✓ Diversity algorithm: Favourability Score
✓ Network performance: Nei Only

Graphical Output
Diversity graph type: Diversity Index
View graphical output sim. no.: 1
Attack graph type: Number per state

Figure 7-14 - Main GUI window of the Matlab implementation

A more abstract and efficient implementation was also created which captured the individuality of the devices whilst making use of the software tool in a single software program environment to utilise global parameters. For example the malware exploit data can be stored as a global variable and matched to genotypes individually, producing the same result as if the exploit data had been transmitted, received and stored individually. This implementation improved simulation speed by an order of magnitude and has been used as the basis for the results.

A flow chart describing the implementation flow of the software is given in Figure 7-15. Where relevant the individual components are referenced to their appropriate section or figure number describing the process in more depth. The source code for the Matlab implementation of the models can be found at the permanent link: http://wrap.warwick.ac.uk/98458.

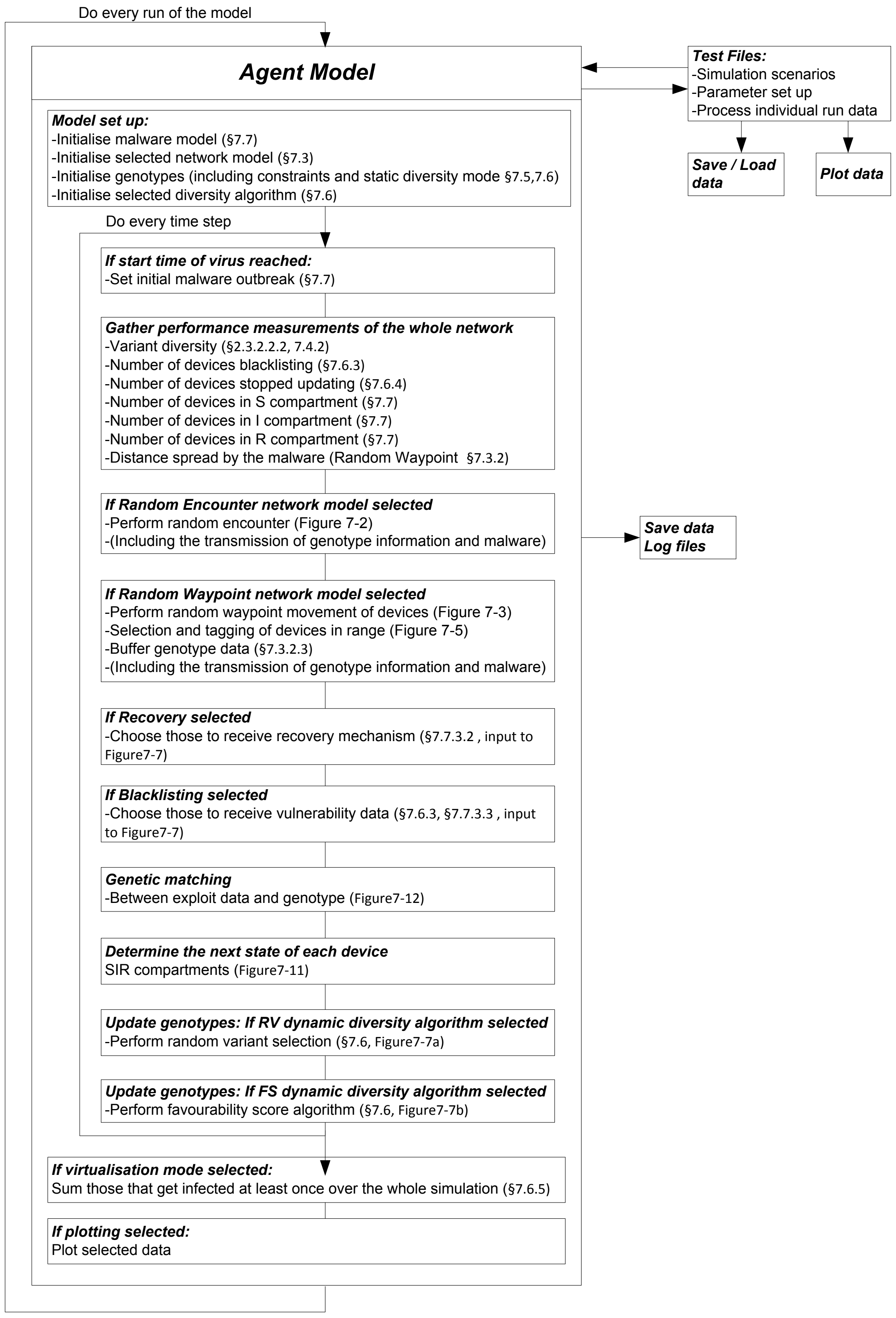

Do every run of the model
Agent Model
Test Files:
-Simulation scenarios
-Parameter set up
-Process individual run data
Save / Load data
Plot data
Model set up:
-Initialise malware model (§7.7)
-Initialise selected network model (§7.3)
-Initialise genotypes (including constraints and static diversity mode §7.5,7.6)
-Initialise selected diversity algorithm (§7.6)
Do every time step
If start time of virus reached:
-Set initial malware outbreak (§7.7)
Gather performance measurements of the whole network
-Variant diversity (§2.3.2.2.2, 7.4.2)
-Number of devices blacklisting (§7.6.3)
-Number of devices stopped updating (§7.6.4)
-Number of devices in S compartment (§7.7)
-Number of devices in I compartment (§7.7)
-Number of devices in R compartment (§7.7)
-Distance spread by the malware (Random Waypoint §7.3.2)
If Random Encounter network model selected
-Perform random encounter (Figure 7-2)
-(Including the transmission of genotype information and malware)
Save data
Log files
If Random Waypoint network model selected
-Perform random waypoint movement of devices (Figure 7-3)
-Selection and tagging of devices in range (Figure 7-5)
-Buffer genotype data (§7.3.2.3)
-(Including the transmission of genotype information and malware)
If Recovery selected
-Choose those to receive recovery mechanism (§7.7.3.2 , input to Figure7-7)
If Blacklisting selected
-Choose those to receive vulnerability data (§7.6.3, §7.7.3.3 , input to Figure7-7)
Genetic matching
-Between exploit data and genotype (Figure7-12)
Determine the next state of each device
SIR compartments (Figure7-11)
Update genotypes: If RV dynamic diversity algorithm selected
-Perform random variant selection (§7.6, Figure7-7a)
Update genotypes: If FS dynamic diversity algorithm selected
-Perform favourability score algorithm (§7.6, Figure7-7b)
If virtualisation mode selected:
Sum those that get infected at least once over the whole simulation (§7.6.5)
If plotting selected:
Plot selected data

Figure 7-15 - Matlab software implementation flow

7.10 Summary

This chapter describes an agent-based simulation approach to a diverse system model that, like the epidemic model, has been developed to incorporate software diversity and malware at the genetic level of an ad hoc network ecosystem. The chapter predominantly describes the design of the model and, unlike the epidemic model, the outputs of the agent model are analysed purely through simulation. The agent-based diverse system model is comprised of three components; a network model, a diversity model, and a malware model. It has been designed with greater flexibility than the epidemic based method with the inclusion of dynamic genotype configuration, device geographical location, and practical constraints. In the agent model, diversity does not remain as a stand-alone security strategy. The dynamic approach is exploited through the integration with other security mechanisms such as publically available vulnerability data and virtualisation technology to enhance its effectiveness. However, it is also capable of simulating the same conditions as the epidemic model to allow the comparison of results between diversity methods and for the comparison of results where the inclusion of additional mechanisms is made. Optimisation of diversity is measured through simulation across a parameter range to either tolerate or mitigate a specific attack.

The development of the agent model provides a simulation framework for incorporating biodiversity concepts and algorithms, different network models and malware models, and integrating them with other security mechanisms. In relation to the hypothesis, the framework provides a method for analysing how diversity can make ad hoc networks more resistant to cyber security attacks.

Chapter 8

Results and Analysis

8.1 Introduction

The purpose of this chapter is to demonstrate through mathematical modelling and simulation the hypothesis that *Incorporating biodiversity within peer-to-peer mobile wireless computer networks makes them more resistant to multi-exploit malware propagation*. This is achieved using the two models developed and detailed in chapter's 6 and 7. This chapter is split into two sections:

Constrained Diverse System Model: Epidemic Based - This section details the results of the epidemic based system model (§6). It firstly looks at the susceptibility relationship with diversity and malware types, since for the epidemic model, this can inform on the extent of the malware attack when there are no recovery mechanisms. Secondly it looks at the optimisation of diversity in order to predict how much is needed to either tolerate different malware in order to maintain Quality of Service, or mitigate malware when recovery mechanisms are present. Thirdly it looks at the resistance and resilience outputs of the ad hoc network ecosystem to show the relationship with diversity in relation to different types of malware attacks.

Diverse System Model: Agent-Based- This section details the results of the agent-based system model (§7). It firstly compares the agent model to the epidemic model under the same input conditions, specifically comparing the amount of static diversity needed to mitigate different types of malware. The purpose of which is to verify functionality of the agent model. Secondly the performances of the dynamic algorithms are assessed in reaching the maximum diversity level, both under ideal conditions, and when generic constraints are applied. The aim is to compare random with intelligent decisions, and compare the maximum achievable diversity with the actual diversity achieved. Thirdly the different modes of operation are analysed using the random encounter network model, including dynamic diversity as a standalone approach, and then with the additional security mechanisms in order to assess their beneficial impact. The effect of constraints on resisting the different malware types are considered including spatially constraints which are analysed using the Random Waypoint (RWP) network model. Unless explicitly stated, results are given for the one locus malware and the four locus malware configurations which cover the two extreme cases over the range considered. The results compare, for example a 4 exploit cross layer malware such as Stuxnet in the AND scenario (but in an ad hoc network), with the equivalent OR scenario and with malware targeting a single locus (which by its definition, additionally compares other epidemic malware models which consider malware and devices as single entities (§6.3.2.1). As previously stated susceptibility relationships are important for the static diversity case because they directly impact the magnitude of the malware attack. Here the implication of other loci configurations are also discussed (§8.2.1).

8.2 Constrained Diverse System Model: Epidemic Based

8.2.1 Susceptibility Relationships for Static Diversity

Devices that are susceptible to a specific instance of malware are at risk from being attacked. When diversity remains static for a period of time, so too does the susceptibility, allowing epidemic based models to predict the extent of the malware attack. The question is whether *increasing* static diversity can reduce the susceptibility of the ad hoc ecosystem, and lower the security risk.

The answer to this question depends upon several factors including the type of malware, the number of exploits and the variants they target, the initial diversity of the ad hoc ecosystem, and how diversity is increased (§5.3.2.2).

8.2.1.1 The One Locus Model with Increasing Variant Diversity

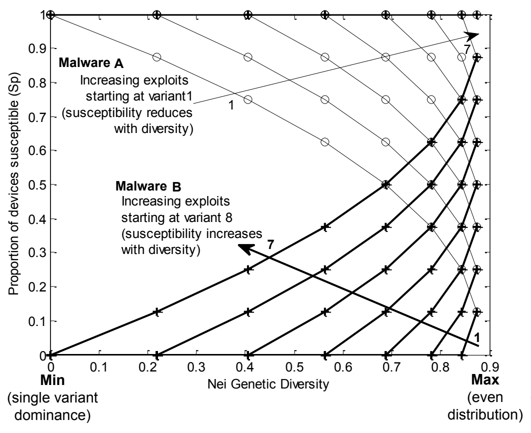
When the initial diversity is realised from an unequal distribution of software variants there are many solutions for increasing diversity from a minimum (single variant dominance) to a maximum (evenly distributed variants). Figure 8-1 gives two examples for a single locus, equivalent to the *one locus model*, with a maximum of eight possible variants being used (v1 to v8). For both examples, at minimum variant diversity ('min' column) only one variant is utilised with the highest possible frequency. At maximum variant diversity ('max' column) all eight variants are evenly distributed with identical frequencies. The two examples differ in that the first has one dominating variant that becomes less dominant as diversity is increased, by use of the other variants in a minimal way, whilst the second example maintains an even distribution of variants as more are utilised.

		min							***max***
	variants		**Frequency of variant**						
malware A -->	v1	1	7/8	6/8	5/8	4/8	3/8	2/8	1/8
	v2	0	1/8	1/8	1/8	1/8	1/8	1/8	1/8
More exploits	v3	0	0	1/8	1/8	1/8	1/8	1/8	1/8
	v4	0	0	0	1/8	1/8	1/8	1/8	1/8
	v5	0	0	0	0	1/8	1/8	1/8	1/8
More exploits	v6	0	0	0	0	0	1/8	1/8	1/8
	v7	0	0	0	0	0	0	1/8	1/8
malware B -->	v8	0	0	0	0	0	0	0	1/8

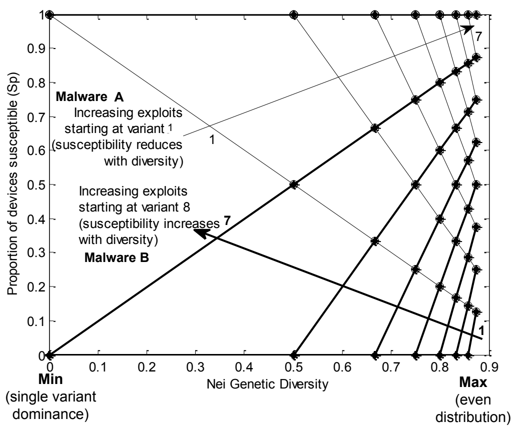
a) Example 1 - Single dominant

		min							***max***
	variants		**Frequency of variant**						
A -->	v1	1	1/2	1/3	1/4	1/5	1/6	1/7	1/8
	v2	0	1/2	1/3	1/4	1/5	1/6	1/7	1/8
More exploits	v3	0	0	1/3	1/4	1/5	1/6	1/7	1/8
	v4	0	0	0	1/4	1/5	1/6	1/7	1/8
	v5	0	0	0	0	1/5	1/6	1/7	1/8
More exploits	v6	0	0	0	0	0	1/6	1/7	1/8
	v7	0	0	0	0	0	0	1/7	1/8
B -->	v8	0	0	0	0	0	0	0	1/8

b) Example 2 - Even distribution



c) Example 1 – Single dominant



d) Example 2 – Even distribution

Figure 8-1 – Susceptibility relationship with increasing variant diversity

For the one locus model, the AND and OR malware types are identical and therefore it is the number of exploits that characterise the malware. In both examples, when a malware instance, labelled as *A,* with one exploit, targets variant 1 (v1) (Figure 8-1 (a) and (b)), the resulting susceptibility (Figure 8-1 (c) and (d)), *reduces* as variant diversity is increased as measured by Nei Genetic Diversity. If instead a second malware instance labelled as B, also with a single exploit, targets variant 8 (v8) the resulting susceptibility *increases,* at the point where the variant becomes utilised as diversity is increased. As more exploits are added to each malware instance, more variants become susceptible (e.g. malware A with 2 exploits targets variant v1 and v2, and malware B with 2 exploits targets variant v8 and v7, and so on). Increasing variant diversity can therefore either reduce susceptibility, as in the case of malware *A*, or increase susceptibility, as in the case of malware *B*. This result has been seen within natural systems and is explained by the *dilution effect* (§2.2.2.2.3) where increasing diversity *dilutes* the proportion of susceptible variants, and the *amplification effect* whereby increasing diversity increases the representation of susceptible variants. This result is intuitively expected in a static diversity scenario and confirms that exploits targeting higher frequency variants result in a more susceptible ad hoc ecosystem, and pose a higher security risk. The modelling of non-maximally diverse scenarios is not considered by other malware models, but it can allow the effects from the diversity of current networks to be analysed in response to different malware types, assuming the necessary data can be collected for each analysed layer of the software stack (§9.3).

8.2.1.2 Multiple Loci at Absolute Maximum Diversity

When a fixed number of software variants are evenly distributed, susceptibility is no longer dependent on which variant the malware is targeting at a given locus: The malware could target any single variant and the susceptibility would be the same. The susceptibility is now dependent upon the malware type and the number of exploits targeting the variants at each locus. Two malware types have been defined, the logical AND, and the logical OR (§5.3.3). The difference in their susceptibility relationships with varying numbers of exploits (exploit richness) for a fixed variant richness of 8 is shown in Figure 8-2 (a) and (b) respectively. The relationships in these graphs show the condition of *absolute maximum diversity* (§5.3.2) where the *maximum number of unique genotypes* are utilised and equally distributed in addition to variants at each locus being equally distributed. The relationships follow the susceptibility equations for the logical AND and logical OR types (§6.3.2), and allow comparisons between multiple loci and the '*one locus model*' (single locus, §6.3.2.1).

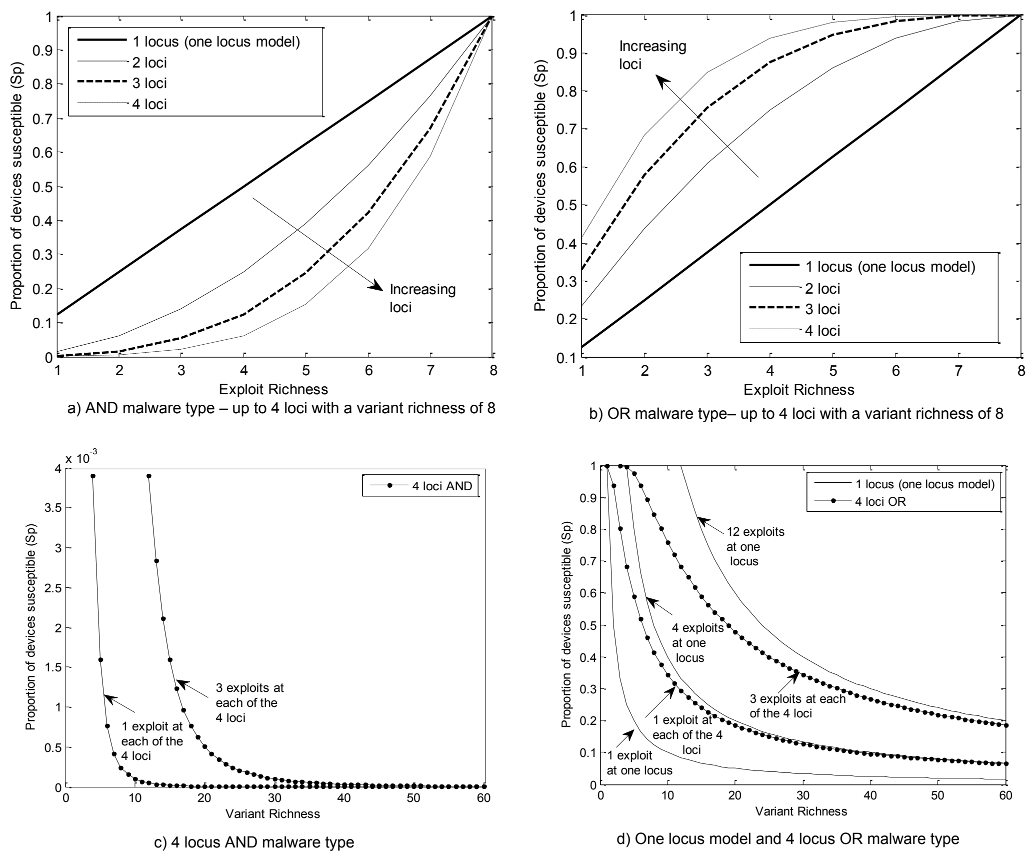


Figure 8-2 - Susceptibility relationships at absolute maximum diversity

For the one locus model the AND and OR relationships are identical where the susceptibility increases linearly with the number of exploits until there is an exploit for each of the 8 variants. This identical relationship is as expected and was derived by equations (6-12) and (6-15). For the AND malware type the susceptibility decreases with the number of loci, for a given exploit and variant richness. This is because of the nature of the AND relationship where increasing the number of loci means additional specific variants need to be present in a single genotype before it can be infected by malware, hence reducing the scope of those susceptible.

Understanding susceptibility relationships is important because it defines the magnitude of the malware attack when there are no recovery mechanisms in place. Particularly when considering homogeneous mixing models where devices are assumed to make contact with each other at random which forms an underlying assumption of the mathematical epidemic model (§4.2.5). When there are recovery mechanisms the reproduction number (§4.3.2.5) additionally contributes to the extent of the malware attack.

For a fixed exploit and variant richness (as shown in Figure 8-2 (a)) increasing the number of loci in which the AND malware type targets not only reduces the susceptibility but the reduction in susceptibility also becomes less. This means that in practice it is beneficial from a security perspective to 'encourage' malware to 'have' to use multiple exploits across layers to infect and propagate, for example designing loci divisions that make it difficult for malware to spread using only one exploit. Although, the benefit of malware using an increasing number of cross layer exploits diminishes. For the OR malware type the susceptibility increases with the number of loci, for a given exploit and variant richness. This is because the more exploits there are available, which increase with the number of loci in this example, the greater number of genotypes there are that will be susceptible, making this type of malware a high security risk. However, opposite to the AND scenario, for a fixed exploit and variant richness (as shown in Figure 8-2 (b)) increasing the number of loci in which the OR malware type targets not only increases the susceptibility but this increase diminishes as more cross layer exploits are added.

However increasing diversity through *variant richness* for either of the malware types, under absolute maximum diversity conditions, reduces the susceptibility as shown in Figure 8-2 (c) and (d). The difference being in the gradients of their reduction. For the AND malware type, every increase in variant richness produces a large reduction in the proportion of susceptible devices. For an exploit richness of one the proportion of devices susceptible becomes very small by the time a variant richness of ten is reached. For the OR malware type the reduction is smaller, and when the variant richness is large relative to the *total* number of exploits, the susceptibility can be approximated by the one locus model with the same total number of exploits (§6.4.4.4.4). As shown by the curves, the equivalent one locus model (with the same total number of exploits) has either the same or a higher susceptibility, and therefore malware writers with OR capability wishing to inflict maximum damage regardless of the variant richness could better do so by targeting any single locus with multiple exploits (see Figure 5-6 Malware threat model) rather than spreading the exploits across loci.

8.2.2 Optimum Diversity to Tolerate or Mitigate a Malware Attack

8.2.2.1 Optimisation and Simulation Process

This section considers how much diversity is needed to tolerate or mitigate a specific type of malware attack (§6.4.4). When tolerance (Q_T) is required to maintain a specific QoS for the ad hoc network ecosystem, or mitigation is required to prevent a major outbreak (reproduction number $R_0 < 1$), the diversity optimisation process follows that pictured in Figure 8-3. Firstly the specified constraint is used to calculate the necessary susceptibility (S_p) using equations (6-21), (6-23), and (6-28) defined in section 6.4.4. The calculated S_p is then used to determine the diversity in terms of the optimum number of variants needed per locus to tolerate or mitigate a specific malware attack (§6.4.4.4) using equations (6-32),(6-33) for the AND malware and (A-7),(A-8) for the OR malware. To simulate and verify the calculated predictions using a malware model, the S_p value is recalculated using the discretised optimum number of variants. This is used to calculate susceptible infection rate (β_s) and the number of susceptible devices (N_s) for input in to the malware model. Running the deterministic or stochastic malware model (§4.3.1, §6.4.2) can be used to measure peak infection times, resistance or resilience, for either optimum diversity conditions, or other specified diversity levels for a specific malware attack.

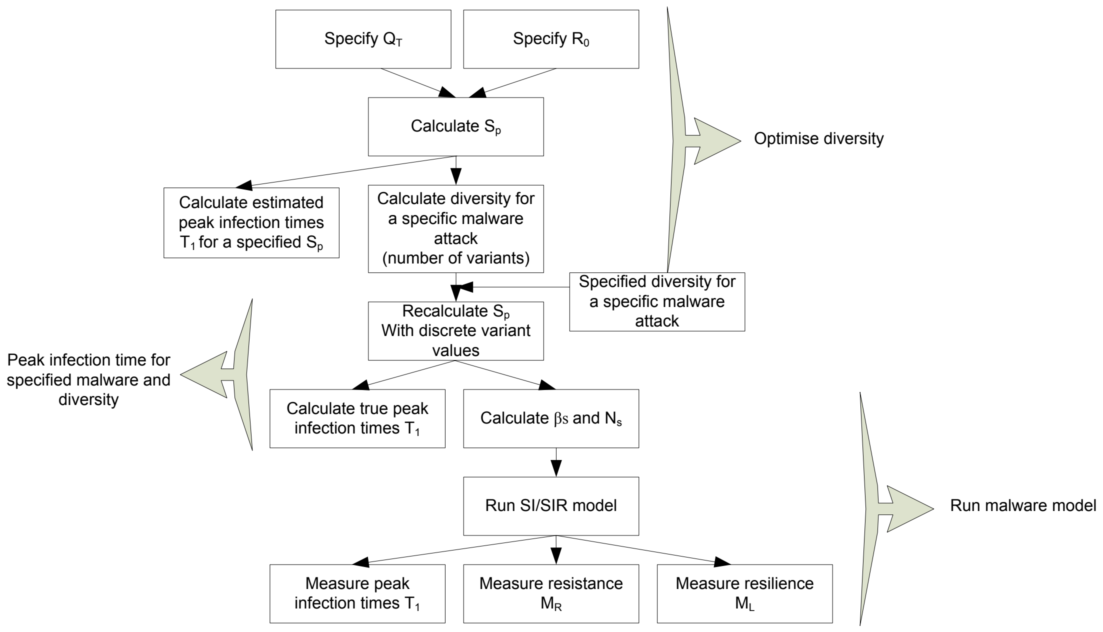

Specify Q_T
Specify R_0
Calculate S_p
Optimise diversity
Calculate estimated peak infection times T_1 for a specified S_p
Calculate diversity for a specific malware attack (number of variants)
Specified diversity for a specific malware attack
Recalculate S_p With discrete variant values
Peak infection time for specified malware and diversity
Calculate true peak infection times T_1
Calculate βs and N_s
Run SI/SIR model
Run malware model
Measure peak infection times T_1
Measure resistance M_R
Measure resilience M_L

Figure 8-3 - Process for optimising diversity and simulating the malware model

8.2.2.2 Tolerance of a Specific Attack with no Recovery (SI)

When there are no recovery mechanisms, tolerance of a security attack is critical in maintaining QoS. It is presumed that high QoS levels are required to maintain adequate functioning of an ad hoc ecosystem. However 100% tolerance is likely to be difficult to achieve in practice since the susceptibility asymptotically approaches zero as the variant richness increases producing diminishing benefits. A tolerance below this may be sufficient to maintain an adequate QoS for the network. QoS Tolerances of 80%, 90% and 95% are used to compare and determine the required diversity in terms of the optimum number of software gene variants at each of the four loci to tolerate a specific attack. Capturing the exact tolerance requirement for real networks is difficult in the absence of data, or event based simulators which have the ability to model lower levels of abstraction such as traffic generation and communication protocols (see §9.3.1.2).

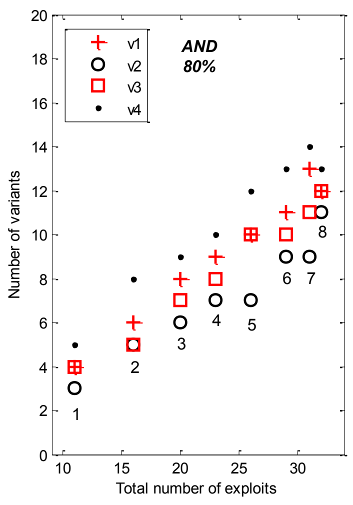
8.2.2.2.1 Optimum Diversity (General)

In the first instance the general equations are used (§6.4.4.4.1, §6.4.4.4.4) to calculate the exact number of variants necessary to tolerate a specific number of exploits targeting each of the four loci in the modelled software stack (§5.3.1.3, Figure 5-5). A range of exploits are demonstrated for both the AND and OR malware types up to a maximum of eight per locus. Table 8-1 shows the exact number of exploits used and Figure 8-4 shows the calculated results.

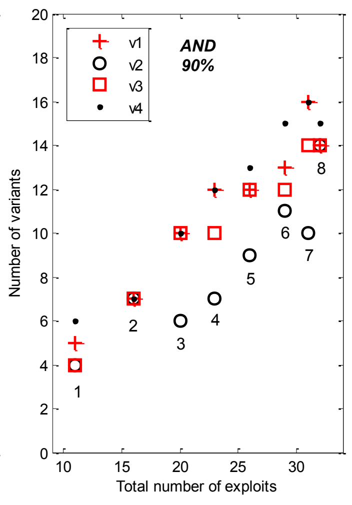
Table 8-1 – Specific AND and OR malware examples

Malware Number AND / OR	Number of exploits (*x*) at each locus			
	x1	*x2*	*x3*	*x4*
1	2	3	2	4
2	4	3	5	4
3	5	4	6	5
4	6	4	6	7
5	6	5	8	7
6	8	5	8	8
7	8	8	7	8
8	8	8	8	8

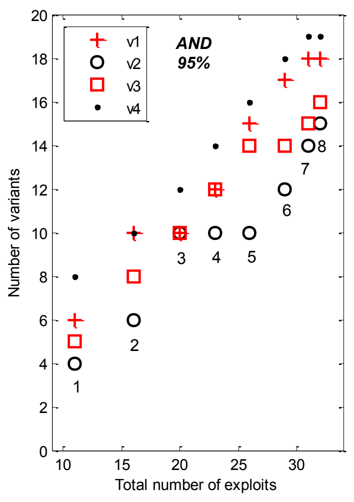
The optimum number of variants for the four locus AND and OR malware types are computed using the general equation (6-32) and the general approximation (because a high tolerance equates to a low S_p) equation (6-39) respectively, together with the minimisation equation (6-29) (§6.4.4.4) where the first computed optimum solution is used as the final result. Optimum diversity results for the AND and OR malware types for the three selected tolerance levels are given in Figure 8-4. To show the results graphically the total number of exploits used by each malware is on the x axis of the graphs. The required number of variants for each locus (v1,v2,v3,v4) to tolerate each malware is shown in each vertical set on the graphs as numbered. For the AND malware type fewer than 20 variants are needed per locus to maintain a QoS Tolerance between 80% and 95% for up to eight exploits at each locus. For the OR malware type however, up to 160 variants are needed for an 80% tolerance rising to over 600 to maintain a 95% tolerance.



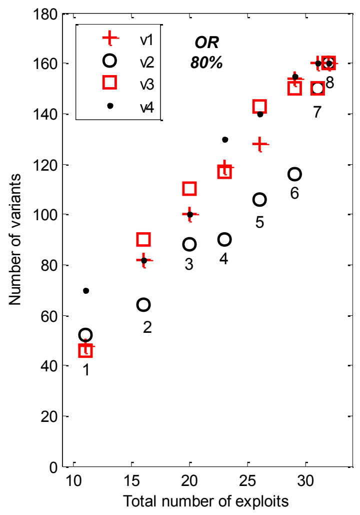
a) 80% Tolerance – 4 locus AND



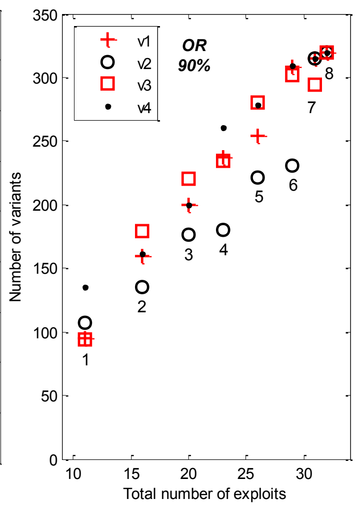
b) 90% Tolerance – 4 locus AND



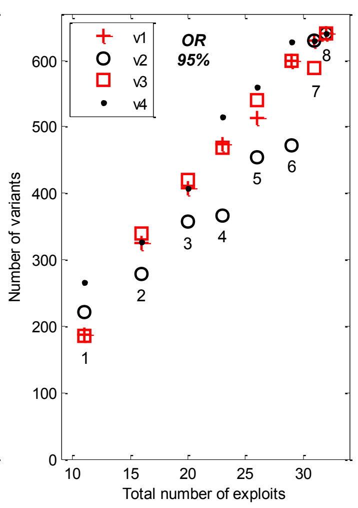
c) 95% Tolerance – 4 locus AND



d) 80% Tolerance – 4 locus OR



e) 90% Tolerance – 4 locus OR



f) 95% Tolerance – 4 locus OR

Figure 8-4 - Optimum diversity (variants at each locus) to tolerate an attack

The advantage of the general method is that there can be several optimum solutions from which to choose. This could be particularly useful for the AND malware type where there are relations across loci, even for a large number of variants. For example if there are an abundance of variants available at one locus, but the variants are restricted at another, the QoS could be maintained by allocating an alternative valid solution, aiming to comply with the variant restrictions. One realistic example could be a limited number of available core OS library variants balanced by having a greater number of application service variants (§3.2.1.3).

8.2.2.2.2 Optimum Diversity (Average)

In situations when the exact security risk at each locus is not clear, an average number of exploits (exploit richness) can instead be specified resulting in a diversity optimisation of an average number of variants (variant richness). Here there is only one diversity solution and no minimisation is required. The one locus model is compared to the four locus genotype model with the AND malware type using the variant richness equation (6-33), and the OR malware type using the approximation equation (6-40). The results are shown in Figure 8-5 for the three specific Q_T values. Results for two and three loci are given in Appendix B.

The variant richness follows a linear relationship with the exploit richness. The non-exact linear relationship for the four locus AND case (Figure 8-5 (c)) is due to rounding when calculating an exact integer number of variants. For the AND malware type, as more exploits are added *across* loci (e.g across four loci compared to one locus) to propagate the malware and act out its malicious intent, the less variant richness is required to achieve the same QoS Tolerance. However the resultant number of unique genotypes is very large as shown in Figure 8-5 (d) and hence the minimum network size to achieve the required tolerance, under maximum diversity conditions is also very large. For smaller network sizes the number of genotypes present will equal the number of devices so that not all of those possible genotypes will be utilised. This may alter the susceptibility and hence the QoS. For the OR malware type (Figure 8-5 (b)), more variant richness is required to achieve the same QoS tolerance, however under approximation conditions (§6.4.4.4.4), the variant richness can also be used to define the minimum network size which is considerably less than for the AND type. For the one locus model, the resultant number of genotypes is the same as the variant richness.

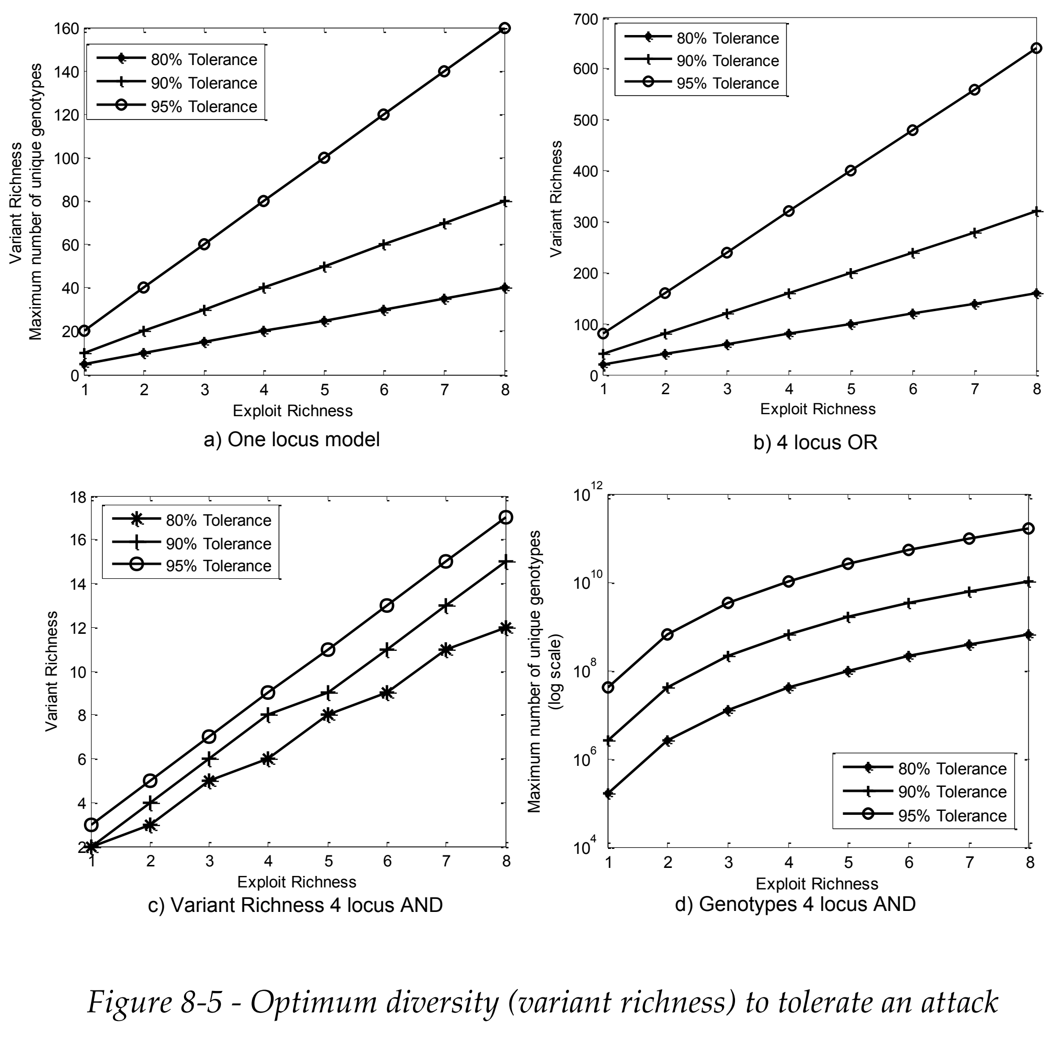


Figure 8-5 - Optimum diversity (variant richness) to tolerate an attack

An example of a four locus – four exploit AND malware type is the Stuxnet worm (§3.3.4). The worm caused the disruption of a nuclear facility, but closely related versions, were later found propagating elsewhere [17]. To tolerate such a worm at a network level where 95% of QoS is maintained, only three variants at each of the four loci would have been needed. For an equivalent OR malware type, such as those generated using exploit kits (§3.3) 80 variants would be needed.

8.2.2.3 Peak Infection Time with No Recovery (SI)

For a fixed number of devices N_n and contact rate c_n $(where\, \tau = 1)$ the peak infection time T_1 can either be estimated by calculation (for the deterministic model §6.4.5.1) for a *specified QoS Tolerance* Q_T (results given in §8.2.2.3.1) or calculated from a *specific malware attack with a specified* Q_T (results given in §8.2.2.3.2) following the process in Figure 8-3. In the first calculation no knowledge of the specific malware attack or diversity is used. For a specified Q_T, equation (6-21) is used to calculate S_p and then substituted in equation (6-42) to calculate the peak time T_1. For a more accurate result diversity is firstly optimised for a specific malware attack to account for the discrete values needed for the number of variants at each locus (§6.4.4). This information is then used to recalculate the susceptibility S_p for a specific attack where the true peak infection times, using equation (6-42), can be calculated.

8.2.2.3.1 Specified QoS Tolerance

Figure 8-6 (a) shows the calculated time of peak infection for a specified QoS Tolerance (80%, 90% and 95%), whilst Figure 8-6 (b) shows the simulated time (measured from simulation) for both the deterministic and stochastic SI models. The total network size was made large (20,000) so that even after reduction to those susceptible, the simulated network was still considered large. The peak infection time parameters are given in Table 8-2. The stochastic simulation was averaged over 500 runs, with the standard deviation bar as shown. As predicted, the simulated deterministic result agrees with the calculated deterministic result. The difference between the deterministic and stochastic curves is known as the stochastic lag [228]. The higher the specified QoS tolerance the fewer the number of devices that become infected overall. Also the time to reach the peak of infection takes longer giving more time available to react to the malware if intervention mechanisms such as detection or recovery are present.

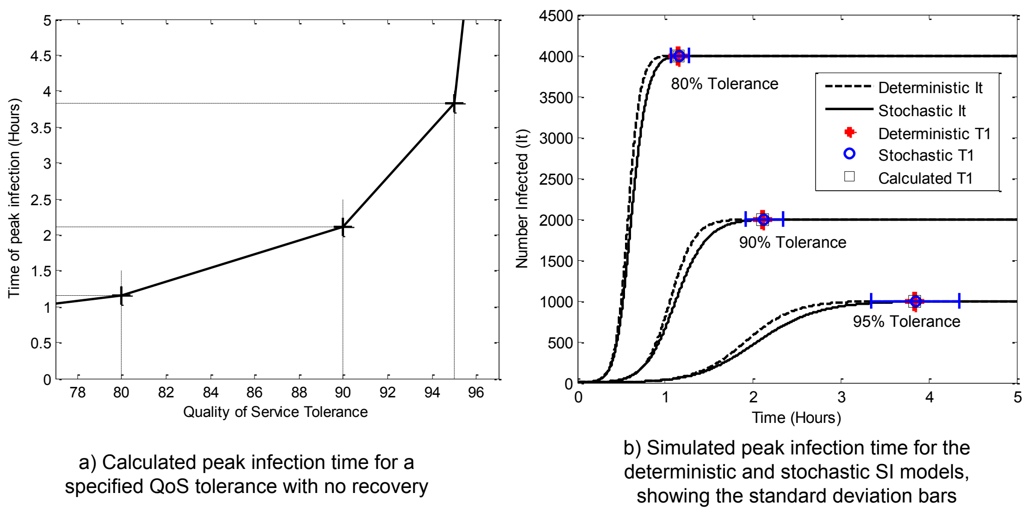


Figure 8-6 - Calculated and simulated peak infection times

Table 8-2 – Peak infection time parameters

Tolerance	Sp	Nn	Ns	βn	βs
80%	0.2	20000	4000	0.02	0.004
90%	0.1	20000	2000	0.02	0.002
95%	0.05	20000	1000	0.02	0.001

8.2.2.3.2 *Specific Malware Attack with QoS Tolerance*

Figure 8-7 show the corresponding calculated peak infection times for the optimum variant richness given in Figure 8-5 for the three specified Q_T values. For the one locus model and the OR type, the peak infection times in this scenario are the same as those calculated and shown in Figure 8-6 (a) since the calculation resulted in an integer number of variants for the Q_T values specified, although this is not generally the case for all Q_T values. For the AND malware type there is a difference due to both the power terms in the average variant richness equation (6-33) and integer rounding. As an example, for the four locus AND type with one exploit per locus, the true peak infection time is calculated to be over twelve hours for 95% tolerance instead of just under four hours as previously estimated from calculation, and thereby lengthening the time window of performance degradation and hence increasing the reaction time for intervention. Results for two and three loci are given in Appendix B and show similar differences in the peak infection times for the AND case.

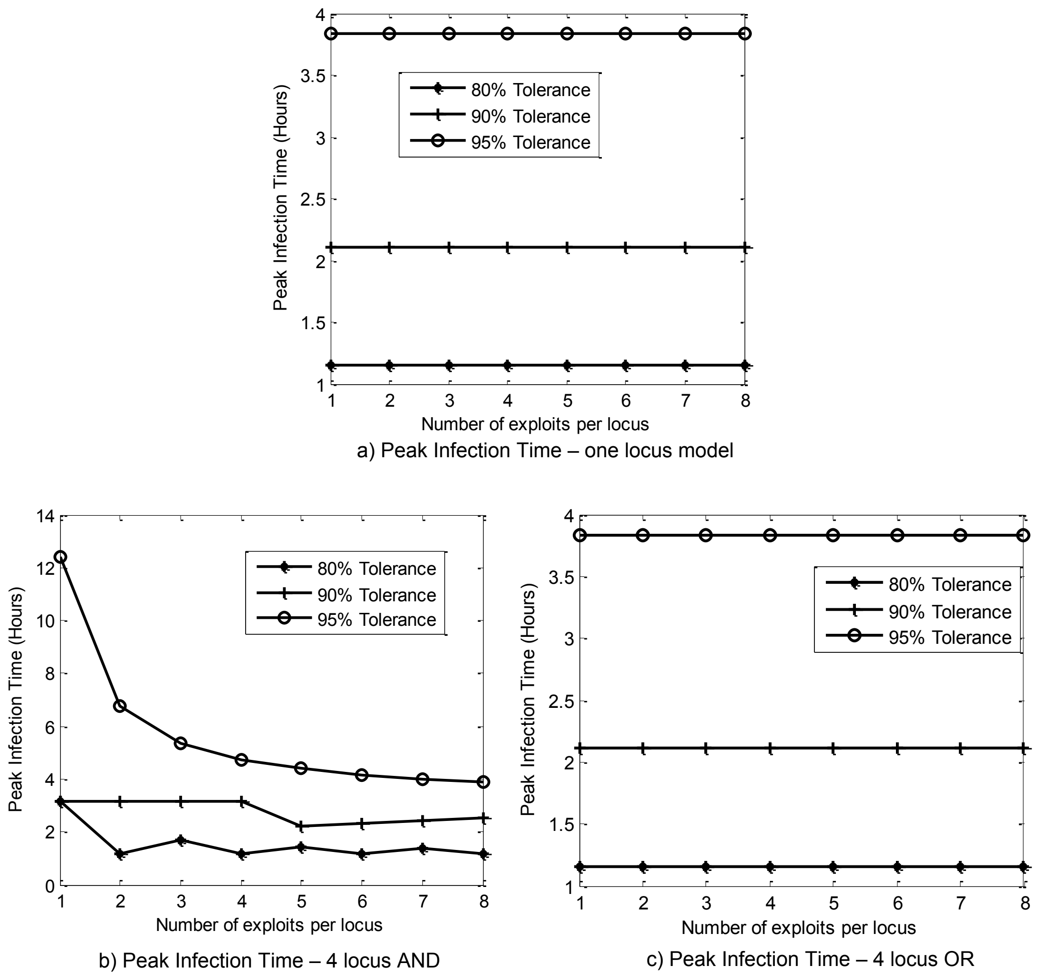


Figure 8-7 - Calculated peak infection times with no recovery

8.2.2.4 Tolerance of a Specific Attack with Recovery (SIR)

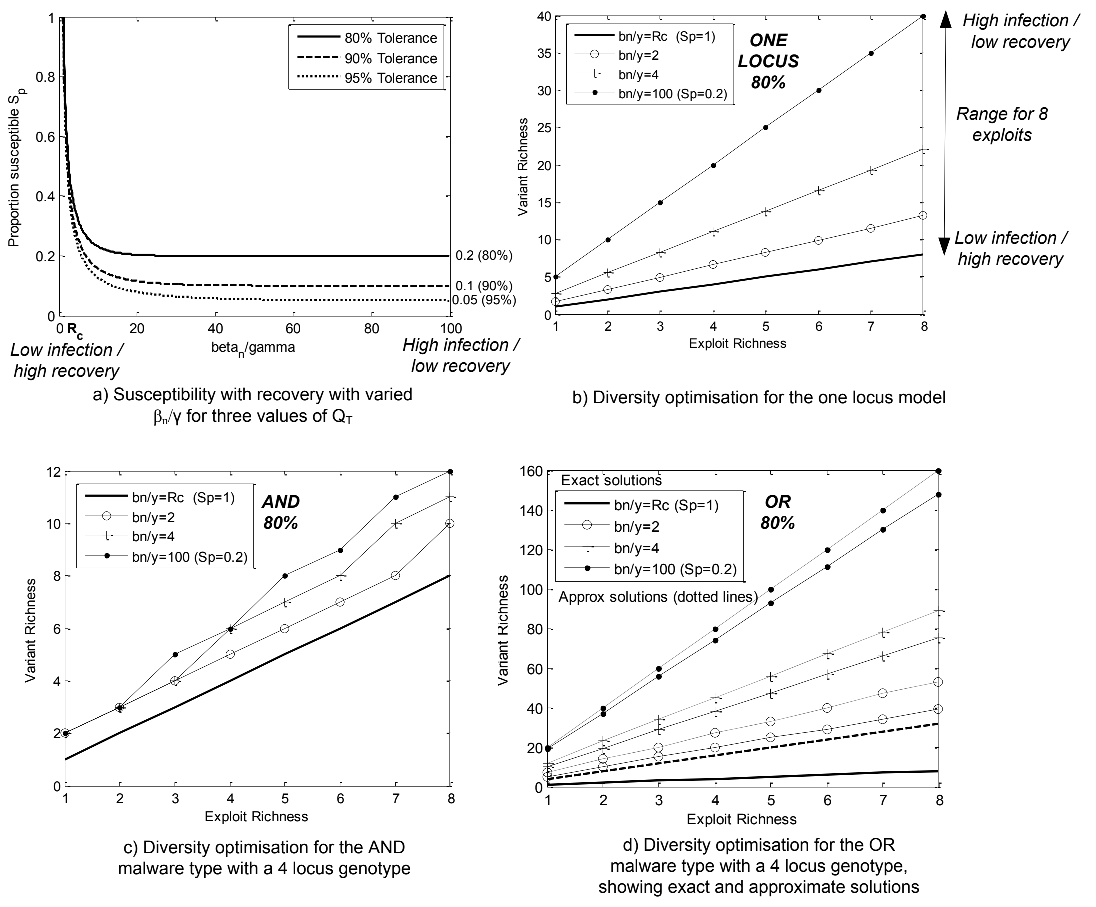
When recovery mechanisms are introduced at a rate of γ, rearrangement of the final size equation for the SIR model defines the susceptibility S_p for a specified QoS tolerance (Q_T) (Equation (6-23)) (§6.4.4.2). Figure 8-8 (a) shows the relationship between S_p and β_n/γ, where the ratio varies between R_c and a large number (100) for the three fixed values of Q_T. The large number represents the case where the infection is so high, or the recovery is so low that it becomes representative of the SI model with no recovery. In this case S_p levels out to a value corresponding to the specified Q_T. The three critical values of R_c are calculated and shown in Table 8-3 using the Bound C equation (6-26).

Table 8-3 – Critical R_c values for a given Q_T

Q_T	R_C
0.8 (80%)	1.1157
0.9 (90%)	1.0536
0.95(95%)	1.0259

Small β_n/γ, values indicate either a low infection rate or a high recovery rate. Under this condition, a high proportion of devices can become susceptible whilst the QoS is still maintained.

Optimisation of diversity for a fixed Q_T of 0.8 (80%) with recovery is shown for the one locus model (Figure 8-8 (b)), the four locus AND malware type (Figure 8-8(c)), and the four locus OR malware type (Figure 8-8 (d)). Malware types with up to eight exploits at each locus are calculated. Four fixed values of the β_n/γ, ratio are used between the upper and lower bounds, where $S_p = 1 - Q_T$ (bound A) and $S_p = 1$ (bound B) (§6.4.4.2). For the one locus case as shown in Figure 8-8 (b), the required variant richness (V_R) varies between 8 and 40 for 8 exploits. The required upper limit of 40 equates to the result when there is a very low recovery rate or a high infection rate since this is representative of when $S_p = 0.2$ for the non recovery scenario. The lower limit of 8 equates to the result where $S_p = 1$ where all devices are susceptible and so the number of variants equates to the minimum number imposed to accommodate 8 exploits. This equates to the result when there is a high recovery rate or a low infection rate.



a) Susceptibility with recovery with varied β_n/γ for three values of Q_T

b) Diversity optimisation for the one locus model

c) Diversity optimisation for the AND malware type with a 4 locus genotype

d) Diversity optimisation for the OR malware type with a 4 locus genotype, showing exact and approximate solutions

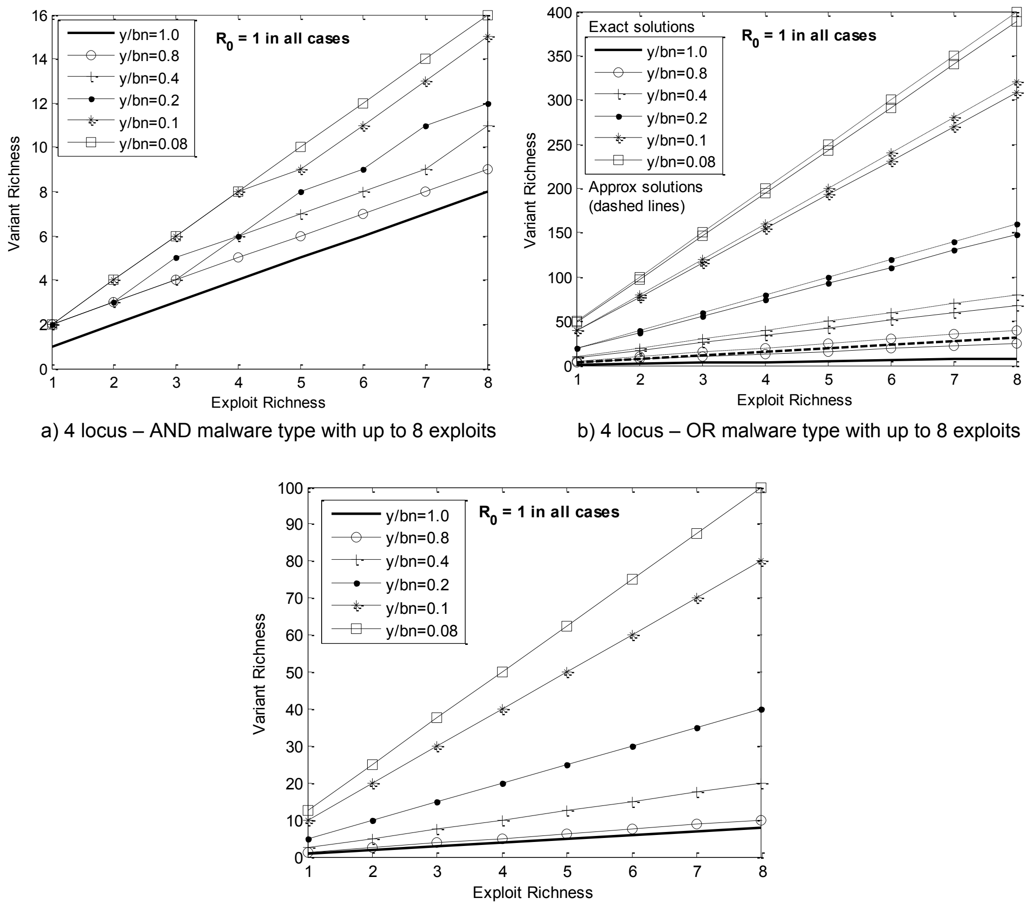
Figure 8-8 - Optimum diversity to tolerate an attack with recovery.

A similar relationship can be seen for the four locus AND shown in Figure 8-8 (c), except the variant richness range is smaller, and varies between 8 and 12 for 8 exploits. For the four locus OR case shown in Figure 8-8 (d), the variant richness range is bigger and varies between 8 and 148 for the exact solution, and up to 160 for the approximate solution. The upper limit of the approximate solution equates to the non recovery scenario. As shown in the graph using the OR approximation for high S_p values does not give an accurate result and so the full exact equation should be used. High rates of recovery relative to the rate of infection can have a large impact on the number of variants required for the OR malware type at each locus to maintain an acceptable QoS. There is therefore a trade off between maintaining QoS, the speed of recovery, and the variant richness.

The ability to recover quickly through patching in a system with diversity will depend upon the way in which variants are generated and the way in which variants are stored, and maintained. When variants are generated by COTS software the total number of different patches required to maintain them will be higher than a non-diverse system as there will be more underlying software. When comparing patching against malware only however, the same number of different fixes will be required in a non-diverse system as a diverse system to recover from the same number of different malware. Even if the malware is comprised of multiple exploits, each of the targeted vulnerabilities would need to be fixed in the software in both a non-diverse and diverse system. In practice however for a non-diverse system multiple fixes may be combined into a single patch, but would need to be distributed to all devices. For a statically diverse system a smaller number of devices would require patching. However the fixes may need to be spread over several individual patches to be applied in parallel to different variants at different loci. For the AND malware type, recovery would begin the moment the first patch is applied due to the dependent relationship across loci. In a dynamically diverse system the distribution of patches will depend upon how variants are stored. If variants are stored in a globally accessible pool, patches would be applied to the pool so that when they are next downloaded the patched versions are retrieved. However if variants are stored locally, a single patch would need to be disseminated to all devices with the vulnerable variant. When variants are generated using automated techniques, the amount of patching may depend upon the source of the vulnerability. If an exploit attacked a single variant where the vulnerability source was in the software code then all associated variants may need to be recompiled and patched to avoid similar exploits attacking the same vulnerability in different variants. If the vulnerability is associated with a single variant, only that would need to be patched or replaced.

8.2.2.5 Mitigation of an Attack with Recovery (SIR)

To prevent a major outbreak of a specific malware type the reproduction number R_0 must be less than 1, where, using equation (6-28) S_p must be less than the ratio γ/β_n. Figure 8-9 shows the relationship in terms of variant richness necessary to mitigate an attack for a range of γ/β_n ratios with varying exploit richness. The one locus model is compared to the four locus genotype for both the AND and OR malware types. For example, for a γ/β_n ratio of 0.2, the one locus model would require 20 software variants to mitigate a malware attack with 4 exploits. A four locus AND malware attack with 4 exploits, each targeting a different locus (exploit richness of one) would required a variant richness of 2 to mitigate the attack. A four locus OR malware attack with the same exploit richness would require 19 variants.



a) 4 locus – AND malware type with up to 8 exploits

b) 4 locus – OR malware type with up to 8 exploits

c) 1 locus model with up to 8 exploits

Figure 8-9 - Optimum diversity to mitigate an attack with recovery

Practical recovery times in a network can vary widely. Antivirus response times can range from around two hours upwards [3], and full software patching can take several months [14]. With a minimum recovery time of two hours $\left(\gamma = \frac{1}{2\times60\times60}\right)$ the necessary diversity in terms of variant richness to mitigate an attack is calculated where the average time between contacts is varied between 1 second $(c_n = 1)$ and 10 minutes $\left(c_n = \frac{1}{10\times60}\right)$. The result for malware with an exploit richness of 1 is given in Figure 8-10. Experimental data suggests that malware propagated over wireless communication protocols such as Bluetooth could take in the order of 10 seconds, with most of this time spent setting up the communication link [263] [3]. In a densely populated area where there may be many ad hoc devices, the average time between successful contacts may therefore be around 10 seconds. With this contact rate, the four locus AND malware type would only require a variant richness of 6 to mitigate the attack, the one locus model would require 720 variants and the four locus OR malware type would require a very large variant richness of 2,879 before preventing the spread.

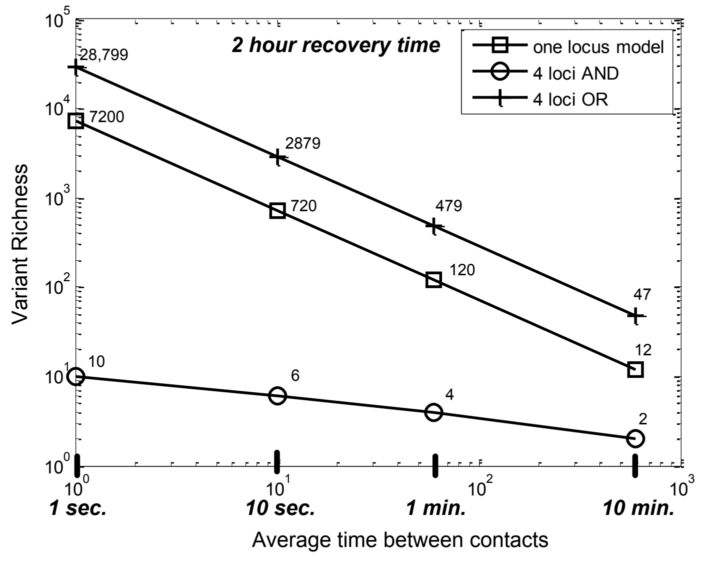


Figure 8-10 – Malware mitigation of a practical scenario

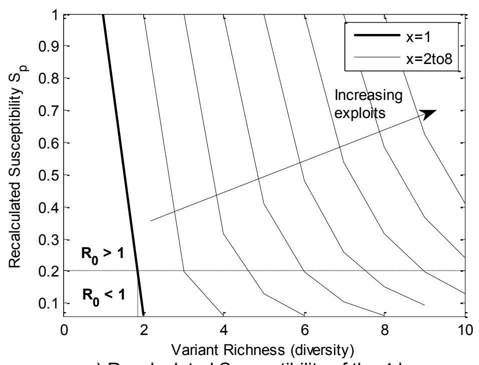
This practical scenario re-iterates the danger of the OR malware type. Within a designed software stack genotype it is assumed that the granularity and the functionality is carefully defined in relation to potential malware (§5.3.1.3). Defining a single locus as a complete operating system (including services, libraries and kernel) would be a very course coarse grained approach with the possibility that diversity applied in this way would be insufficient to reduce the susceptibility to that of a potential AND malware type. It would be beneficial to apply granularity and diversity of variants to 'encourage' malware to be represented as being of the

AND type leading it to span true 'multi-loci' since the generated genotype configurations (assuming they encompass the separate locations at which the malware targets) will reduce the susceptibility (§8.2.1.2). Equally there needs to be consideration of the OR malware type which increases susceptibility as the number of exploits across loci increases (§8.2.1.2). If the granularity and functionality definition of the software stack genotypes is ill-considered for potential OR cases, it may not improve the security benefit against these types of malware.

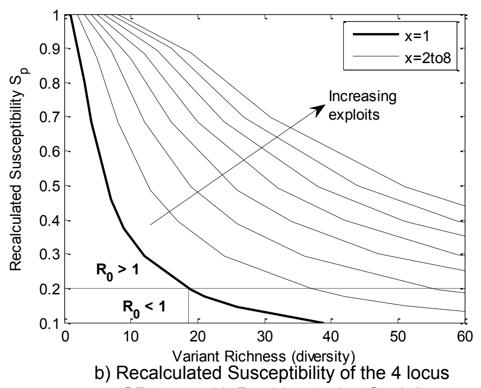
8.2.2.6 Simulated Resistance and Resilience to Mitigate an Attack

The resistance and resilience outputs of the ad hoc network ecosystem are used to show their relationship with diversity in a constrained diverse system model. The four locus AND and OR malware types are simulated using the deterministic malware model (stochastic results are shown in §8.3.1.2) to measure the resistance and resilience properties for mitigating a specific attack. A fixed γ/β_n ratio of 0.2 is used to show the relationship. For example, from inspection of Figure 8-9 (a), mitigation of the four locus AND type, with an exploit richness of one should occur at a variant richness of two. S_p values are calculated for a specified diversity and malware attack using the process pictured in Figure 8-3. The diversity is varied over a range to include the calculated optimum diversity points given by the $\gamma/\beta_n = 0.2$ lines in Figure 8-9. The deterministic malware model is run for a fixed network size of 1000 devices (N_n) with those calculated to be susceptible as N_s at an infection rate of β_s using equations (6-1),(6-3),(6-4). Both the resistance and resilience parameters are measured from the output of the simulations, where 100% resistance equates to the point at which there is no malware outbreak. The calculated susceptibility values S_p for specific AND and OR malware attacks are shown in Figure 8-11 (a) and (b) respectively. The dashed lines are representative of an R_0 value of 1 for an exploit richness of 1, and correspond to the calculated non discretised V_R values. Figure 8-11 (c) and (d) show the simulated number infected for the AND and OR malware types, each with one exploit, respectively for above and below the $R_0 = 1$ critical threshold. Variant richness values corresponding to $R_0 > 1$ show a clear malware outbreak with devices being infected. Variant richness values corresponding to $R_0 < 1$ show the single infection dying away.

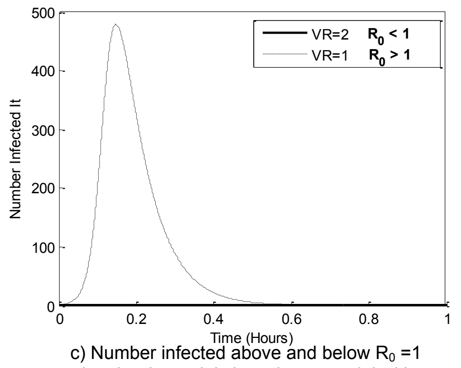
Figure 8-11 (e) and (f) show the simulated resistance from the deterministic malware model for the AND and OR malware types respectively by measuring the final size of the epidemic simulated and using the resistance equation (6-19) to determine the resistance. For each specific malware simulated, as diversity increases so too does the resistance, until it asymptotically approaches 100% past the critical variant richness value where $R_0 < 1$. The calculated points for the single exploit case are indicated as dashed lines. The result is as expected and matches the variant richness values calculated in Figure 8-9. With recovery mechanisms in place therefore it is possible to determine from the simulated resistance, the variant richness required to mitigate against specific malware types.



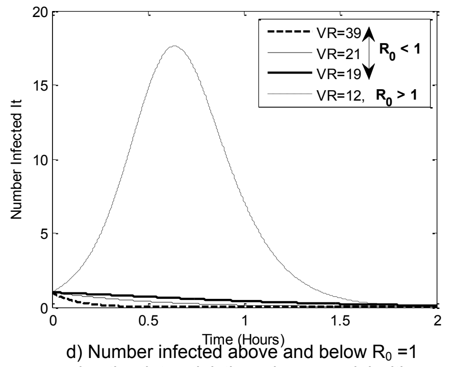
a) Recalculated Susceptibility of the 4 locus AND type, with R_0=1 located at S_p=0.2



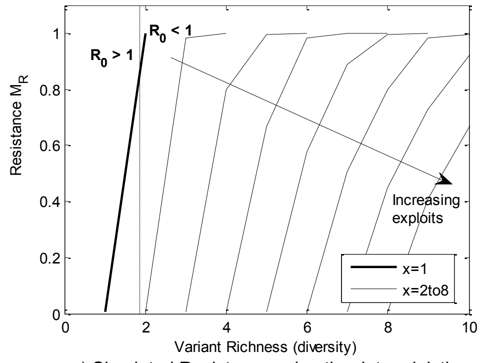
b) Recalculated Susceptibility of the 4 locus OR type, with R_0=1 located at S_p=0.2



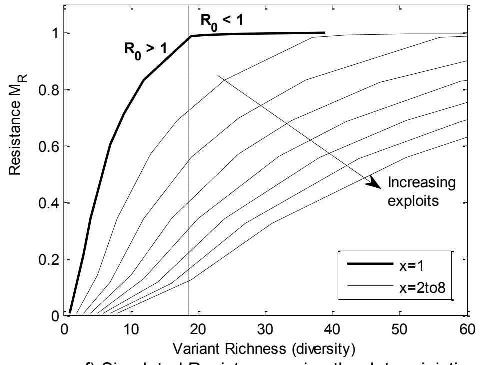
c) Number infected above and below R_0 =1 using the deterministic malware model with 4 locus AND type, and an exploit richness of 1,



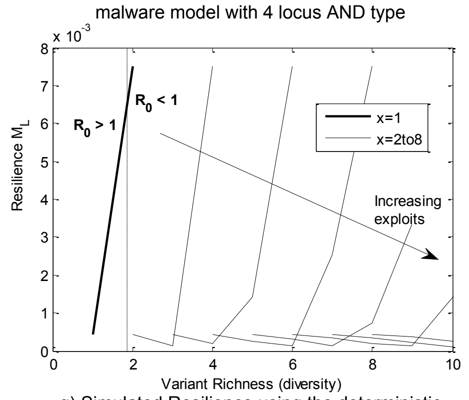
d) Number infected above and below R_0 =1 using the deterministic malware model with 4 locus OR type, and an exploit richness of 1,



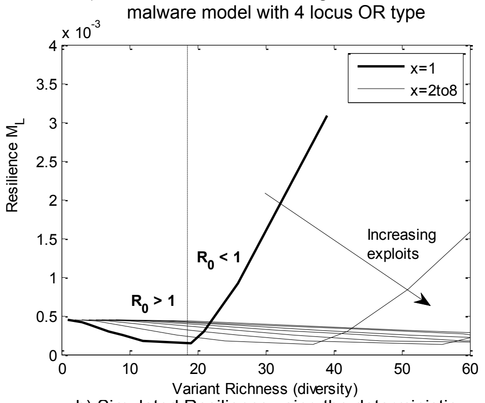
e) Simulated Resistance using the deterministic malware model with 4 locus AND type



f) Simulated Resistance using the deterministic malware model with 4 locus OR type



g) Simulated Resilience using the deterministic malware model with 4 locus AND type



h) Simulated Resilience using the deterministic malware model with 4 locus OR type

Graphs show calculated optimum diversity and simulated resistance and resilience outputs from the deterministic malware model

Figure 8-11 – Simulated resistance and resilience to mitigate an attack

Figure 8-11 (g) and (h) show the measured resilience for the OR and AND malware types respectively by recording the reciprocal of the time in which the number recovered settles to within 1 device of the final value (§6.4.5.2). As diversity increases towards the point of mitigation, the resilience actually worsens. The reason for this can be explained: As diversity increases, fewer devices are susceptible to the malware but the overall density of the devices remains the same. This results in the malware taking longer to spread since it will take longer for a susceptible device to come into contact with an infected device. Consequently this means longer to recover from the point of initial infection (the infection curves become shallower and more spread out). This has the result of reducing the resilience until it reaches a minimum at the $R_0 = 1$ point. As diversity is increased further beyond this point resilience rapidly increases as the malware infection dies away faster. This result indicates that for the SIR epidemic model, an increase in diversity can either reduce or increase resilience depending upon which side of the $R_0 = 1$ line it sits. However the model assumes that recovery can only occur after infection has already taken place, since this is what happens in a biological system where the recovery rate indicates the average time in which an individual remains unwell before recovering. In a practical computing network patching and antivirus updates now tend to occur at regular intervals regardless of whether a device is infected and recovery could occur whilst still susceptible potentially changing the resilience response to diversity and the point of mitigation and is discussed further in future work (§9.3.1.3).

8.2.3 Section Summary

This first section detailed the results of the epidemic based constrained diverse system model. Despite its constraints it is able to model some general diversity principles in relation to an ad hoc network ecosystem. The OR malware type poses a higher security risk over the AND malware type since the more exploits there are available, the greater the overall susceptibility. Additionally, malware with multiple exploits targeting the same locus is a greater threat than those targeting multiple loci with the same number of multiple exploits. Increasing static diversity in terms of variant diversity in the ad hoc ecosystem can either increase or decrease susceptibility of the devices depending upon which variants the malware is targeting and the starting point of diversity in terms of variant frequencies. When absolute maximum diversity is already achieved in an ad hoc ecosystem for a fixed number of loci and variants, increasing diversity further in terms of variant richness reduces the susceptibility and hence the security risk for both the OR and AND malware types.

When there are no recovery measures in place the susceptibility defines how resistant the ad hoc ecosystem is since all those susceptible eventually become infected with the malware. With no recovery, static diversity can be optimised to tolerate a specific type of attack in order to maintain a specified quality of service. The process of diversity optimisation can be used to inform the minimum number of software gene variants required at each locus of a software stack genotype. The general method of diversity optimisation can be used to choose from several solutions, which could benefit situations where there are a limited number of variants available at one particular locus, such as a limited number of operating system core library variants, and an abundance of variants at another. This could be particularly useful for guarding against the AND malware type where there are relations across loci. The time taken for the malware to fully spread, denoted as the peak infection time, can be calculated from a specified quality of service tolerance or a specific type of attack.

When there are recovery measures in place such as through the release of antivirus signatures and software patches it may be possible to have a higher level of susceptibility whilst maintaining an adequate quality of service. There is a trade off between optimising diversity, maintaining quality of service, and the speed of recovery. Additionally, when there is recovery, diversity can be used to not only tolerate, but also mitigate against a specific attack. The minimum variant richness is calculated to prevent the spread of specific malware types which occurs when the reproduction number is less than unity. For the four locus AND malware type, such as a Bluetooth version of the Stuxnet worm, mitigation of the attack at the network level could have occurred with a variant richness of 6 assuming it could be detected and patched within a couple of hours.

Ecosystem resistance and resilience can be measured from the malware model given a constrained set of input parameters. Resistance to malware increases with static diversity, which asymptotically approaches 100% once past the critical mitigation point. Simulated resistance can therefore also be used to determine the necessary diversity needed to mitigate an attack. Static diversity can both reduce and increase resilience depending upon which side of the mitigation point it sits. As diversity increases fewer devices are susceptible resulting in the malware taking longer to spread and consequently longer to recover (since the model assumes recovery only takes place after an infection), and hence reducing the resilience until mitigation occurs. As diversity is increased further beyond this point resilience rapidly increases as the malware infection dies away faster.

8.3 Diverse System Model: Agent-Based

8.3.1 Constrained Agent System Model as an Epidemic Model

Firstly, the agent model is constrained by the random encounter (RE) network model, static diversity, and the SIR malware model as detailed in chapter 7, and compared to the epidemic model by measuring susceptibility and simulating the mitigation of the different malware attack types. The purpose is to verify and baseline the agent model to allow further comparisons with additional and dynamic mechanisms. All simulations are conducted at maximum variant diversity for a given number of variants.

8.3.1.1 Susceptibility Relationships: Agent vs. Epidemic

The proportion of devices susceptible to a particular malware type within a static diversity system of the agent model is compared to that calculated using the epidemic model equations (§6.3.2) and is given in Figure 8-12. The initial diversity conditions are generated using the static diversity random assignment (§7.5.1), which when matched directly with malware gives the initial susceptibility of the network. The network was simulated with 1000 nodes over 10 runs (although the same accuracy can be achieved with 10 nodes over 1000 runs). The susceptibility influences how many and how quickly the devices become infected and the result is generated from the combination of diversity and malware type. For the epidemic model only those susceptible are simulated in a malware attack. For the agent model, all devices are simulated, however it is necessary to ensure that the number susceptible can be adequately represented within the population to achieve an accurate result without requiring the full calculated minimum network size (§7.8.1).

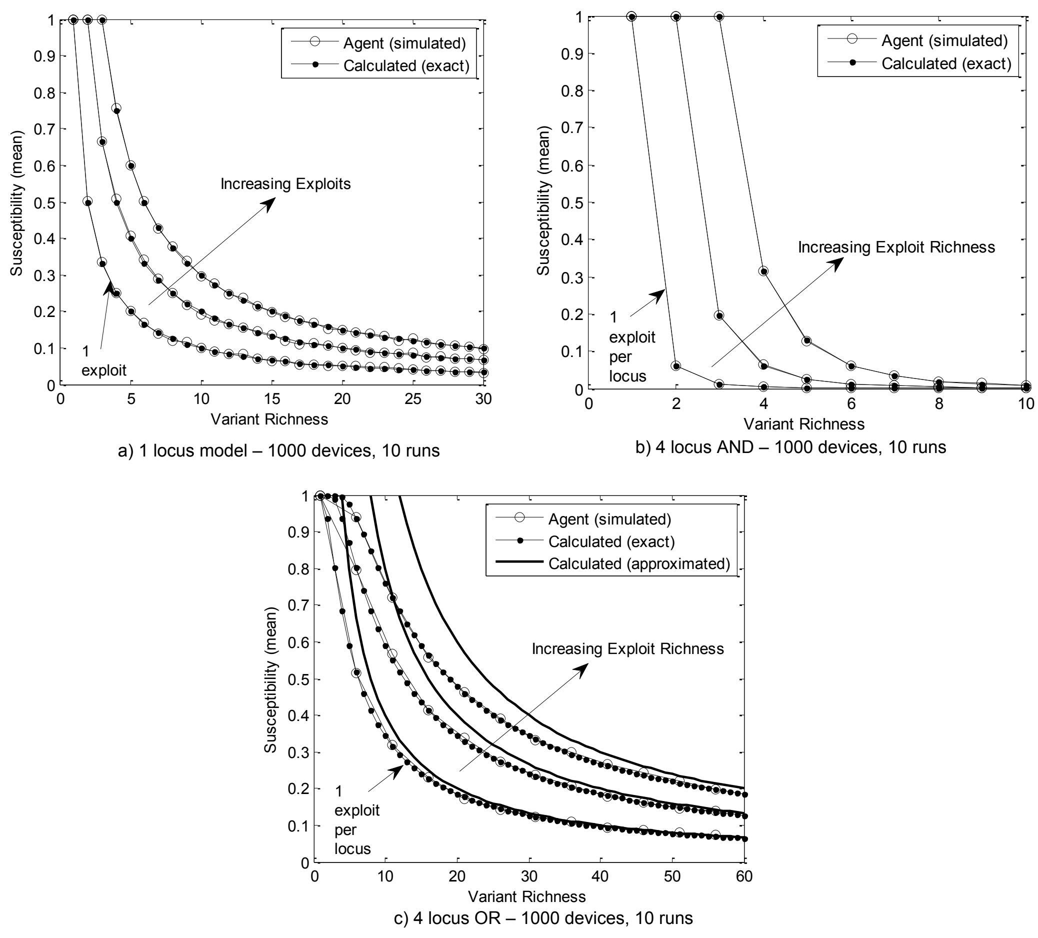


Figure 8-12 – Susceptibility: agent model vs. epidemic model

The simulation includes the variant richness parameter range used in subsequent results to capture the point of mitigation for the examples modelled (Upper bound of 10 for the 1 locus malware, 30 for the 4 locus OR malware, 5 for the 4 locus AND malware). Figure 8-12 confirms how the simulated agent model susceptibility result is closely aligned to the calculated epidemic model result for the static diversity scenario averaged over the 10 runs.

8.3.1.2 Comparative Mitigation of an Attack

For the epidemic model, the mitigation point of an attack where 100% resistance occurs, for a given exploit richness and malware type can be calculated to determine the necessary variant richness and is previously shown in section 8.2.2.5, for fixed γ/β_n ratios. The constrained agent model behaves as a stochastic epidemic model (§7) and so the resistance and resilience output measurements of the deterministic example along with the approximation properties of the stochastic SIR (§4.3.3.6) are used to predict the measured agent result. As with the deterministic epidemic model example a fixed γ/β_n ratio of 0.2 is used where $\beta_n = 0.02 \, and \, \gamma = 0.004$. Comparisons are conducted for an exploit richness (E_R) of one so that the calculated variant richness (V_R), using equations (6-33) and (A-8), to mitigate an attack under *absolute maximum diversity* (§5.3.2.3) conditions is estimated for the agent result as given in Table 8-4. Results from the agent model are shown with an exploit richness of one, to focus upon additional aspects that the mathematical model does not simulate. Increasing the exploit richness will follow the relationships already shown by the mathematical model with the same input conditions (§8.2).

Table 8-4 - Calculated variant richness to mitigate an attack

Malware type	**Exploit Richness (E_R)**	**Calculated mitigation point: Variant Richness**	**Discrete mitigation point: Variant Richness (V_R)**
One locus model	1	5.00	5
4 loci AND	1	1.50	2
4 loci OR	1	18.43	19

Simulations are conducted with 1000 devices. For the stochastic and agent models each data point is averaged over 100 runs. Statistical properties of the output simulations are graphically compared, in Figure 8-13 to Figure 8-15, to the approximation calculations of the stochastic SIR (§4.3.3.6) to confirm accuracy of the results. These properties include the proportion of major outbreaks (Equation 4-41), the mean resistance (determined from the mean of the final size – Equation 4-42), and the standard deviation (Equation 4-43) of the output distributions over the variant richness range of interest.

8.3.1.2.1 One Locus Model

The results for the one locus model are given in Figure 8-13 with the compared resistance and resilience given in (a) and (b) respectively. The dashed lines represent the calculated mitigation point ($R_0 = 1$) before rounding to the nearest whole V_R number (note for the one locus model $R_0 = 1$ produces an exact V_R of 5). The measured resistance of the simulated agent and stochastic models are derived from the mean of the final size of the major outbreak distribution, which is shown for the agent model in Figure 8-13 (c) and (d) for two different V_R values. Figure 8-13 (c) shows the result when $V_R = 2$. Under this condition $R_0 \gg 1$ and the minor and major outbreak distributions are far apart. The measured proportion of major outbreaks for each V_R of the agent and stochastic models is given in Figure 8-13 (e) and is closely comparable to the stochastic approximation calculation ((4-42). The infection curves for the simulated outputs are given in Figure 8-13 (f) showing the resultant mean of only the major epidemics against the deterministic model.

Outputs for $V_R = 1,2,3$ are clearly visible and closely resemble the deterministic result.

As R_0 approaches 1, the minor and major distributions merge together as shown in Figure 8-13 (d) for a V_R of 5 at the calculated mitigation point ($R_0 = 1$), thus making it difficult to distinguish the separation between the two distributions. When major outbreaks are no longer detected (determined by a specified cut off point between the two distributions) only minor outbreaks remain. At this point the simulated V_R is assumed to be the point of malware mitigation.

Due to the stochastic nature of the model's output around the mitigation point major outbreaks may still be occurring: It depends on how close the V_R point is to the mitigation point. In the case of the one locus model the V_R point of 5 sits exactly at the mitigation point. For the agent model some major outbreaks will still occur due to the way in which the software variants are assigned randomly from a uniform distribution. The measured susceptibility distribution (s_p) of the agent model at $V_R = 5$ is shown in Figure 8-13 (h). The calculated s_p at $V_R = 5$ is 0.2 equating to 200 devices. The distribution sits around this point leading to some runs with major outbreaks (those above 200 susceptible devices). By the time $V_R = 6$ only minor outbreaks are detected and the resistance for both stochastic and agent models is measured as 100%. Both the stochastic and agent models have therefore detected the mitigation point to be located at $V_R = 6$.

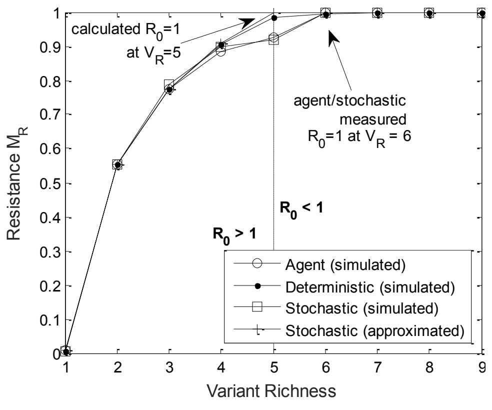
The resilience measured by both the stochastic and agent models as shown in Figure 8-13 (b) reduces with variant richness to a minimum at the mitigation point due to the infection taking longer to spread and hence longer to recover before jumping to maximum resilience where no major outbreaks occur. The deterministic model however rises comparatively slower as there is no distinction between minor and major epidemics, with the infection dying away faster as diversity increases (§8.2.2.6).

8.3.1.2.2 Multiple Loci

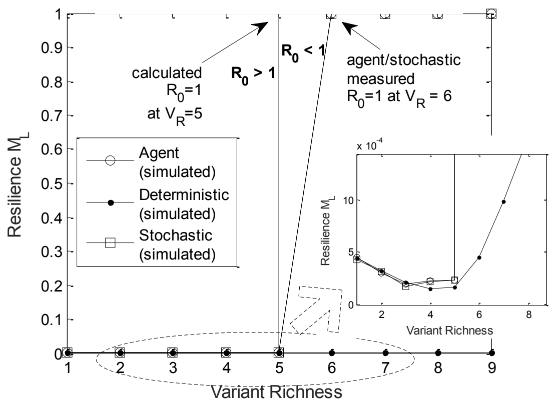
Results for the four locus AND and OR malware types are given in Figure 8-14 and Figure 8-15 respectively. For the AND result both the agent and stochastic models measure the mitigation point correctly as $V_R = 2$ due to three factors. Firstly, the true calculated mitigation point lies under $V_R = 2$ and is shown by the dashed line in Figure 8-14 (a), (b), and (e) so that when $V_R = 2$ there is already no trace of major outbreaks occurring. Secondly, Figure 8-2 of section 8.2.1.2 shows the relationship between V_R and s_p for the different malware types. The AND malware type shows a steep gradient so that when V_R is increased from 1 to 2 there is a large change in s_p (s_p shifts from 1 to 0.0625) which corresponds to $R_0 \gg 1 \; to \; R_0 \ll 1$. This gives the clear simulated result of the mitigation point. Thirdly the approximated standard deviation is very small for a low-valued V_R and so the error will be very small over 100 runs. This can be seen in Figure 8-14 (g) where the measured standard deviation is very close to the approximated value.

For the OR result as shown in Figure 8-15 there is a larger difference between the measured and calculated result for the opposite reasons to the AND case. The steepness of the V_R versus s_p gradient at the critical s_p value of 0.2 ($R_0 = 1$) is much shallower than the AND case and also more shallow than in the one locus model. Therefore, several of the V_R points lie very close to the critical s_p value with the susceptibility distributions overlapping the mitigation point. The approximated standard deviation of the distribution is very large as shown in Figure 8-15 (g), causing the minor and major outbreaks to merge together and making it difficult to establish a cut off point between the two distributions. This larger standard deviation increases the uncertainty of the mitigation point which must then be extended to a point where no major outbreaks are detected, which occurs after the calculated ($R_0 = 1$) point. For the agent model this is measured as $V_R = 24$ and in the stochastic model measured as $V_R = 23$.

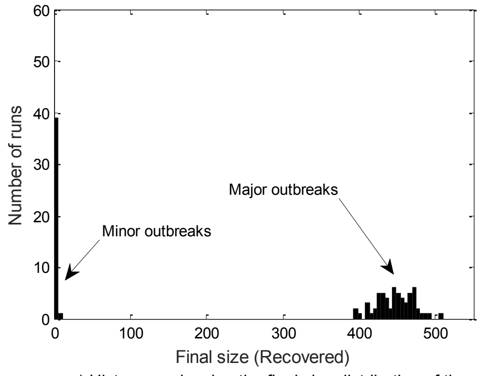
In conclusion therefore the constrained agent model is representative of the stochastic model, however there may be differences in the measured point of mitigation. The stochastic and agent models may show a higher diversity requirement to mitigate an attack than the approximated or deterministic result.



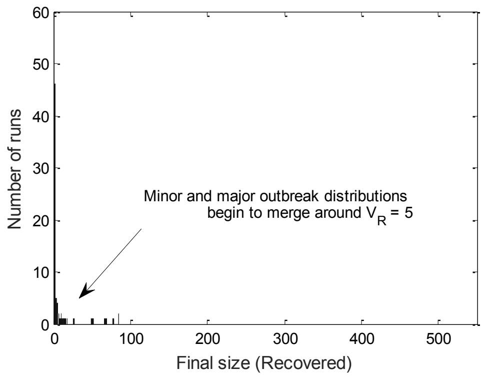
a) Resistance to the one locus malware type with an exploit richness E_R=1 for a fixed βn/γ for varying V_R. Shows the amount of V_R needed to mitigate the attack.



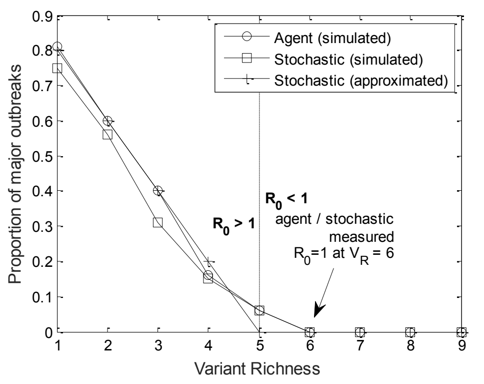
b) Resilience of the one locus malware type with an exploit richness E_R=1 for a fixed βn/γ for varying V_R.



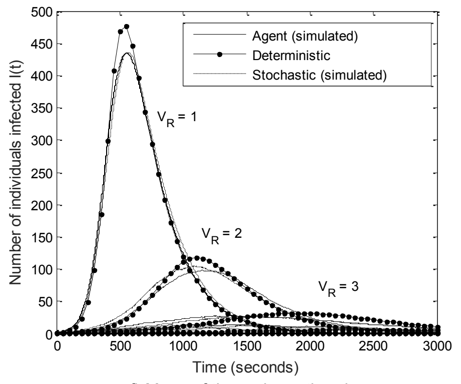
c) Histogram showing the final size distribution of the agent model when V_R = 2. The minor and major outbreak distributions are clearly separate.



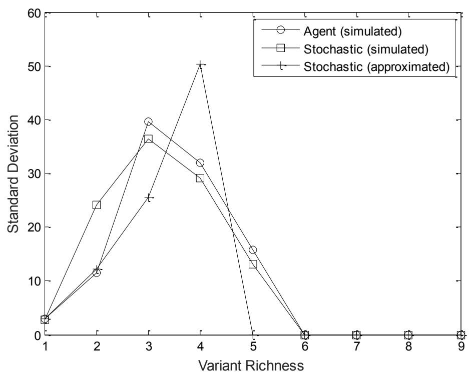
d) Histogram showing the final size distribution of the agent model when V_R = 5 at the calculated R_0=1 point.



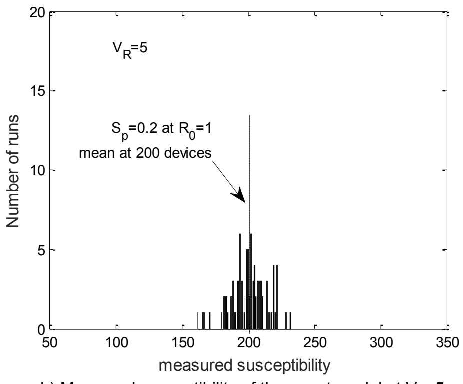
e) Probability of a major outbreak



f) Mean of the major outbreaks

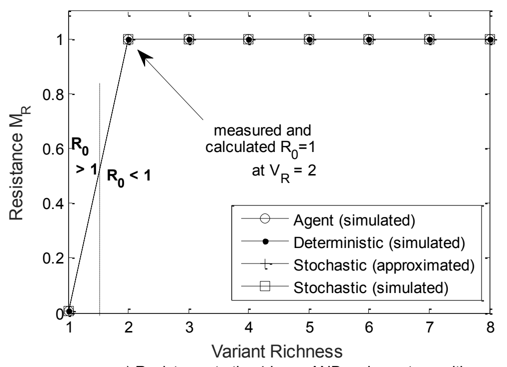


g) Standard deviation of the major distribution

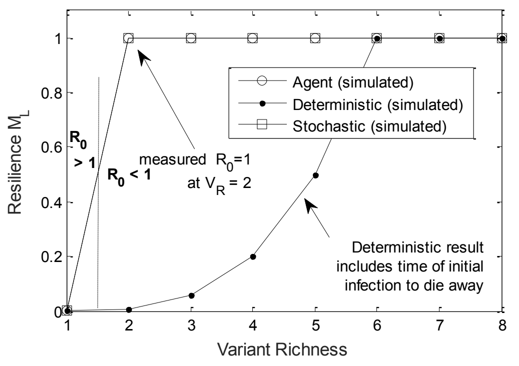


h) Measured susceptibility of the agent model at V_R=5

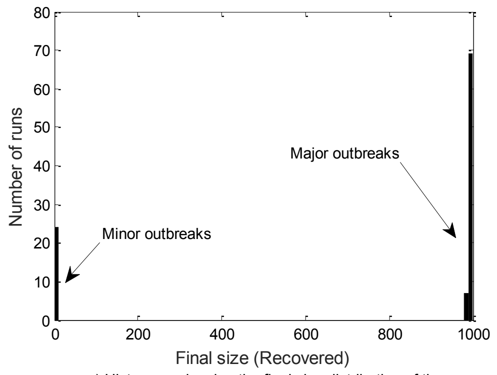
Figure 8-13 - One locus model: agent vs. stochastic and deterministic



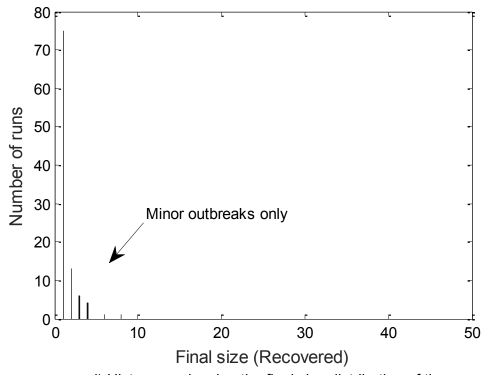
a) Resistance to the 4 locus AND malware type with an exploit richness E_R=1 for a fixed βn/γ for varying V_R. Shows the amount of V_R needed to mitigate the attack.



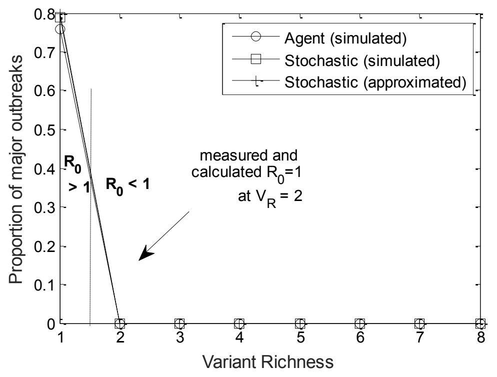
b) Resilience of the 4 locus AND malware type with an exploit richness E_R=1 for a fixed βn/γ for varying V_R.



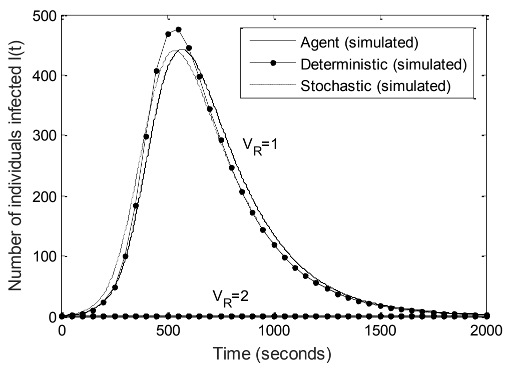
c) Histogram showing the final size distribution of the agent model when V_R = 1. The minor and major outbreak distributions are clearly separate.



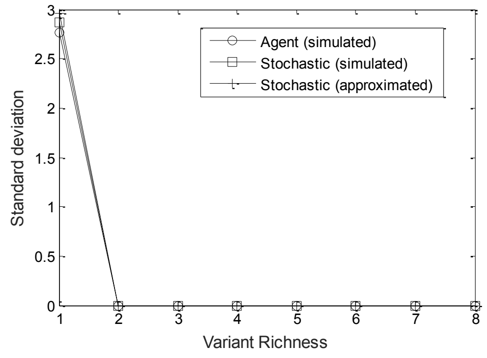
d) Histogram showing the final size distribution of the agent model when V_R = 2 where only minor outbreaks occur.



e) Probability of a major outbreak

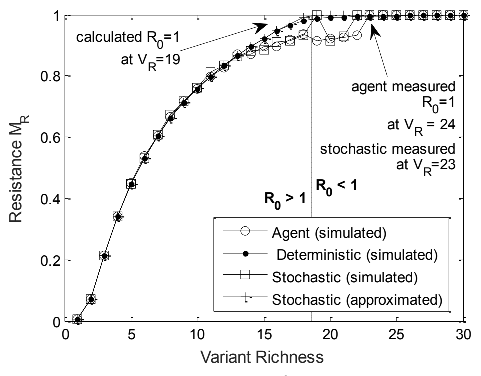


f) Mean of the major outbreaks

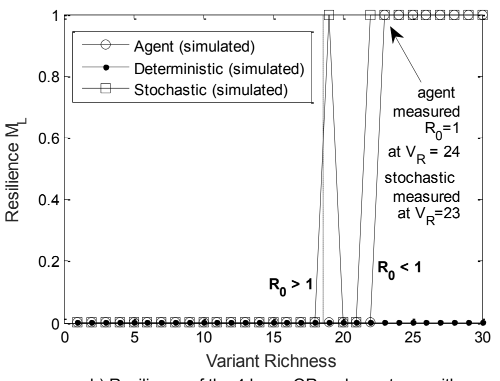


g) Standard deviation of the major distribution

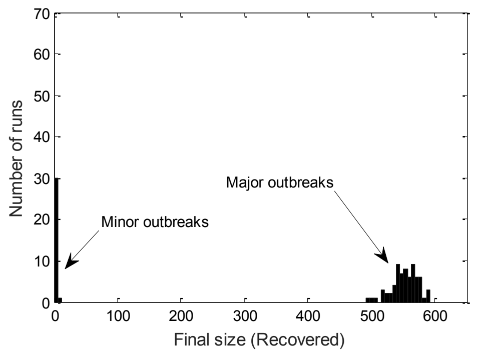
Figure 8-14 – Four locus AND malware: agent vs. stochastic and deterministic



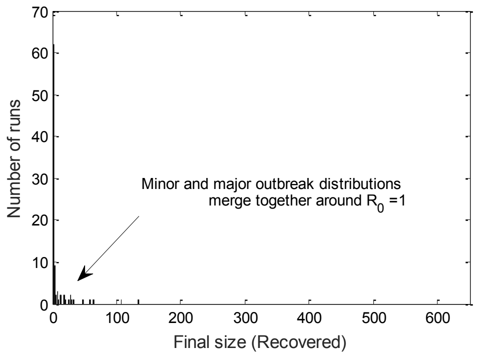
a) Resistance to the 4 locus OR malware type with an exploit richness E_R=1 for a fixed $\beta n/\gamma$ for varying V_R. Shows the amount of V_R needed to mitigate the attack.



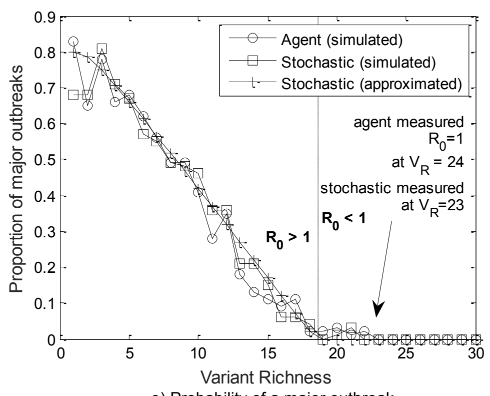
b) Resilience of the 4 locus OR malware type with an exploit richness E_R=1for a fixed $\beta n/\gamma$ for varying V_R.



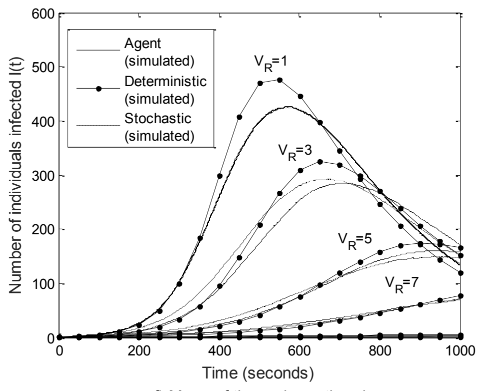
c) Histogram showing the final size distribution of the agent model when V_R = 5. The minor and major outbreak distributions are clearly separate.



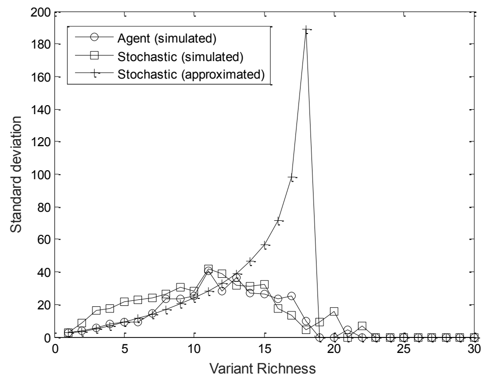
d) Histogram showing the final size distribution of the agent model when V_R = 19 close to the calculated R_0=1 point.



e) Probability of a major outbreak



f) Mean of the major outbreaks



g) Standard deviation of the major distribution

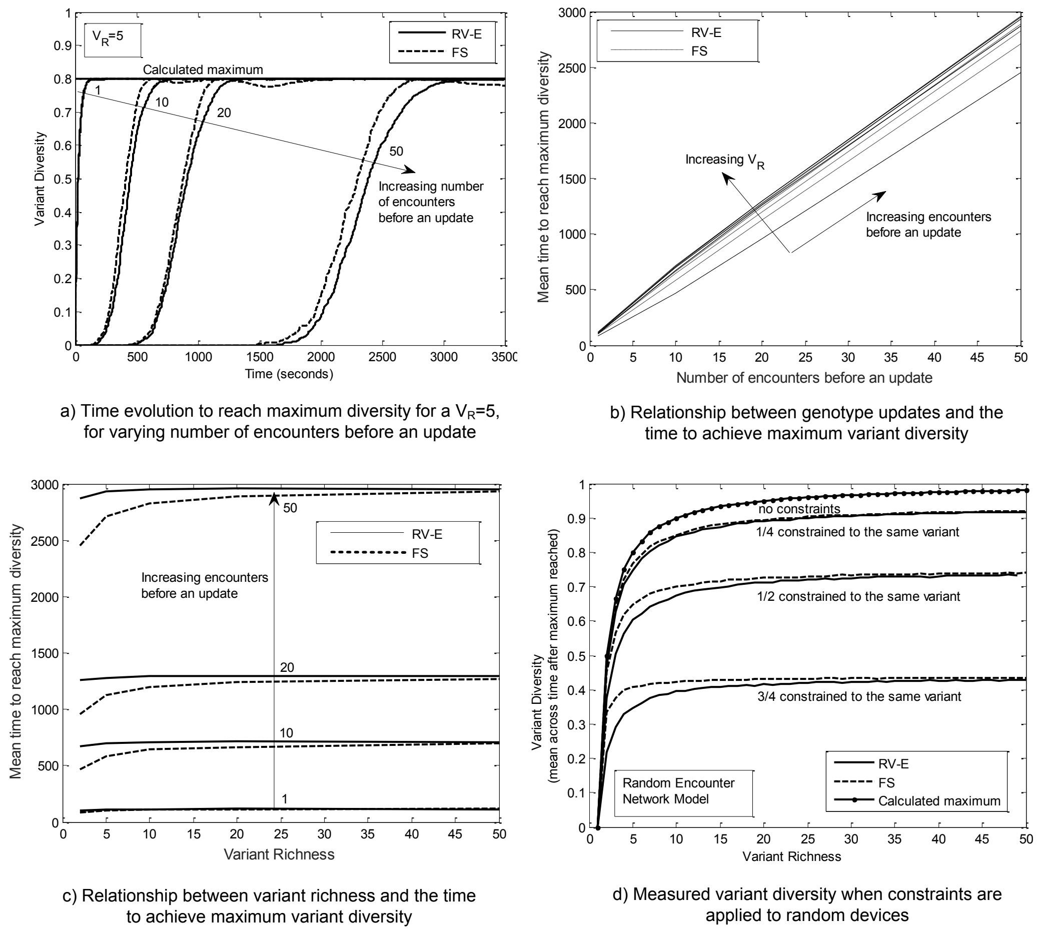
Figure 8-15 – Four locus OR Malware: agent vs. stochastic and deterministic

8.3.2 Dynamic Diversity Performance with the RE Network Model

For the static diversity case modelled by the constrained mode of operation, maximum diversity is already achieved through the uniformly distributed pre-assignment of variants. In a practical ad hoc network diversity may not be pre-assigned, or the starting condition may have no diversity at all in a worst case scenario. In order to achieve maximum diversity and be able to adapt to changing information and constraints, diversity is assigned dynamically using continuous dynamic updating (§7.6). The performance of the random variant (RV) algorithm, which can be seen as an extension to the static case, and the Favourability Score (FS) algorithm, which assumes distributed knowledge, in reaching the maximum obtainable variant diversity is compared with the random encounter (RE) network model (see §8.3.6 for the random waypoint (RWP) network model).

8.3.2.1 Ideal Scenario

The maximum variant diversity that can be achieved in an ideal scenario where there are no practical constraints is limited by the number of variants and can be calculated using equation (7-2). The time taken to achieve this limited maximum variant diversity, given a fixed contact rate, from a starting point where all devices have the same set of variants and hence genotypes (i.e no diversity), is dependent upon how often the devices are updated. Figure 8-16 (a) shows the time evolution of the network in reaching maximum diversity for four different update rates (number of encounters before an update) when the variant richness $V_R = 5$. The linear relationship with time is shown in Figure 8-16 (b) where it tapers towards a point when updates are performed at every encounter.



a) Time evolution to reach maximum diversity for a V_R=5, for varying number of encounters before an update

b) Relationship between genotype updates and the time to achieve maximum variant diversity

c) Relationship between variant richness and the time to achieve maximum variant diversity

d) Measured variant diversity when constraints are applied to random devices

Figure 8-16 - Dynamic diversity performance: random encounters

The difference between the random variant algorithm (RV-E) and the favourability score (FS) algorithm is additionally shown in Figure 8-16 (b), but is better seen in Figure 8-16 (c) as V_R is varied. The contact rate is fixed at 0.02 and the network size is fixed at 1000 devices to be consistent with previous results (§8.3.1.2). The contact rate is fixed to compare the mean time to reach maximum diversity (Figure 8-16) for the two different algorithms. The FS algorithm has a fixed buffer size of 10 genotypes representing a 1% view of the network at any point in time. For a changing network, recording too many genotypes will capture too much historical data and will be unrepresentative of local genotype configurations both in terms of physical locality and time. A buffer size of 10 is chosen to perform decisions based upon the most recently encountered devices and their configurations. For small V_R values, which would be likely in practice if commercial off the shelf (COTS) software programs are utilised to generate the different variants (§3.2.1.3,3.4.4.6), the FS algorithm is faster at reaching maximum diversity. This can be explained by considering the initial conditions. When the initial point of diversity is at a minimum (all devices have the same dominating variant), the dominating variant has an equal probability of being picked again for the RV-E algorithm relative to the other variants, whereas it is less likely to be chosen initially by the FS algorithm since it is already being used with a high frequency. The probability of picking the dominating variant reduces as the variant richness increases hence the time to reach maximum diversity in this case is approximately the same for both algorithms. For the RV algorithm that is dependent upon time only (RV-T), maximum diversity can be reached in one time interval (not shown in the graphs) when the update rate is set to one time interval. The fastest way to achieve maximum diversity for the distributed algorithm is to perform updates at every successful encounter where new genotype information is available.

8.3.2.2 The Constrained Scenario

For a generically constrained scenario, the calculated maximum obtainable variant diversity (§7.4.2) can no longer be reached, instead the actual diversity achieved can be measured and is compared between the RV-E and FS algorithms. A generically constrained scenario refers to anything in the ad hoc ecosystem constraining the availability of, or reducing the use of variants, and thus limiting the variant diversity (§7.4.3). For example those caused by user desirabilities or hardware constraints, the blacklisting of variants, or software incompatibilities. Constraints are applied to both algorithms such that a proportion of devices (¼ , ½, ¾) are limited to having a selection choice of the same single variant at a single locus such that for the constrained devices they appear to have no diversity between them which remains static over time. The remaining devices continue to have a choice of all variants.

Figure 8-16 (d) shows the resultant variant diversity of the network as V_R is varied for a single locus. The differences between the achievable diversity reached by the two algorithms can be explained. For lower V_R values, the FS algorithm can obtain a higher diversity since it will avoid using the constrained variant whereas the RV algorithm continues to assign the constrained variant using the same uniform distribution. This also accounts for the larger difference when more devices are constrained. For a large V_R, there is less difference in achievable diversity between the two algorithms, since there are more variants to choose from and the likelihood of choosing the constrained variant diminishes.

In an ideal scenario therefore both the RV and FS dynamic diversity algorithms can achieve maximum diversity, with the FS algorithm performing faster with the same input conditions. When a realistic scenario is simulated where there are likely to be constraints imposed on a proportion of the devices, the distributed FS algorithm can have an advantage over the RV algorithm by achieving a higher variant diversity level. The impact of constraints and differences in variant diversity on malware propagation is considered in section 8.3.5.

8.3.3 Malware Attack within an RE Network with Continuous Updating

8.3.3.1 With no Recovery (SI)

During continuous dynamic updating, the dynamic diversity algorithms aim to maximise diversity, and if left, continue to update and maintain the required diversity level responding to encounters with other devices and changes in the network. Constantly changing configurations can confuse a targeted attacker (§3.4.3.4), but its effect on the spread of malware, when configurations are selected from a common pool is shown in Figure 8-17. Figure 8-17 (a) shows the averaged time evolution of a malware epidemic using the different static and dynamic diversity schemes, with the dynamic algorithms using continuous dynamic updates. No additional security mechanisms or constraints are used. Under this configuration the malware model has two states: S and I, and the FS and RV algorithms perform equally following the same curves. The dynamic algorithms are compared to both the static case with the same variant richness of five, and the case of no diversity. The contact rate is fixed at 0.02, since when there is no recovery, there is no reproduction number and varying the contact rate does not change the magnitude of the final state, only the speed at which it happens. It is the impact of diversity that is the focus, not the changing timescales. The malware epidemic is initiated after the dynamic algorithms have reached their maximum diversity level. As shown in Figure 8-17 (a), for such a dynamic diversity scheme eventually all devices become infected over time, and this is unlike the static case which protects a proportion of devices from becoming infected (for a single malware). The reason being is that as devices randomly interact within a closed space, at some point in time an infected device will still come into contact with a susceptible device. The result of continuous dynamic updating from the same pool of software is that the final size of the epidemic is as bad as the non-diverse case so that the resistance is zero. However, the infection process is slowed down by dynamic diversity and the linear relationship between time and the number of variants is shown in Figure 8-17 (c) and (d). The time taken for the whole network to become infected is also dependent on the contact rate of devices and hence the update rate of genotypes as shown in Figure 8-17 (b) since the malware can only spread at the rate of susceptible contact. Attempting genotype update rates more often than every encounter has no further impact on the network infection time, which can be seen where the RV-T (update rate of one time interval) and the RV-E (update rate of one encounter) follow the same curve.

This suggests that although the continuous dynamic mechanism lengthens the time taken to reach the peak of infection in comparison to no diversity at all, it would be more beneficial to employ static diversity when there are no additional security mechanisms in place to minimise the final size of the epidemic. Static diversity however can be open to targeted attacks on specific devices if their configurations remain fixed and become known for a sufficient amount of time. Additionally, devices in reality do not move in random patterns. For example if moving devices with common vulnerable variants congregate, malware may spread quickly in these areas when configurations cannot be changed. Thus the optimum solution would be to maximise diversity as quickly as possible in a changing network and then remain static for as long as possible, particularly during a malware epidemic. The results also suggest that whilst distributed analysis of local diversity could be beneficial for maximising network variant diversity in a practically constrained scenario (§8.3.2), allowing all devices access to all software may not be effective. Restricting software access would not be realistic when variants are provided by COTS software programs, however automated software generation using the techniques described in section 3.4.2 could be used to locally generate variants from established sources with restrictions on their distribution.

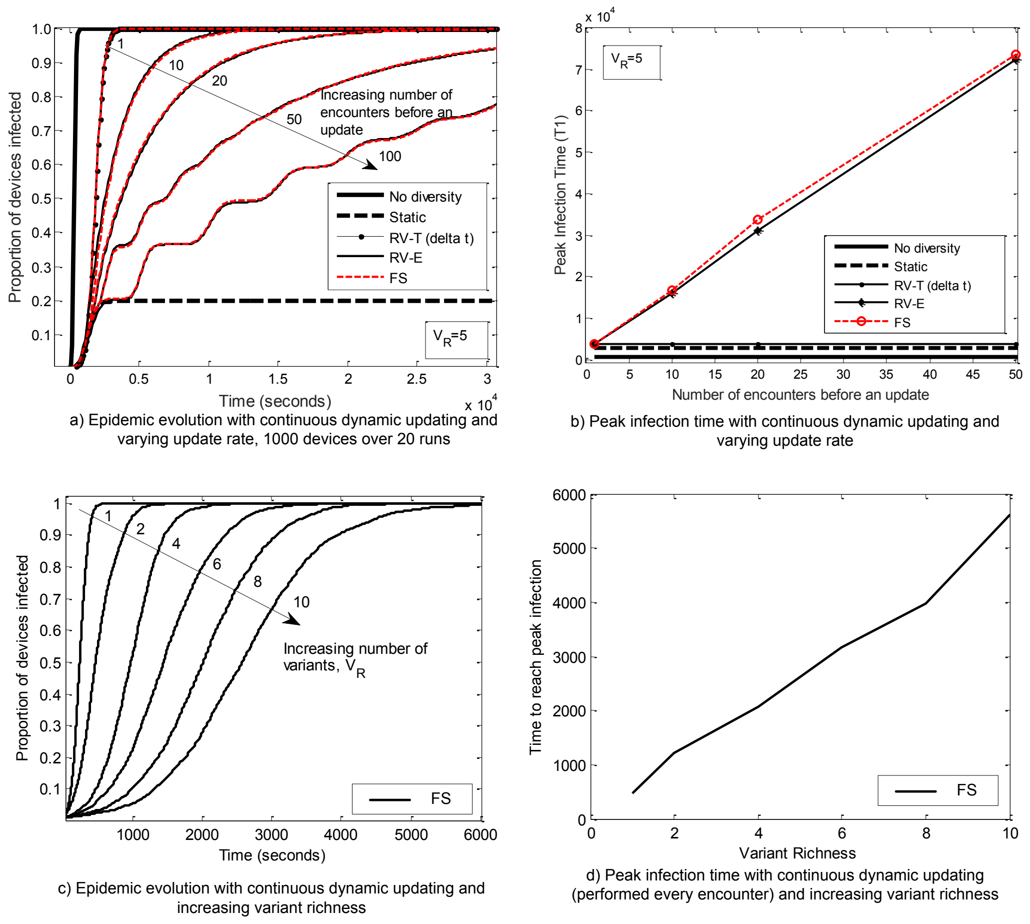
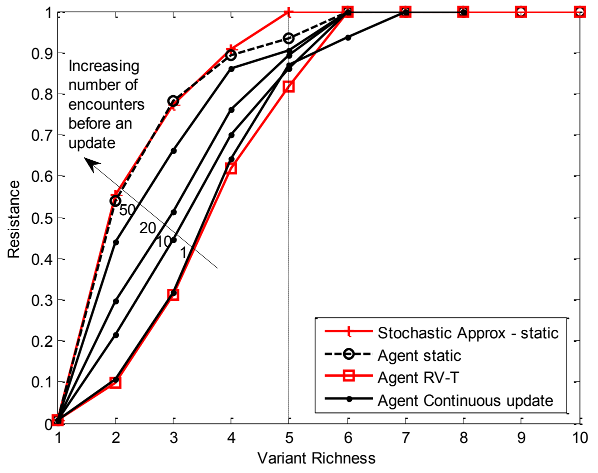


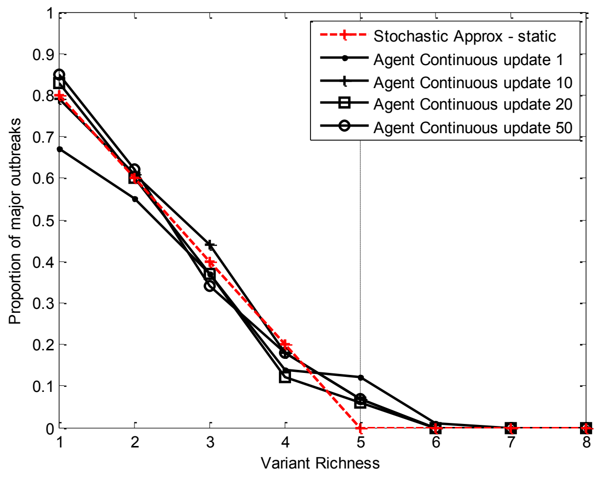
Figure 8-17 - Malware epidemic comparing different diversity schemes

8.3.3.2 With Recovery (SIR)

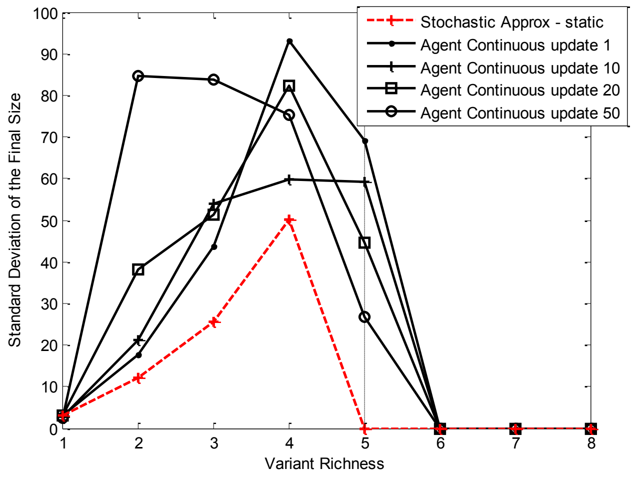
When recovery mechanisms are available to remove malware from infected devices and prevent re-infection, diversity can be used to mitigate an attack even when performing continuous dynamic updates. In fact, the same mitigation point is observed for both static and dynamic diversity as shown in Figure 8-18 for the one locus, and Figure 8-19 for the four locus AND and OR malware types. The proportion of major outbreaks quickly diminishes to zero past the mitigation point. Results are shown for malware with an exploit richness of one. For V_R values less than the mitigation point, the resistance to the malware can be much less for continuous dynamic updating, depending upon the update rate. For the single locus case and the OR type it can be seen that the faster the update rate (small number of encounters), the lower the resistance, until the $R_0 = 1$ point where the amount of variant richness is sufficient to prevent further malware outbreaks. The reduction in resistance due to the dynamic algorithms is not apparent for the AND case shown since for a V_R of 1 all devices are susceptible so there is no diversity, and for a V_R of 2, the $R_0 = 1$ point has already been surpassed, showing no differences in the outcome between static and dynamic diversity.



a) 1 locus - Resistance with continuous update
1000 devices averaged over 100 runs

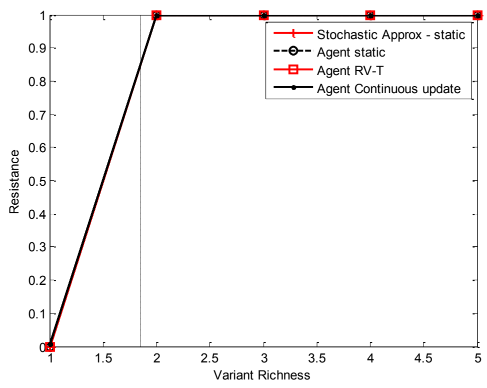


b) 1 locus - Proportion of major outbreaks
1000 devices averaged over 100 runs

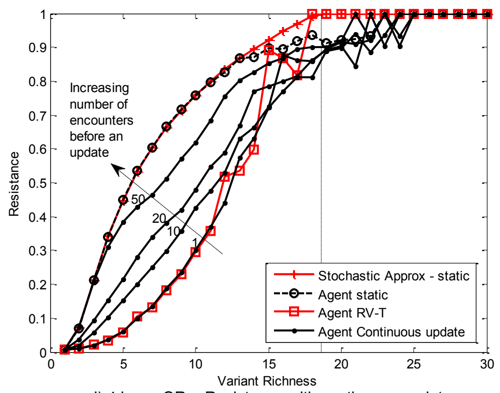


c) 1 locus - Standard Deviation
1000 devices averaged over 100 runs

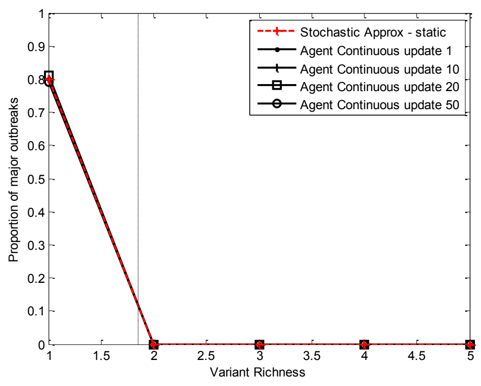
Figure 8-18 – Malware resistance with recovery – one locus



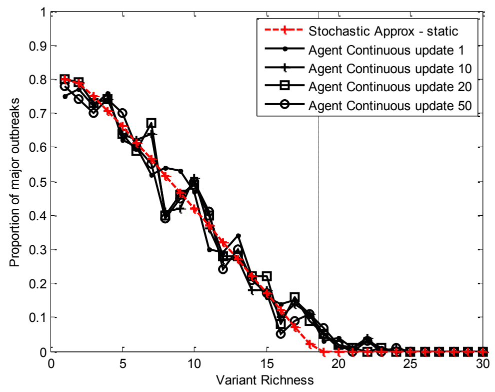
a) 4 locus AND - Resistance with continuous update
1000 devices averaged over 100 runs



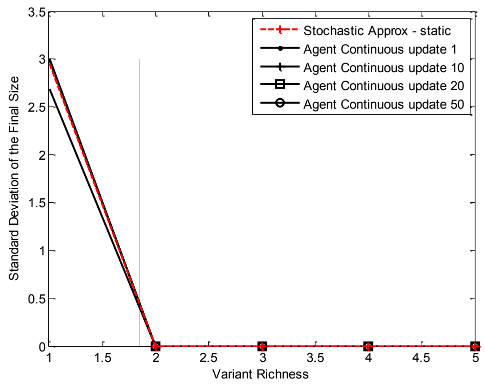
d) 4 locus OR - Resistance with continuous update
1000 devices averaged over 100 runs



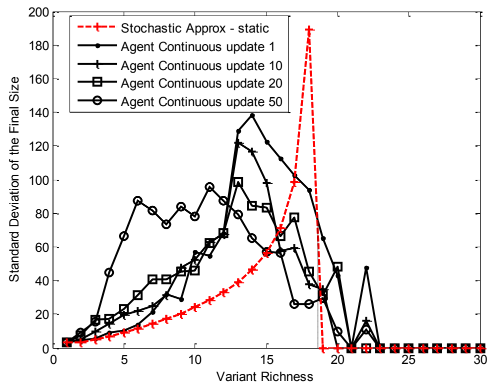
b) 4 locus AND - Proportion of major outbreaks
1000 devices averaged over 100 runs



e) 4 locus OR - Proportion of major outbreaks
1000 devices averaged over 100 runs



c) 4 locus AND - Standard Deviation
1000 devices averaged over 100 runs



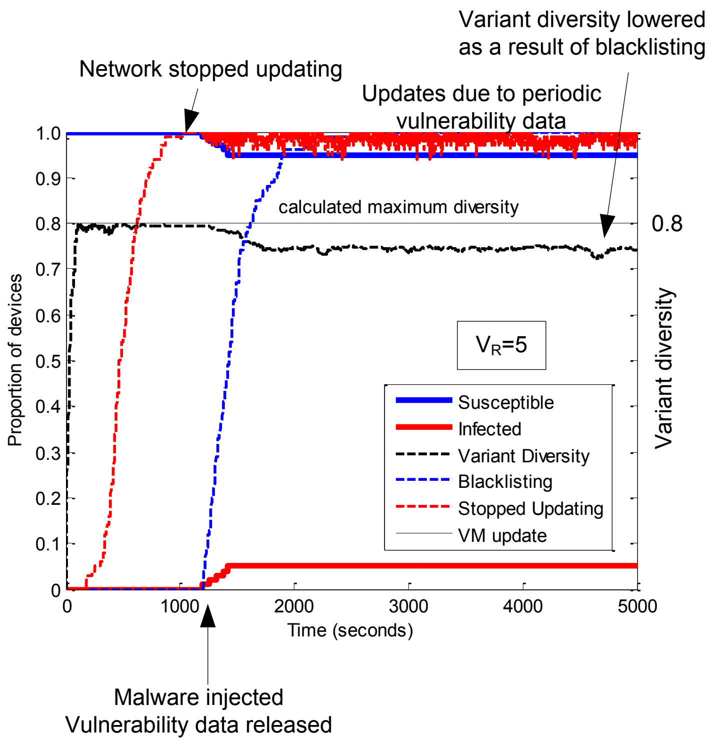
f) 4 locus OR - Standard Deviation
1000 devices averaged over 100 runs

Figure 8-19 – Malware resistance with recovery – AND / OR

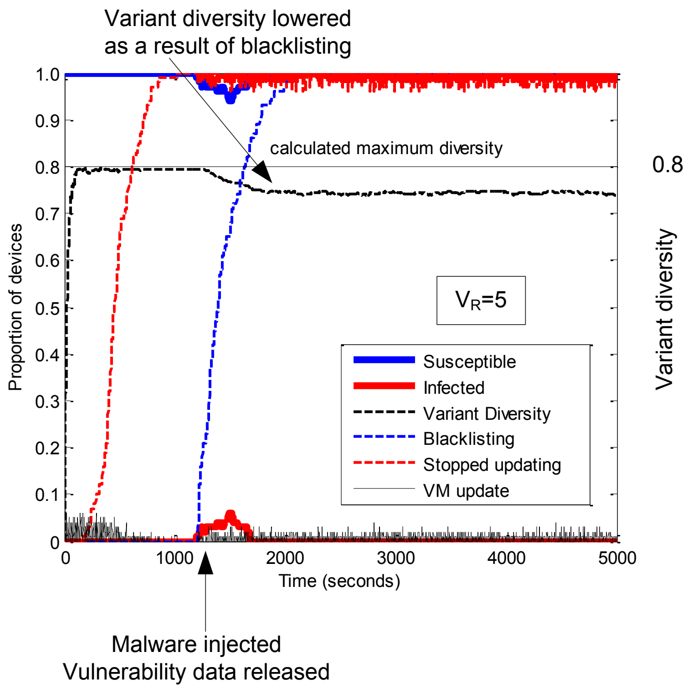
In a fast changing network therefore, where it may be necessary to perform regular updates to maintain maximum diversity or confuse targeted attackers, malware mitigation can still be achieved if there are enough variants available for a given speed of recovery. Below the mitigation point there is a trade off between the update rate to achieve and maintain maximum diversity, and lowering the resistance to malware. A low resistance to malware will lead to a poorer quality of service for the network as a whole (§6.4.3.3).

8.3.4 Malware Attack in an RE Network with Additional Mechanisms

The agent model includes the option to *stop updating* software stack genotypes once variant diversity has been maximised, only resuming the update process if a defined trigger occurs (§7.6.4). Thus, effectively becoming equivalent to the static case if malware is initiated during the static period. This section considers the effect of diversity when additional security mechanisms are present during a malware attack within the static period and how this compares to post infection recovery mechanisms. The additional security mechanisms include *blacklisting* (§7.6.3) of known vulnerable variants, which is only possible when alternative variants are dynamically available, and the effects from the utilisation of a *virtualisation platform* (§7.6.5). The time evolution effects from blacklisting and virtualisation are firstly analysed before considering the comparative resistance and epidemic timescales. The time evolution of a one locus malware epidemic with blacklisting is shown in Figure 8-20 (a) for the FS algorithm. β_n is fixed at 0.02 as with previous examples, and the variant diversity is firstly maximised with a $V_R = 5$. The number of devices that have stopped updating, and the number that are performing blacklisting are measured throughout the simulation and are shown in the graph. After all devices in the network have stopped updating malware is then injected into the system. It is assumed that at the point of injection, the vulnerability has just been publically disclosed and the blacklisting data begins to be disseminated.



a) Epidemic evolution with 'stop updating' and blacklisting for the FS algorithm



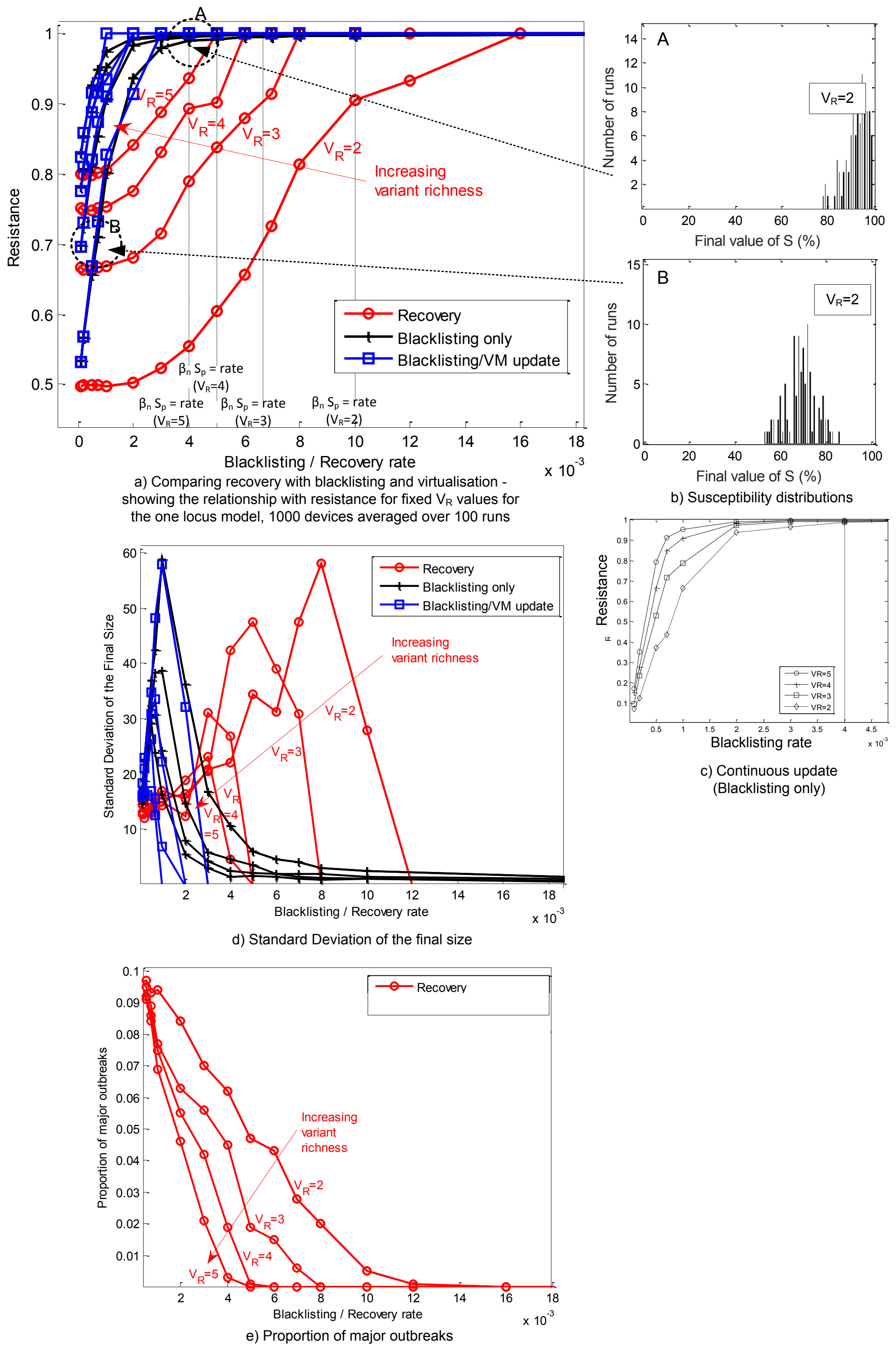
b) Epidemic evolution with Stop updating, blacklisting, and virtualisation (VM updating) for the FS algorithm

Figure 8-20 - Malware epidemic with additional mechanisms

Malware attacks like this can happen as part of a surge in follow on attacks after a zero-day vulnerability has been announced (§3.3.3) and before a patch or antivirus signature has yet to be released. On reception of blacklisting information, devices restart the genotype update process to enable vulnerable variants to be temporarily deselected (§7.6.3) resulting in a drop in the variant diversity of the network. In this scenario the act of *reducing* diversity has made the network less susceptible to a specific malware attack by reducing the representation of those with vulnerable variants resulting in the *dilution effect* (§2.2.2.2.3).

The time evolution of a one locus epidemic with blacklisting plus the additional effects from virtualisation (VM update) is shown in Figure 8-20 (b) where an infection is removed if it is present during a genotype update cycle. This happens during the period of static diversity when a trigger occurs to restart the update process such as new vulnerability data becoming available. In the specific time evolution shown this has had the effect of the final state of all devices returning to or remaining in the susceptible compartment, and in effect recovering the network from the epidemic.

Figure 8-21 (a) shows the comparative relationship between recovery, blacklisting, and blacklisting with virtualisation for the one locus model, against malware resistance for four values of variant richness (V_R). The recovery and blacklisting rates are varied over the same range, with the infection rate (β_n) fixed at 0.02 as in previous examples. Unlike the recovery mechanism, blacklisting does not result in minor and major outbreaks (Figure 8-21 (e)), instead it results in a single distribution of the final number left in the susceptible state corresponding to the resistance. The reason being is that blacklisting impacts only those still susceptible and cannot remove an infection once it is present. Two example distributions at different blacklisting rates for $V_R = 2$ are shown in Figure 8-21 (b) to illustrate this. Resistance to malware is measured by the proportion that does not become infected (§6.4.3). When virtualisation is included it is measured by firstly counting the number of devices that have been infected at least once during the epidemic (since a single device may become infected multiple times), and then subtracting this from the network size. For all security mechanisms the minimum resistance is dictated by the variant richness, since at the lowest rate where the mechanisms do not exist, it is only the effects from static diversity that persist. Comparative results show that blacklisting is more effective at resisting malware when applied at the same rate. This is because recovery mechanisms are applied after infection has already occurred and blacklisting can be effective before infection occurs. However, the mitigation point is the same for blacklisting as it is for recovery, and occurs when the rate increases beyond the rate of new infectives (β_s). So that the point of mitigation occurs when $\beta_n S_p = rate$, where S_p is defined by the malware and variant richness (§6.3.2). For dissemination rates of blacklisting below the mitigation point, variant richness becomes effective at increasing the resistance. Similar to the recovery mechanism, there is a trade off between the speed of dissemination, diversity, and resistance. Figure 8-21 (c) shows the effect of blacklisting during continuous dynamic updating, where although the resistance against malware is very poor for low blacklisting rates, the approach can still very quickly outperform the recovery mechanism.



a) Comparing recovery with blacklisting and virtualisation - showing the relationship with resistance for fixed V_R values for the one locus model, 1000 devices averaged over 100 runs

b) Susceptibility distributions

c) Continuous update (Blacklisting only)

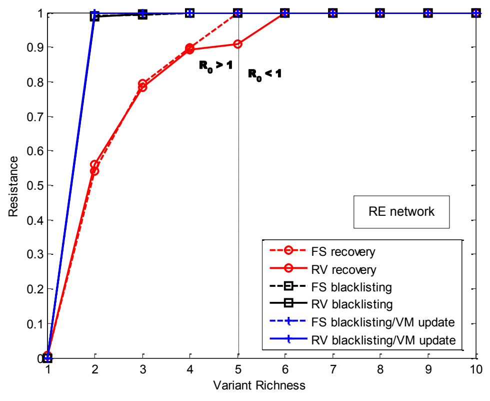
d) Standard Deviation of the final size

e) Proportion of major outbreaks

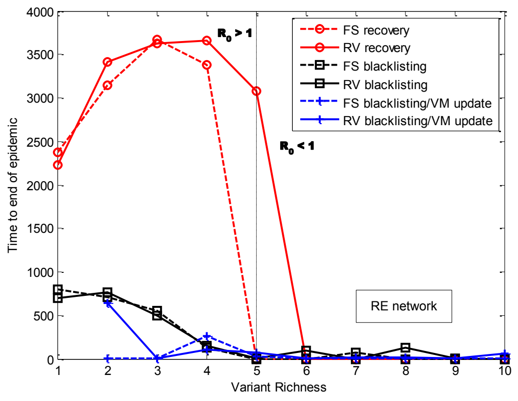
Figure 8-21 – Comparative malware resistance with security mechanisms

The effect of adding virtualisation marginally increases the resistance, but, as stated previously, has the benefit of recovering the network without requiring signature based protection or patching. However without these signature or patching mechanisms the network is open to a repeat attack from the same malware.

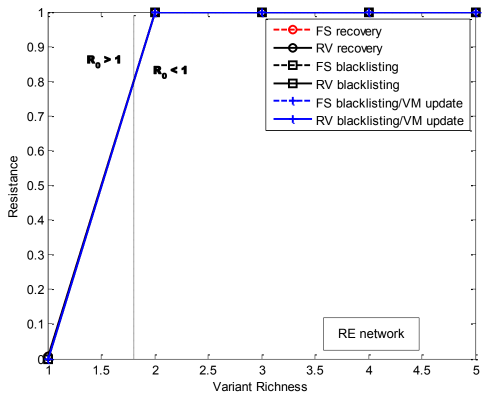
Figure 8-22 shows the comparative resistance and epidemic times for the different security mechanisms for the specific one locus, four locus AND, and the four locus OR examples. The variant richness is varied for the two dynamic algorithms at maximum variant diversity. The contact rate and hence β_n is fixed at 0.02, the recovery rate γ is fixed at 0.004, resulting in a γ/β_n ratio of 0.2 as per previous examples. The blacklisting rate is fixed at the same rate as the recovery to compare resistive performance. For the recovery scenario the end of the epidemic time occurs when, after a major outbreak, all devices have recovered and is equivalent to the resilience time. During blacklisting, devices that become infected before they have received the necessary vulnerability data do not recover and therefore the end time of the epidemic occurs when the peak infection occurs. With additional virtualisation, and assuming the malware cannot escape outside the VM isolation (§7.6.5), individual recovery occurs when a VM is deleted and recreated during the update process. The device however does not become immune, only re-susceptible and therefore the time to the end of the epidemic is measured when the infected state reaches a minimum after the initial infection has started (such as that shown in Figure 8-20 (b)). When there is no diversity ($V_R = 1$) blacklisting and hence VM updating does not occur and therefore devices do not return to the susceptible state, and time in this case is not measured. All times are measured when the system is within one of its final value.



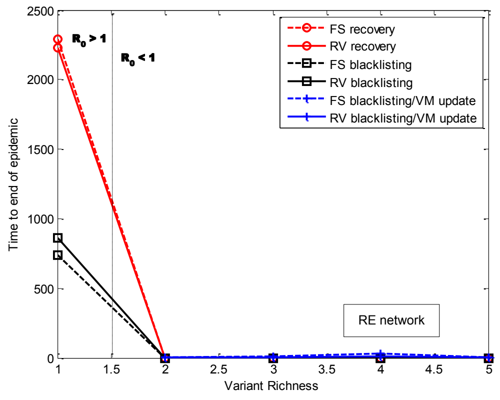
a) 1 locus - Resistance



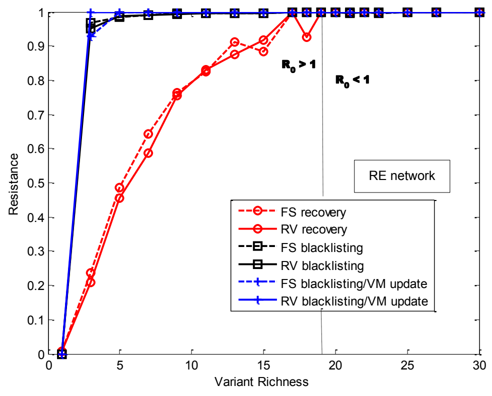
b) 1 locus - time to the end of the epidemic



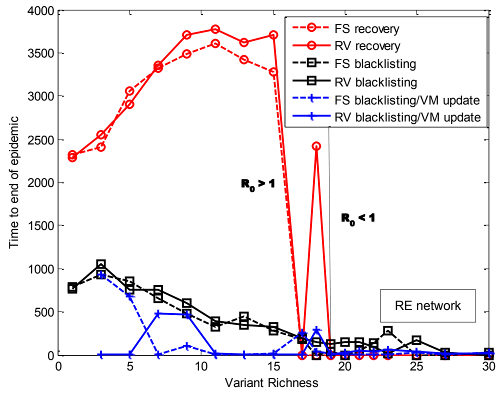
c) 4 locus AND - Resistance



d) 4 locus AND - time to the end of the epidemic



e) 4 locus OR - Resistance



f) 4 locus OR - time to the end of the epidemic

Figure 8-22 – Comparative resistance and epidemic times with different malware

All dashed lines within Figure 8-22 represent the point of mitigation. For the comparatively chosen blacklisting rate (0.004), the resultant resistance is very high for all malware types regardless of the variant richness (as can also be seen in Figure 8-21 (a)). When virtualisation is added to additionally recover those infected resistance is increased further and in the scenario simulated has increased the resistance against all malware types to 100%. As shown by the graphs in Figure 8-22 (b), (d) and (f) the additional mechanisms correspondingly reduce the end time of the epidemic.

8.3.5 Malware Attack in an RE Network with Constraints

This section analyses the effects of diversity on the resistance and resilience to malware when different constraints are applied. Firstly the effects of generic constraints are considered for both the dynamic algorithms in relation to the diversity performance analysed in 8.3.2.2. Secondly the effects of user desirabilities and software compatibility on malware resistance and peak infection times are considered for the FS diversity algorithm.

8.3.5.1 Constraints with Single Locus Malware and Recovery (SIR)

Figure 8-23 shows what happens to malware resistance and resilience when the generic constraints are applied (§8.3.2.2), where ¼, ½, and ¾ of devices are limited to one variant. The resultant variant diversity achieved by the FS and RV algorithms for such constraints were previously shown in Figure 8-16 where the FS algorithm is able to achieve a higher variant diversity level for small variant richness values. This result is reflected in the achieved resistance against malware as shown in Figure

8-23 where the FS algorithm is able achieve a higher resistance for small variant richness values. For this scenario recovery mechanisms are in place where malware mitigation for the one locus model ($V_R = 5$) is indicated by a dashed line in Figure 8-23 (a). When ¼ are constrained to the same variant the susceptibility becomes $S_p = \left(\frac{3}{4} \times \frac{1}{V_{R1}}\right) + (\frac{1}{4} \times \frac{1}{V_{R2}})$, where V_{R2} is fixed at one since there is only one variant and V_{R1} can vary along the variant richness axis. When $V_{R1}$is set very large the first term approximates zero so that $S_p \rightarrow 0.25$, meaning that the mitigation susceptibility of 0.2 can never be reached. The resistance therefore levels out as variant richness is increased, and so too does the resilience as shown in Figure 8-23 (b).

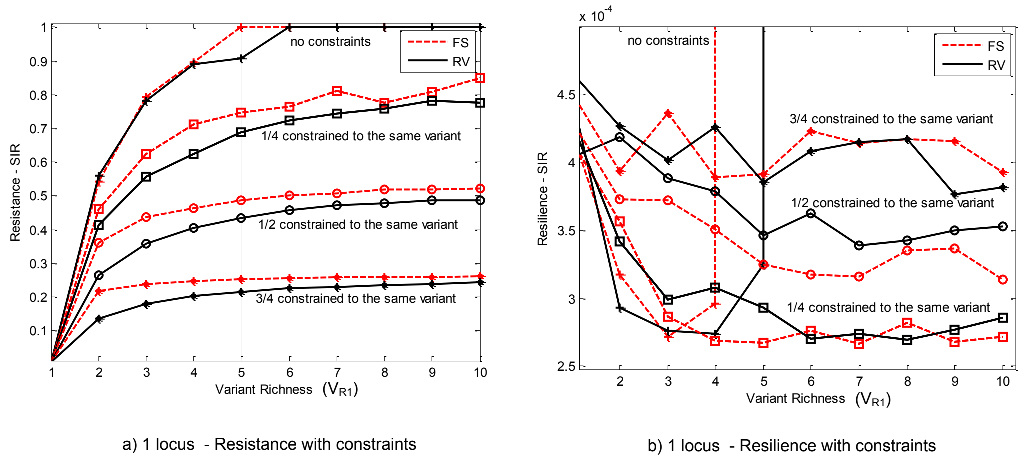


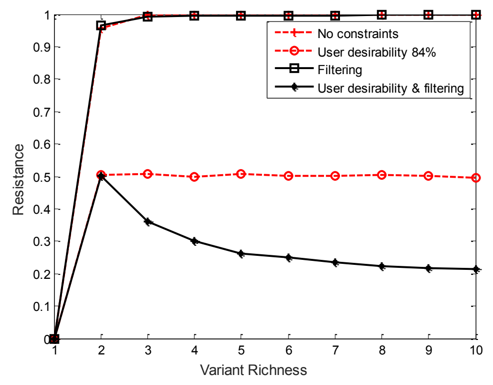
Figure 8-23 – One locus SIR with generic constraints

The maximum fraction of constrained devices that can be tolerated to enable malware mitigation is 1/5th which will occur when V_{R1} is large where $S_p \rightarrow 0.2$, otherwise the speed of recovery would need to be increased to improve the γ/β_n ratio. In a practical system therefore consideration needs to be given to devices that are not participating in the diversity scheme and subsequently become vulnerable, since it may not be possible to mitigate the malware in the network through increasing the variant richness of the remaining devices without improving recovery times.

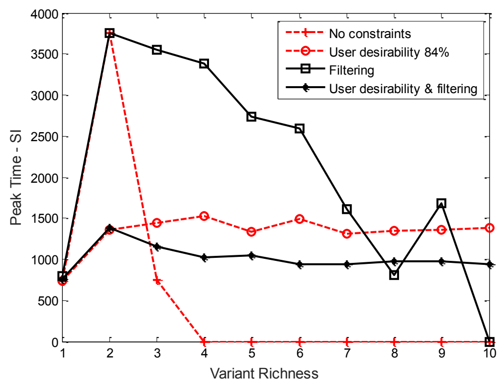
8.3.5.2 FS Constraints with Multi Locus Malware

The distributed FS algorithm is able to model additional multi-locus constraints including user desirabilities and software compatibility across loci (§7.6.2.2). Results are shown in Figure 8-24 for the four locus AND and OR malware types when there are no additional security mechanisms in place and devices have stopped updating. The resistance and peak infection times are shown for four scenarios: 1. Without constraints, 2. With user desirability constraints, 3. With software compatibility filtering, and 4. With user desirabilities and compatibility filtering. β_n is fixed at 0.02 as previous examples.

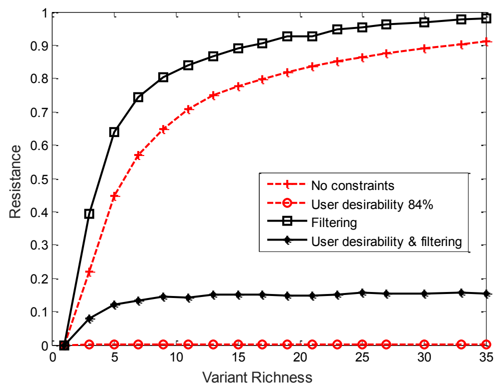
User desirability constraints can be set based upon the data from real networks if it is available to gain an understanding with regard to their vulnerability to different malware types relative to a network with maximum diversity. Here constraints are set at the community scale to represent a plausible scenario where a favoured variant from each locus has 84% usage, equivalent to a market share held by Android during the first quarter of 2016 as reported by Gartner [264]. The remaining variants are favoured with equal probability. The malware is set so that the favoured variant at each locus is targeted by an exploit. The resultant effect of the four locus AND malware type on resistance is shown in Figure 8-24 (a) where it is reduced to 0.5 from almost 1 with no constraints, and correspondingly reducing the peak infection time as shown in Figure 8-24 (b).



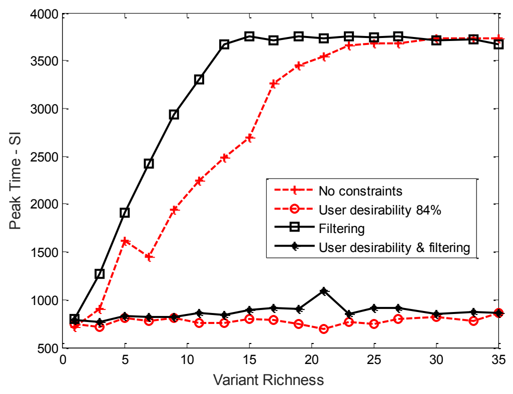
a) 4 locus AND - Resistance with constraints



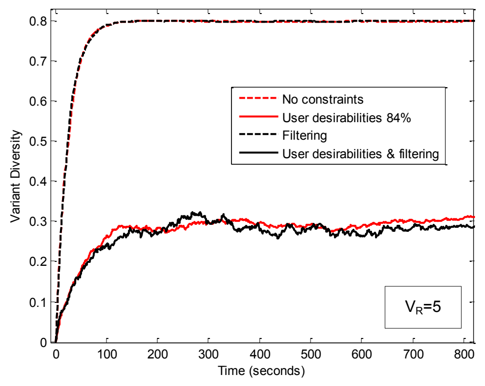
b) 4 locus AND - Peak Infection Time



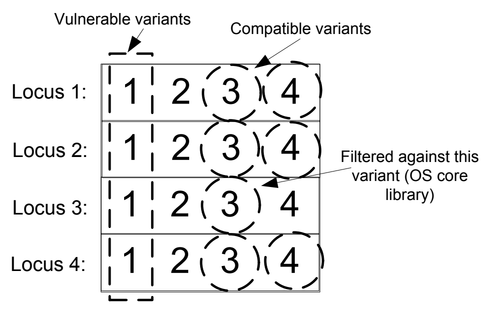
c) 4 locus OR - Resistance with constraints



d) 4 locus OR - Peak Infection Time



e) Variant Diversity under constraints



f) The effect of software compatibility filtering

Figure 8-24 – Four locus malware with FS algorithm specific constraints (SI)

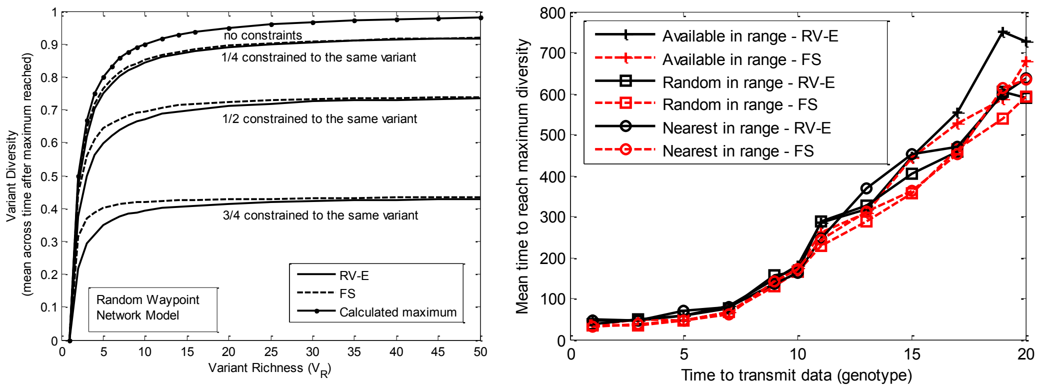
This can be explained mathematically since without any additional security mechanisms the resistance for an SI model is $N_n - N_s = 1 - S_p$ (Equation (6-18)) since all those susceptible become infected. Equation (6-12) defines the susceptibility for the four locus AND malware type where $P(j) = 0.84$, resulting in $S_p = 0.5$. In this constrained scenario the susceptibility value is the same regardless of the number of variants (greater than one) since the same variants are always susceptible by the same percentage, hence giving the flat line. The resultant effect of the four locus OR malware type on resistance and peak infection time is shown in Figure 8-24 (c) and (d) respectively where the four locus OR susceptibility Equation (A-4) results in $S_p = 1.0$ hence the measured resistance is zero, also with a flat line across all variant richness values.

Software compatibility filtering is constrained such that each OS core library variant is compatible with only two variants at each of the other loci. Variants are chosen such that the compatible variant number is the same in each of the other loci, plus the next one, if it exists, otherwise it is wrapped around as shown in Figure 8-24 (f). This configuration is representative of software that is dependent upon other software at different layers with some overlap that may occur through compatibility with closely related versions of the same program. The resultant effect is a reduction in the number of genotypes, however the variant diversity remains maximised as shown in Figure 8-24 (e) (filtering) since the variants themselves remain equally distributed (§5.3.2). As shown in Figure 8-24 (a) the effect of compatibility filtering has reduced the resistance against the AND type only when user desirabilities are additionally assigned, but interestingly has increased the resistance against the OR type as shown in Figure 8-24 (c) when used both stand alone and with user desirabilities. This is because the act of forcing only specific variants across loci to be compatible introduces an AND relationship in the genotypes. For the OR case the resulting reduction in genotypes from filtering means that a higher proportion of the genotypes available to use are without a vulnerable variant. For the AND case little difference is seen since there is already an AND relationship across the loci, unless the user desirabilities are set, where the resistance reduces because a higher proportion of devices (84%) has the vulnerable genotype (variant 1 in each locus). This results in the amplification effect (§2.2.2.2.3) where the representation of those vulnerable is increased. Although only one specific example is given here for compatibility filtering, it highlights its effect on constraining genotypes and introducing an AND relationship for the OR malware type which could occur for any combination of compatibility filtering across loci, although the exact result would vary depending upon the filter applied.

8.3.6 Dynamic Diversity Performance with the RWP Network Model

For the RWP network model, when devices move around randomly in a closed space the resultant diversity and resistance relationships that appear at the network level are similar to those of the RE model. This is because the random movement of devices results in all devices eventually coming into contact with each other. Differences arise in terms of time scales, where for the RWP model, the contact rate is determined by several parameters (§7.3.2). Additionally it is possible to introduce spatial effects into the RWP model that the RE model cannot analyse.

For example when generic constraints (¼, ½, and ¾ of devices are constrained to the same variant) are applied to a random selection of devices, a similar relationship is observed between the RWP as shown in Figure 8-25 (a) and the RE network (Figure 8-16 (d) §8.3.2.2) models. Simulations were run with 1000 devices, and a fixed FS buffer size of 10 genotypes as per previous results (§8.3.2) where the buffer size represents the most recently encountered devices and their configurations. The device selection method chosen is *random in range* comparative to the RE network model, although the selection method does not affect the final variant diversity obtained. Updates occurred every encounter once the data buffer was full, together with a genotype time out period of 10 seconds and a 1 second genotype data transmission window so that the maximum variant diversity was achieved as fast as possible. For randomly moving devices, their average speed, transmission range, and bounded area does not affect the final variant diversity obtained, only the time at which it is achieved. It is difficult to compare the two network models directly since the successful contact rate of the RWP model is determined by these additional parameters, rather than being specified directly. A fixed wireless transmission range of 10m equivalent to a standard Bluetooth connection [3], a bounded area of 600m by 600m (the size of a large park, or small campus facility), and an average walking speed of 1.4ms^{-1} was modelled. The mean time to reach maximum diversity when there are no constraints was measured for each of the three device selection methods (available in range, random in range, nearest in range) as the time required to transmit data between devices was varied. The result is shown in Figure 8-25 (b), where for high data transmission times the *available in range* selection method is shown to marginally (in this scenario) take longer to reach maximum diversity since the devices have to wait longer between communications.



a) Random Waypoint network model (RWP): Measured variant diversity in a constrained scenario, when constraints are applied to random devices

b) Random Waypoint network model (RWP): Measured time to reach maximum diversity without constraints for different device selection methods as the time to transmit data between devices is varied

Figure 8-25 – Random waypoint variant diversity relationships

With the RWP model, additional spatial effects that are likely to occur in practice can be included that are not modelled, or cannot be visualised with the RE model such as spatially located constraints affecting spatial diversity patterns and the distance and speed at which malware spreads. With the FS algorithm it is possible to pictorially visualise the local diversity as measured by each device, and which variants are chosen, and where, as the simulation runs. Figure 8-26 shows the time evolution of the *variant diversity* of a one locus model as it is being maximised from a starting point where there is no diversity by the FS algorithm. The network is confined to an area of 30m by 30m, equivalent to a medium sized conference room or sports hall. Additionally shown is the *local diversity* measured by 50 devices, along with the selected *variant* at each device for a fixed variant richness of five. Two local perspectives are given for a point in time when firstly the variant diversity is being maximised but still at a low level, and secondly when maximisation has been reached and devices have stopped updating. The transmission range was fixed at 10m, the time to successfully transmit genotype data was set at two seconds and the devices were limited to a slow walking pace of 1ms^{-1}. As shown, when the network variant diversity is low, the locally measured diversity by the devices is also low, with some devices measuring 0.2, and others 0.4, with a majority of the devices operating with variant 1. When variant diversity of the network is at its maximum level, the local variant diversity measured by each device is also at a maximum where in the majority of cases different variants are located adjacent to each other.

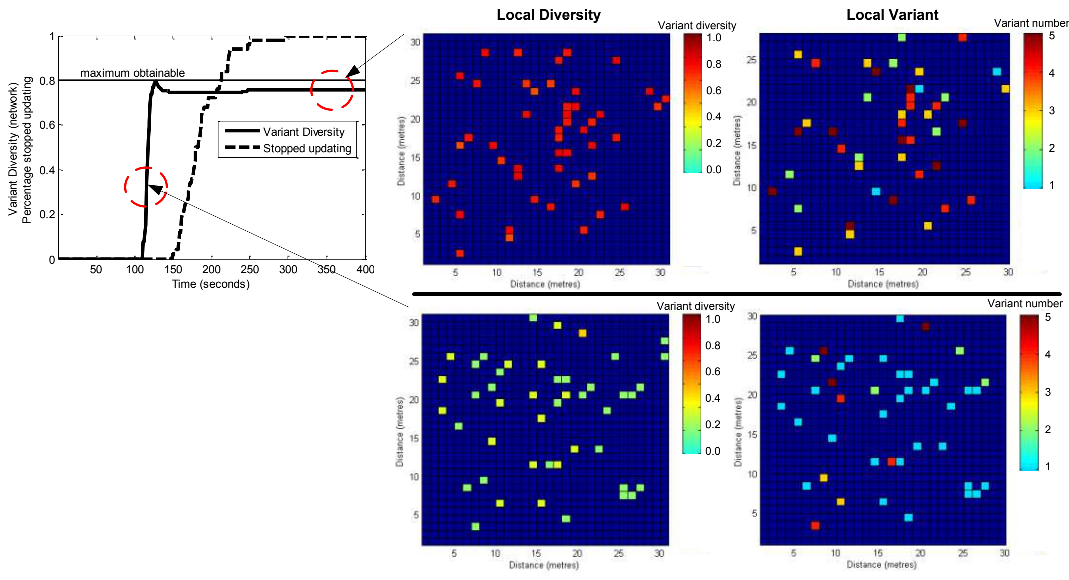


Figure 8-26 – Dynamic diversity performance: random waypoint

8.3.7 Spatial Constraints with Multi Locus Malware (RWP)

As discussed previously the RWP network model can be used to analyse effects such as spatial constraints which are likely in a practical ad hoc network. Three scenarios are modelled: Malware attacks with 1) no constraints, 2) randomly placed generic constraints, and 3) location based generic constraints (room or building). The scenarios are simulated and compared for both moving and stationary devices, and for single locus, four locus AND, and four locus OR malware types. No additional security mechanisms are applied. Due to simulation times, in the order of a day per simulation set, a small network size of 100 devices was modelled with a genotype and virus transmission time of 1 second and averaged over 100 runs for each data point.

The two generically constrained scenarios are pictured in Figure 8-27, where (a) shows the random assignment of constraints, and (b) shows the location based constraints (such a room or office). Constraints are applied such that ¼ of the devices are constrained to using the same single variant at each locus, equivalent to devices using the same software stack. The location constrained devices are bounded spatially to an area ¼ the size of the simulation area, whilst those remaining are free to move or be positioned anywhere in the bounded simulation space. This could be representative of a work place with devices that have no diversity, surrounded by devices that employ the dynamic diversity scheme, some of which may also enter the work place and then leave again, for example customers visiting a shop, a tourist attraction, or a public service. The network is simulated until it becomes maximally diverse and the devices have stopped updating before malware is released. The source of malware is modelled so that it is always initiated from the device closest to the origin. The resistance to malware is measured, along with the peak infection times, and the average distance the malware travelled from its origin.

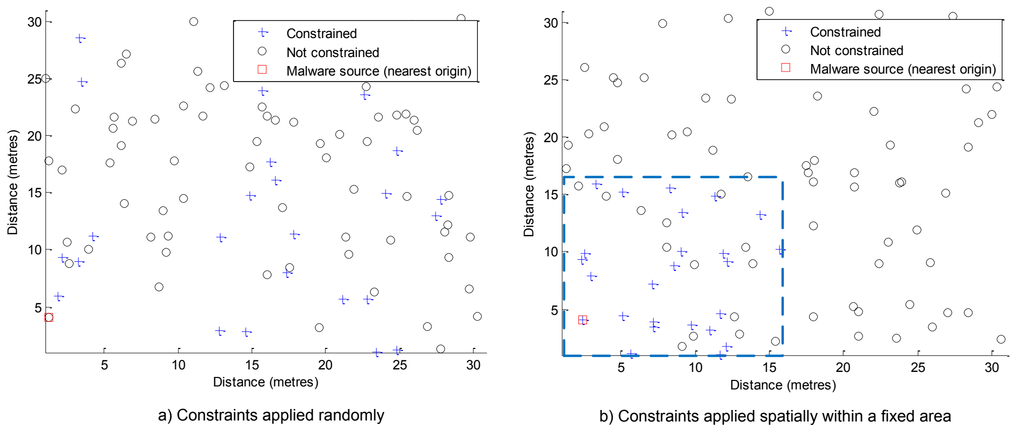


Figure 8-27 – Random waypoint constraints

Results are pictured in Figure 8-28 and Figure 8-29. For moving devices, constraints reduce the resistance to malware as expected (§8.3.5) and as is shown in Figure 8-28 (a) with no differing effect between randomly placed and location placed constraints. This is because eventually all susceptible devices come into contact and become infected. However, the peak infection times as shown in Figure 8-28 (b) differ as a result of the locality of constraints relative to the source of malware. For all malware types the room-constrained scenario results in a faster infection time than randomly placed constraints since those vulnerable are more concentrated near the source of the outbreak. This would mean that a faster recovery time would be required to achieve the same malware mitigation V_R point through variant richness. The AND malware type reaches its peak the fastest since this has the least susceptibility, followed by the one locus, and then the OR type. Increasing diversity in terms of variant richness has less effect on the peak infection time for the constrained scenarios since the constraints create a minimum bound on the number susceptible, and hence time to infect them. Figure 8-28 (c) shows the distance travelled by the malware from its origin. For moving devices, the distance travelled is skewed by their random movement such that for a single infected device, its final location reflects the distance even if it has not infected any other devices.

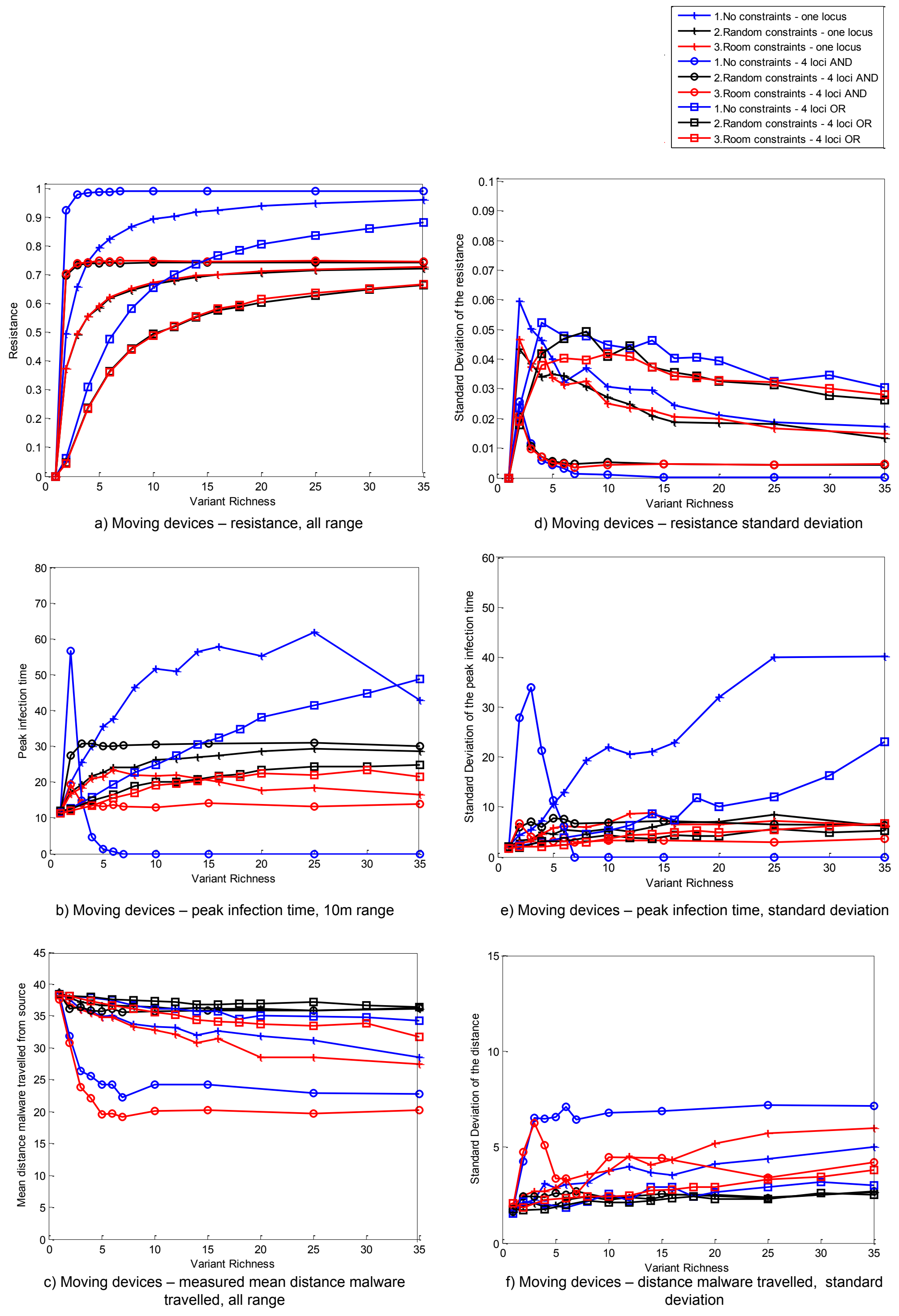
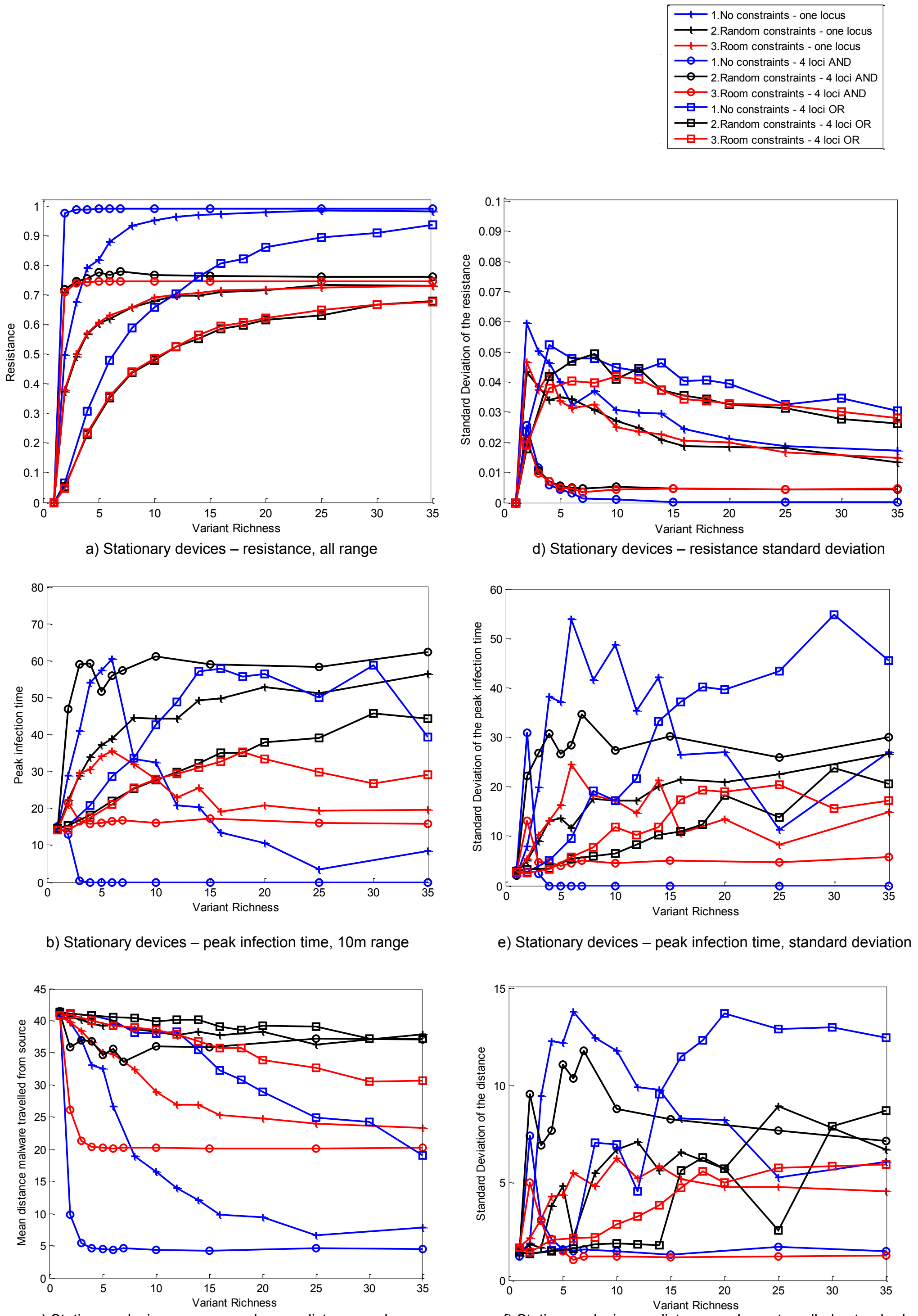


Figure 8-28 – Effects of spatial constraints on malware types – moving devices



a) Stationary devices – resistance, all range

d) Stationary devices – resistance standard deviation

b) Stationary devices – peak infection time, 10m range

e) Stationary devices – peak infection time, standard deviation

c) Stationary devices – measured mean distance malware travelled, all range

f) Stationary devices – distance malware travelled, standard deviation

Figure 8-29 – Effects of spatial constraints on malware types – stationary devices

Malware travels furthest for the randomly placed constraints since they start and remain spread out, and increasing variant richness only causes a small reduction in its distance. Room placed constraints result in the least travelled malware with an increasing variant richness reducing the distance further, particularly for the AND malware type which was reduced to approximately the diagonal length of the room indicating it had been spatially contained for only a handful of variants.

For stationary devices, the transmission range relative to the density and location of those susceptible impacts upon the resistance, the peak infection time, and the distance malware travels. For small transmission ranges, such as 1m, the malware does not travel at all even when there is no diversity since devices are too far apart to communicate (not shown). As the transmission range increases, the network becomes less resistant and malware travels further on average. When the transmission range is set at 10m, the resistance relationship with diversity is comparable to the moving case as shown in Figure 8-29 (a) and Figure 8-28 (a). However the peak infection times as shown in Figure 8-29 (b) are higher and there is a larger difference between the three constrained scenarios. Stationary devices therefore can tolerate longer recovery times to achieve the same V_R mitigation point. For stationary devices malware travels the least when there are no constraints (as expected), and the most for randomly placed constraints as shown in Figure 8-29 (c). Similar to the moving devices result, the four locus AND malware type is shown to be confined within the constrained area by relatively few variants. Further increases in variant richness will eventually confine the one locus and the four locus OR malware types preventing further spatial spread.

8.3.8 Section Summary

This second chapter section detailed the results of the agent-based diverse system model. In constrained static diversity mode, the agent model performs as a stochastic epidemic model where there may be differences in the measured malware mitigation point from the approximated or deterministic result due to the rate of change of the variant richness versus susceptibility relationship for specific malware types. Static diversity however can lead to targeted attacks on specific devices if their configurations remain fixed and become known. Also real devices do not move in random patterns and therefore random static assignment of software may not be the best distribution for a particular scenario where there may be continuous changes in the network topology or influences from constraints.

Incorporating dynamic diversity allows software stacks to be changed in response to network conditions, new information, or to confuse a targeted attacker. The fastest way to achieve maximum diversity from a starting point of no diversity is to perform update decisions at every successful encounter. The FS algorithm can be faster than the RV algorithm at reaching maximum variant diversity for the same input conditions, and can also achieve a higher diversity level when practical constraints are applied and few variants are available, which may be likely in practice. This is reflected in the amount of resistance provided by the two algorithms during a malware attack, with the RV algorithm requiring a faster recovery rate to achieve the same variant richness mitigation point. If any malware type is released during continuous updating of genotypes (from the same pool) without security measures, eventually all devices become infected resulting in no resistance. The time to reach the peak infection however in comparison to no diversity is lengthened because of dynamic updating which is dependent on either reducing the update rate or increasing the variant richness. Without any mechanism for recovery the optimum solution to preserve resistance against malware is to maximise diversity as quickly as possible and then remain static for as long as possible. When recovery mechanisms are applied at a fixed rate, the same mitigation point is observed for both static and continuous dynamic diversity for each malware type, meaning constantly changing configurations, for example to confuse a targeted attacker can be tolerated if the variant richness is high enough. Operating with a variant richness that is below the mitigation point may give a lower resistance for continuous updating depending upon the malware type and relative recovery rate.

As well as responding to changing network conditions, dynamic diversity allows integration with potentially more effective security mechanisms that can be applied sooner than antivirus signatures or patching. When compared to recovery mechanisms, blacklisting can be more effective at increasing resistance and reducing the duration of the epidemic, even at the same rate since it can be applied before an infection occurs. Similar to the recovery mechanism there is a trade off between dissemination speeds, diversity and resistance. When a virtualisation platform is added where infections are removed during blacklisting updates, resistance is increased even further. In the scenario simulated this resulted in the mitigation of all malware types with an exploit richness of one as soon as a second variant became available at each locus.

Desirability constraints can be set for the FS algorithm to reflect the diversity of real networks. An example has been shown to reflect the market share of the currently dominating mobile phone operating system and its impact on the resistance to the different malware types comparative to maximum diversity. For an 84% dominance at each locus of the vulnerable variant, the resistance against the four locus AND type is halved, and the resistance against the four locus OR type is reduced to zero, rendering any remaining diversity ineffective. Introducing software compatibility to reflect problems arising from the use of COTS software as diverse variants can result in a reduction of the number of available genotypes, whilst maintaining variant diversity. This can have the effect of introducing an AND relationship across loci, reducing the number of genotypes with vulnerable variants, and increasing the resistance against the OR malware type. For the AND malware type there is already an AND relationship so this has little effect, unless the genotypes are not equally distributed where the vulnerable genotype has a greater representation resulting in the amplification affect and reduced resistance.

Introducing spatially located constraints modelled by the RWP network model can change peak infection times, and the average distance travelled by different malware types, in comparison to randomly located constraints. Increasing variant richness in the remaining unconstrained network can contain the spread of the malware such as preventing its spread beyond a vulnerable office with little diversity. Differences in peak infection times however will require differing rates of recovery to achieve mitigation of malware for a given variant richness.

8.4 Summary

The results from two different, but closely related, diverse system models have been analysed. Under static diversity conditions, and within an ideal scenario, the epidemic model can predict the diversity requirement needed to tolerate or mitigate specific types of malware attacks. The agent model can simulate the same conditions as the epidemic model (albeit with much longer simulation times) and subject to some differences in the results due to its stochastic nature. When recovery is included it resembles the stochastic epidemic model. Predictions from the epidemic model can be used by the agent model to make comparative measures against dynamic diversity algorithms, practically constrained scenarios, or the inclusion of additional security mechanisms.

The combined results show that resistance to multi-locus malware within an ad hoc network ecosystem can be improved by maximising variant diversity and increasing the number of variants at each locus, with the additional effect of lengthening the time at which the peak infection is reached. When recovery mechanisms are in place there is a trade off between optimising diversity, maintaining quality of service or mitigation, and the speed of recovery. The exact diversity requirements can be calculated by the epidemic model or simulated by the agent model. The multi-locus OR malware type poses a higher security risk than the AND malware type and consequently requires considerably more software variants to tolerate or mitigate against the malware for the same number of exploits.

The results confirm the hypothesis (§1.2) that incorporating biodiversity concepts within computer networks can make them more resistant to cyber security attacks.

When diversity becomes dynamic and integrated with other security mechanisms it can become even more effective. When compared to recovery mechanisms for example, blacklisting can be more effective at increasing resistance and reducing the duration of the epidemic since it can be applied before an infection occurs, helping to alleviate surges in attacks from newly disclosed vulnerabilities.

Simulating constrained scenarios can aid in understanding the impact of diversity on current networks, or where practical limitations may affect the outcome. Software compatibility, for example, may be beneficial in increasing the resistance against the multi-locus OR malware type due to the effect of introducing an AND relationship across loci. Spatially located constraints modelled through the RWP network model have shown that diversity can be used to contain malware outbreaks to local areas, when there is both very little diversity and susceptible configurations within these areas.

Chapter 9

Conclusion and Future Work

9.1 Introduction

This chapter is split into two sections. The first section summarises the motivation for the work performed, the research carried out, and the conclusions from the results and analysis. The second section outlines some practical limitations of the models and provides some suggestions for further work.

9.2 Conclusion

9.2.1 Motivation

Motivation behind the work in this thesis has been inspired through a number of topics including malware epidemics exasperated through monoculture software and criminals responding faster to new vulnerabilities, multiple exploits targeting different layers of the software stack, the benefits and relationships of biodiversity in natural ecosystems, and future trends in the growth of ad hoc networks and peer-to-peer connections.

Although diversity for cyber security is already considered as a beneficial mechanism, it has yet to be fully quantified. A diverse system model incorporating ecosystem concepts, the modularity of software stacks, potential diversity enabling technologies, and practical constraints, together with cross layer multiple exploit malware propagating within ad hoc networks, provides a method for analysing the benefits of diversity and how much is required to tolerate or mitigate specific types of attacks. In creating the model, metrics for the diversity of computing systems have been defined.

9.2.2 Research

Using the ecosystem as a framework, together with the mechanisms that link biodiversity to functionality, relevant analogies were made to define an ad hoc network ecosystem. Although the focus is on ad hoc networks, many of the principles described are also applicable to computer networks in general. The biodiversity functionality is generated by software and hardware components where individualised software stacks are defined as genotypes with multiple loci. Some constraints were applied to limit the size of genotypes and focus on software gene variation, as oppose to software gene functionality, and incorporate malware with cross layer multiple exploits. A threat model has been defined with two types of malware: the logical AND and the logical OR, which are representative of malware using multiple exploits in different ways to gain entry and propagate (chapter 5). The AND and OR types are limited to a single stage logical function, but there may be other types of malware exploit relationships such as both OR and AND across loci requiring multi-stage logic. However, the AND and OR logical functions (together with inversion) form the basic blocks for which all other logical functions can be created (see future work §9.3.1.3).The number of loci was limited to four to both represent one function from each of the broad layers of the software stack (§3.2.1.3), and to correspond with real malware using up to 4 exploits targeting different layers (§3.3.3). This limitation on the number of loci still allows diversity and multi-loci malware concepts to be modelled, when there are no dependencies between functionality such as lower level libraries, however in practice there would be many loci, with differing amounts on different computers with many dependencies.

Two diverse system models have been developed to incorporate software diversity and malware at the genetic level utilising the ad hoc network ecosystem concept. The first is a constrained system level mathematical model, and the second is an agent-based model.

The constrained diverse system model builds upon the traditional mathematical SIR epidemic model and is comprised of a network model, a susceptibility model and a malware model. The mathematical approach is constrained by assuming homogeneous mixing, static diversity, compatible software functions, and non–influential users. Despite these constraints some key mathematical results have been established to investigate the security protection offered by diversity and how much diversity is needed to tolerate or mitigate against specific types of attacks. Additionally the mathematical model provides a stepping stone between, and a method of comparing, an existing *one locus model* to the multiple locus method and the agent model developed here (chapter 6).

The agent-based diverse system model is able to simulate the same conditions as the epidemic model, but additionally incorporates dynamic genotype configuration which can be based on local interaction, user mobility and practical constraints. The dynamic approach is further exploited through the integration with other security mechanisms such as publically available vulnerability data, and virtualisation technology to enhance its effectiveness. This allows the exploration of dynamic diversity and malware propagation beyond the constraints of the epidemic approach. The agent-based approach is comprised of a network model, a diversity model and a malware model. Optimisation of diversity is measured through simulation across a parameter range to either tolerate or mitigate a specific attack. Although simulation times are much longer for this model, its development provides a simulation framework for incorporating additional biodiversity algorithms, network models and malware models, as well as integrating them with other security mechanisms as part of an integrated security approach (chapter 7).

Single measures of diversity in computing systems have been defined in the literature, however several metrics are necessary to define diversity at the genetic level of computing systems such as ad hoc networks, all of which provide a different, but important perspective. The number of software variants at each locus of the software stack, termed variant richness when the quantity is the same, indicates the amount of variation at the locus level. However it is also necessary to understand variant distribution in order to maximise the diversity for a given set of variants and hence a measure of variant diversity is needed. The number of variants and their distribution determines the number of unique genotypes in the network which can be limited by the number of devices. For multiple locus malware, maximising the utilisation of the different genotypes keeps susceptibility at a minimum. Outputs from the two models include the two key components of ecosystem stability: resistance and resilience (chapter 6 and 7).

9.2.3 Results and Analysis

Results from the epidemic and agent models have shown that biodiversity applied within a simulated ad hoc network ecosystem can provide tolerance against multi-locus malware, or provide improved mitigation when recovery mechanisms are in place. This has the overall effect of improving the resistance against such attacks and benefiting cyber defence. The exact diversity requirements needed to tolerate or mitigate malware can be calculated by the epidemic model or simulated by the agent model. Predictions from the epidemic model can additionally be used by the agent model to make comparative measures against dynamic diversity algorithms, practically constrained scenarios, or the inclusion of additional security mechanisms. The results are limited in that they show malware that can only target up to four loci in a single targeted attack. In a real system there may be multiple malware targeting different combinations of loci with varying dependencies.

The epidemic model showed that few software variants are needed to drastically reduce the susceptibility and increase resistance of the overall network, with differences depending upon the type of malware attack. The logical AND malware type with multiple exploits spread across layers of the software stack poses the least risk and can be tolerated or mitigated with very few software variants. The OR malware type poses a higher security risk since the more exploits the malware has available, the greater the overall susceptibility. In a practical system therefore it is not necessary for every device to have a different software variant installed at every locus to adequately tolerate or mitigate different types of malware. For the epidemic model it is possible to quantify through calculation, an optimum diversity, given a specified quality of service that will tolerate, or mitigate an attack for the two different types of malware. The optimum diversity calculated assumes the diversity of the devices remain static (i.e. unchanged) for a period of time equal to or longer than the duration of the epidemic. When recovery mechanisms are in place there is a trade off between optimising diversity, maintaining quality of service or mitigation, and the speed of recovery. The faster the recovery, the higher the tolerated susceptibility and hence less diversity is required. Modelling static diversity can allow epidemic based models to predict the extent of a malware attack under such conditions. Static diversity however can lead to targeted attacks on specific devices. Additionally real devices do not move in random patterns and therefore random static assignment of software may not be the best distribution for a changing network topology with user influences and constraints.

The flexibility of the agent model allows both static and dynamic diversity to be modelled whereby software stacks are able to be dynamically modified in response to changing network conditions, new information, or as a response to additional security mechanisms. However simulation times are much longer, with large networks and a large variant richness or long timescales becoming impractical to simulate. The distributed favourability score diversity algorithm can be beneficial over the random variant algorithm when there are few variants (likely to occur in practice with COTS software) by achieving a higher variant diversity, more quickly under the same input conditions and constraints. This is reflected in a higher resistance provided by the distributed algorithm during a malware attack. The concept of dynamic diversity is based upon the assumption that it is possible to change variants without disrupting the user experience or device operations. It is also a mechanism for confusing a targeted attacker regarding existence of vulnerabilities at a particular device. In continuous updating mode dynamic diversity can linearly extend the peak infection time as the number of variants, or the time between updates, increases, but without recovery or intervention mechanisms in place the entire network can become infected, when the same set of variants are available to every device (e.g. when standard commercial software is available as variants). Static diversity therefore is necessary to maintain long term resistance in the absence of recovery or intervention. For a dynamic scheme it is beneficial to maximise diversity as quickly as possible and then remain static for as long as possible. When recovery is available at a fixed rate, the same mitigation point, in terms of variant richness, is observed for both static and dynamic diversity meaning that constantly changing configurations, for example to confuse a targeted attacker, can be tolerated if the variant richness is high enough. When dynamic diversity is integrated with other security mechanisms it can become even more effective: In comparison with recovery mechanisms for example, blacklisting can be more effective at increasing resistance and reducing the duration of the epidemic, even if at the same rate, since it can be applied before an infection occurs helping to alleviate surges in attacks from newly disclosed vulnerabilities. Similar to the recovery mechanism there is a trade off between dissemination speeds, diversity and resistance. When a virtualisation platform is added to allow infections to be removed during blacklisting updates, resistance is increased even further.

Simulating constrained scenarios can help to understand the diversity impact of current networks, or where practical limitations may affect the overall resistive outcome of a diversity scheme. Software compatibility for example may be beneficial in increasing the resistance against the multi-locus OR malware type due to the effect of introducing an AND relationship across loci. Spatially located constraints modelled by the RWP network model can change peak infection times in comparison to those of a random placement, requiring differing rates of recovery to achieve mitigation for a given variant richness. Additionally, increasing the variant richness of the unconstrained devices in these scenarios can contain malware outbreaks to local areas, such as a vulnerable office, where diversity maybe lower.

The combined results confirm the hypothesis that incorporating biodiversity concepts within ad hoc networks, a form of peer-to-peer mobile wireless network, can make them more resistant to cyber security attacks (chapter 8). The contributions of the research are listed in §1.3.

9.3 Future Work

This section considers a number of parallel avenues that are necessary to take the research further including additional functionality, improved modelling approaches, real world scenarios, and practical considerations.

9.3.1 Additional Functionality

9.3.1.1 Specific to the Mathematical Epidemic Model

The epidemic model is much faster computationally than the agent model in predicting diversity requirements and resistance to different malware types. Remaining statically diverse during a malware epidemic has resistive benefits and modelling such a scenario can be representative of a time window of a dynamic scheme, or a situation where it may not be practical or desirable to keep changing software configurations. However the epidemic model is currently limited in terms of functionality where the specified contact rate alone defines how quickly the malware can spread. In a practical ad hoc network, wireless transmission characteristics and the time taken to transfer malware contribute to a successful contact. These aspects have been incorporated into the agent-based model on top of the random waypoint mobility algorithm, but simulation times are very long. A mathematical network model of moving devices has been incorporated into the deterministic Bluetooth malware epidemic model previously developed by the author [3]. It includes additional states to simulate different malware spreading mechanisms. The research focus was on diversity, but a next step for the epidemic model would be to integrate these aspects together with diversity, multilayer software stacks and logical malware types, as well as the additional security mechanisms such as blacklisting and virtualisation to generate a more functional and representative mathematical model. Blacklisting can be added through the inclusion of an additional state which will have the effect of removing susceptible devices causing them to become temporary immune at a specified rate. Virtualisation causes feedback from the infected state into the susceptible state, and both of these two additional mechanisms will change the dynamical equations of the malware model. This will provide results faster than the agent model with the benefit of being able to simulate networks with a large variant richness over long timescales.

9.3.1.2 Specific to the Agent Model

A benefit of the agent model is in its flexibility to add functionality without the requirement to describe the process mathematically meaning additional concepts that are difficult or impossible to model using the epidemic method can still be incorporated. For example diversity through evolutionary principles (§3.4.3.3) alongside multiple and mutating malware, locally generated variants, or limiting variant choices at each device rather than allowing selection from a pre-existing software pool. This may be feasible when variants are generated as different binaries from the same source code. Modelling malware with the potential to attack the diversity concept could also be investigated.

A simple example has been demonstrated in the results to highlight the effect of user influence on diversity patterns and hence malware resistance. Collected data of computer configurations from a medium sized network could give a more realistic insight into the current diversity of software stacks and their variation at different layers. This data could be fed into the agent-based model to analyse its current resistance to different multi-exploit malware. The concept of blacklisting and resetting of software stacks through virtualisation could be integrated with heuristic methods used by antivirus software of malware detection to provide a model of detection, removal, and temporary immunity. A reflection on the expectation of homogeneous mixing within ad hoc networks and whether this is realistic would depend upon the scenario under which it is being considered. A market, science fair, careers event, conference, or other clearly defined area where devices may move around with users could be considered representative. However in many cases particular patterns of mobility may occur, for example people and cars travelling along repeated routes. This will alter the assumption of random contact into non-random, with some cars or people never coming into local contact with each other at all. It should also be noted however that although homogeneous mixing is not realistic in many cases it does however form a baseline for which many models and research is based. Additionally the random waypoint model has the flexibility to be extended to include other geographically shaped areas of interest with different spatial effects, or modified to incorporate non-random waypoints such as movement patterns of devices travelling between non-random destinations. Alternatively the random waypoint algorithm could be replaced with real waypoint trace data of mobility patterns providing true movement of devices and resulting in more realistic diversity and malware relationships. The modelling approach used to represent an ad hoc network has been from a high level abstract perspective. Lower levels of abstractions are necessary to capture the true dynamics of an ad hoc network and how malware or other types of attack may interfere with operations. It would be beneficial to incorporate routing algorithms, traffic generation models and emulate true multilayer software stacks using event based simulation. This will better inform on the effect of diversity and malware propagation on the measurable quality of service parameters such as throughput, latency, and end user impact. For the simulation scenarios diversity optimisation results were obtained for QoS tolerances between 80% and 95% resulting in a large range in diversity requirement (Figure 8-5). The measurable QoS parameters listed above will aid in the assessment of more realistic QoS requirements in order to define the optimal diversity necessary. Finally, the agent-based model as currently implemented takes a significant time to execute a scenario, in the order of a day to run a small simulation set across a parameter range, with even longer times prevailing when the random waypoint network model is used. More efficient coding techniques or a different language, such as C++, could be used to accelerate processor intensive functions.

9.3.1.3 General Functionality

Recovery mechanisms integrated into the SIR epidemic model assume antivirus or patching occurs after malware infection has already taken place, since this is what happens in a biological system where the recovery rate indicates the average time in which an individual remains unwell before recovering. In a practical computing network patching and antivirus updates now tend to occur at regular intervals regardless of whether a device is infected; if a patch is developed in time recovery could effectively occur whilst some devices are still susceptible, potentially changing the resistance and resilience response to diversity and the point of mitigation. Future work could include the modelling of more realistic patching to account for regular updates. Additionally, in practice not all users patch their software, for example to avoid potential conflicts between components or across a network, and so future work could analyse this aspect and include it in the model.

The way in which malware targets exploits at multiple layers of the software stack is defined by the logical AND and OR types applied as a single stage logical function. The AND and OR logical functions (together with inversion) form the basic blocks for which all other logical functions can be created. The model could be extended to include multi stage logic to model more complex malware exploit functions (§5.3.3) or relationships between layers impacting whether software becomes vulnerable or not (§5.3.1.3).

Both diverse system models are limited such that each device can only utilise one variant per locus at a time leading to monoploid genotypes. In practice, for some users, it may be necessary to have the use of more than one variant from the same locus for example if incompatible software programs are used as variants where access to different files and data are needed. The models could therefore be extended to include multiple variant selections. Finally, the biodiversity concepts explored here are also applicable to other types of computer networks, the exact mechanisms will differ due to how and when connections are made and differences between distributed and centralised architectures. The work focuses on ad hoc networks and is bounded by the characteristics of such a topology (which will inherently be different to other topologies). Ad hoc networks by themselves, for example, do not scale easily due to excessive protocol overhead and tend to be limited to small geographical regions. The lack of scale however does not necessarily have an adverse impact on the effectiveness of a biodiversity scheme. The principle of software variants at different layers of a software stack applies to both small and large networks. The physical separation of different ad hoc networks may actually prove to be advantageous, by helping to contain malware to localised regions. Additionally global connectivity can be achieved through access points to the internet such as in the case of mesh networks (§3.2.3) allowing new variants and vulnerability data to be accessed. Future work could explore the effects on different network models.

9.3.2 Practical Considerations

9.3.2.1 Practical Generation, Dissemination and Storage of Variants

There are still remaining questions regarding what should be made diverse, for example results suggest that only a handful of variants may be necessary at each software layer to mitigate against a single exploit four locus AND malware attack such as Stuxnet. In this case it may be practically viable to have a small number of different software programs available at each stack layer. On the other hand to mitigate a four locus OR malware type of attack, several hundred, or even thousands of variants may be necessary in which case the automated generation of software variants from a limited source of software programs would be required to generate the large volumes of variants required. The practicability in disseminating and maintaining large volumes of different software variants still needs to be addressed. Creation and dissemination methodologies still remain at the concept stage within the literature and further practical trials are needed [179]. For example, there is suggestion that compiler generated software originating from single source code (§3.4.3), potentially conducted within the cloud (§3.2.1.1), could provide the necessary functionally equivalent variants when requested by users [110]. Such concepts could be embedded within current software download areas, for example 'App' stores, requiring no additional user input. This suggests the generation of automated variants could be via a centralised source, but the option of generating variants locally and dynamically from known good raw software sources has largely been unexplored. Within a distributed network such as an ad hoc network it may be advantageous to be able to both download variants at relatively slow timescales, and generate or swap in new ones locally on faster timescales to be able to adapt to local situations. Generating diversity at the compiler stage is only one approach to generating diverse software (§3.4.3), other techniques can be applied after distribution such as during installation, loading or program execution [178]. These techniques for example could be adopted locally. In practice only a small number of variants may be able to be stored locally due to limitations in memory, and therefore techniques for local and dynamic generation within ad hoc networks would be advantageous. If the concept of blacklisting is to be made viable, fast access to alternative variants are necessary such as alternative choices stored locally. Results showed that regular changing of genotypes to confuse a targeted attacker from the same global pool of software could lead to the entire network becoming infected. Additionally, results showed that limiting variant choices through compatibility filtering can actually reduce the spread of malware, and therefore limiting variant choices by each device and self generation of variants should be explored further. When diversity is generated from independent software programs it may result in a greater difficulty of maintenance, a larger range in their quality since the diverse variants both within and across loci may come from many different sources. Even those that do come from the same source such as through the diversification of binary files or memory allocation may differ in terms of efficiency of resources, and speed. The quality however may only become a problem if it starts to noticeably impact the end user. Finally, as with any software distribution method, there is the possibility of variants being generated from unauthenticated sources containing hidden malicious code, and careful management of distribution is also required.

9.3.2.2 Security of Genotype Exchange for the Distributed Algorithm

With the distributed favourability score algorithm where genotype information is exchanged, careful attention is needed towards the secure sharing of information during the communication link set up and the authenticity of the data since ill-informed genotype information could result in variant choices to the advantage of the attacker. As discussed in §7.6.2 In order for the scheme to become practically viable both a discovery protocol to determine software variant information, and a trust model to authenticate and maintain privacy of the genotype information between trustees would need to be developed.

9.3.2.3 Vulnerability Data and Blacklisting

Software vulnerability information is currently stored in publically accessible databases (§3.4.4.6). In order for blacklisting as described within this thesis to become viable, the speed of dissemination will be critical in preventing infections. There is a trade off between the speed of blacklisting and variant richness to obtain the necessary resistance for the specified malware type. Antivirus companies are best placed to assess vulnerability data, and disseminate blacklisting information as a precursor to disseminating updated signature databases to detect and block the malware. This is because they already have the expertise in understanding the vulnerabilities, their perceived threat, and the necessary infrastructure to disseminate the data using authentication and integrity checks during downloads.

Antivirus companies can already block access to certain websites and terminate suspicious activity, and therefore processing vulnerability data would be an additional capability integrated into a multi-layered security approach. Careful consideration will be necessary to the perceived level of threat since it would be undesirable for many variants to be simultaneously blacklisted as this will reduce the diversity of the network, increasing the susceptibility to other potential attacks. Additionally there may be the need for the recipient to be involved in the blacklisting decision process if the inconvenience of an unusable variant out-weighs the risk of infection. The automated dissemination of vulnerability information could be released as soon as it becomes available and would be more timely than the corresponding antivirus signature and software patch, and more importantly potentially faster than the response from cyber attackers.

9.3.2.4 Virtualisation

Dynamically changing software stacks, or parts of software stacks on devices, including through blacklisting can only be viable, if it is physically and succinctly possible to do so without affecting the user experience and interrupting network operations, such as without having to reboot devices. Virtualisation could potentially provide a mechanism for this. Using virtualisation to swap chunks of functionally equivalent software in an attempt to confuse a targeted attacker has been investigated in the literature (§3.4.3.4) highlighting that the concept is possible. However, there are currently practical limitations that need to be overcome before virtualisation technology can be fully used for dynamic diversity. For example in order for a dynamic strategy to utilise virtualisation, a diversity hypervisor or a hypervisor extension supporting diversity would need to be developed. This would need to automate and manage the usage of virtual machines to create, back-up, and destroy genotypes on-the-fly and enable security loop holes to be evaded until a patch is created. Hiding diversity to maintain an adequate user experience from the device requires additional complexity. A possible solution is given in Figure 9-1. Although the bulk of the software would be embedded within the VM Genotypes it is proposed in the first instance that standard clients together with user selected (constrained) software are run on the user VM with what appears to look like relevant proxies on a VM Genotype. These clients would use data protocols with network traffic such as email, and web access. This would allow client software preferred by the user to remain constant whilst allowing the underlying software to be made diverse using VM Genotypes. The proxies would carry out additional security tasks, above normal servicing requests, such as rendering and recoding of the data before returning it back to the client. This would help prevent malicious exploits from penetrating through to the client. Development of proxies capable of supporting the clients would be complex and made difficult by secure network protocols. An alternative solution would be to construct a specific framework of thin or zero client [265] software for the biodiversity scheme to replace the standard set of clients and provide front ends to common applications such as word processing. The thin client would be a translated 'image' of the software running on the VM, similar to those used for cloud computing, except it would be tailored to the user and would remain constant unless changed by the user, regardless of the diverse software running on the VM genotype. The functionality of the standard client would then be encompassed back into the VM genotype giving more control over the diversity of components, with the associated greater security benefits.

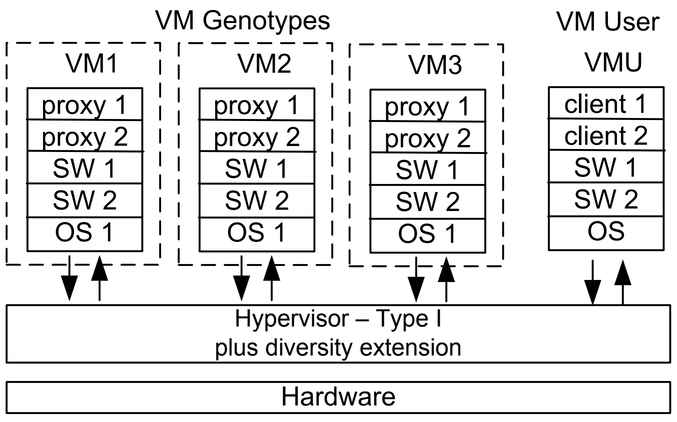


Figure 9-1 - Possible virtualisation architecture to support dynamic diversity

Appendix A

Constrained Diverse System Model: Epidemic Equation Derivations

A.1 Susceptibility Derivation: Three Locus Logical OR Malware Type

Three Loci: For three independent events the probability OR rule (General Addition Rule) can be applied iteratively using the two locus result (D).

$$Let\ D = P(j = 1\ or\ j = 2) \quad = P(j = 1) + P(j = 2) - P(j = 1)P(j = 2) \tag{A-1}$$

$$\begin{aligned} P(or, j = 1\ to\ 3) = P(D\ or\ 3) &= P(D) + P(j = 3) - P(D)P(j = 3) \\ &= P(j = 1) + P(j = 2) + P(j = 3) \\ &\quad - P(j = 1)P(j = 2) \\ &\quad - P(j = 1)P(j = 3) \\ &\quad - P(j = 2)P(j = 3) \\ &\quad + P(j = 1)P(j = 2)P(j \\ &\quad = 3) \end{aligned}$$

And therefore the proportion susceptible S_p for a three locus network becomes

Logical OR type:

(three loci)

$$S_p = \sum_{j=1}^{j=3} P(j) - \prod_{j=1,2} P(j) - \prod_{j=1,3} P(j) - \prod_{j=2,3} P(j) + \prod_{j=1,2,3} P(j) \qquad \text{(A-2)}$$

A.2 Susceptibility Derivation: Four Locus Logical OR Malware Type

Four Loci: For four independent events the probability OR rule (General Addition Rule) can be applied iteratively.

$$Let\ D = P(j = 1\ or\ j = 2) \qquad \text{(A-3)}$$

$$= P(j = 1) + P(j = 2) - P(j = 1)P(j = 2)$$

$$and$$

$$let\ E = P(or, j = 1\ to\ 3) = P(D\ or\ 3)$$

$$= P(D) + P(j = 3) - P(D)P(j = 3)$$

$$then\ P(or, j = 1\ to\ 4)$$

$$= P(E\ or\ 4)$$

$$= P(E) + P(j = 4) - P(E)P(j = 4)$$

$$= [P(D) + P(j = 3) - P(D)P(j = 3)] + P(j = 4)$$

$$-[P(D) + P(j = 3) - P(D)P(j = 3)]P(j = 4)$$

$$= P(D)$$

$$-P(D)P(j = 3)$$

$$-P(D)P(j = 4)$$

$$+P(D)P(j = 3)P(j = 4)$$

$$+P(j = 3) + P(j = 4) - P(j = 3)P(j = 4)$$

$$= P(j = 1) + P(j = 2) - P(j = 1)P(j = 2)$$

$$-[P(j = 1) + P(j = 2) - P(j = 1)P(j = 2)]P(j = 3)$$

$$-[P(j = 1) + P(j = 2) - P(j = 1)P(j = 2)]P(j = 4)$$

$$+[P(j = 1) + P(j = 2) - P(j = 1)P(j = 2)]P(j = 3)P(j = 4)$$

$$+P(j = 3) + P(j = 4) - P(j = 3)P(j = 4)$$

Therefore the proportion susceptible S_p for a four locus network becomes:

Logical OR type: (four loci)

$$\begin{aligned} S_p = \sum_{j=1}^{j=4} P(j) - \prod_{j=1,2} P(j) \\ - \prod_{j=1,3} P(j) - \prod_{j=2,3} P(j) \\ + \prod_{j=1,2,3} P(j) - \prod_{j=1,4} P(j) \\ - \prod_{j=2,4} P(j) + \prod_{j=1,2,4} P(j) \\ + \prod_{j=1,3,4} P(j) + \prod_{j=2,3,4} P(j) \\ - \prod_{j=1,2,3,4} P(j) - \prod_{j=3,4} P(j) \end{aligned} \tag{A-4}$$

A.3 Diversity Optimisation Derivation: Three Locus OR Malware Type

General Equation:

Using Equation (A-2) for 3 loci where

$$S_p \leq \sum_{j=1}^{j=3} P(j)_{max} - \prod_{j=1,2} P(j)_{max} - \prod_{j=1,3} P(j)_{max} - \prod_{j=2,3} P(j)_{max} + \prod_{j=1,2,3} P(j)_{max} \tag{A-5}$$

$$= \left[\frac{x_1}{v_1} + \frac{x_2}{v_2} + \frac{x_3}{v_3}\right] - \left[\frac{x_1 x_2}{v_1 v_2}\right] - \left[\frac{x_1 x_3}{v_1 v_3}\right] - \left[\frac{x_2 x_3}{v_2 v_3}\right] + \left[\frac{x_1 x_2 x_3}{v_1 v_2 v_3}\right]$$

For a given x_1, x_2, x_3 value, and S_p, v_3 can be solved computationally for a range of v_1 and v_2 values to find the valid solutions satisfying the minimum *variant richness* (V_R). Where the valid solutions are positive and non imaginary, and must be greater than the number of exploits.

Average Equation:

Variant Richness (average number of variants for an average number of exploits): Assuming the number of exploits and variants are the same in each loci, the equation can be simplified to:

$$S_p \leq \frac{3x}{v} - \frac{3x^2}{v^2} + \frac{x^3}{v^3} \tag{A-6}$$

Which can be solved numerically for v.

A.4 Diversity Optimisation Derivation: Four Locus OR Malware Type

General Equation:

Using Equation (A-4) for 4 loci where

$$S_p \leq \sum_{j=1}^{j=4} P(j)_{max} - \prod_{j=1,2} P(j)_{max} - \prod_{j=1,3} P(j)_{max} - \prod_{j=2,3} P(j)_{max} + \prod_{j=1,2,3} P(j)_{max} - \prod_{j=1,4} P(j)_{max} - \prod_{j=2,4} P(j)_{max} + \prod_{j=1,2,4} P(j)_{max} + \prod_{j=1,3,4} P(j)_{max} + \prod_{j=2,3,4} P(j)_{max} - \prod_{j=1,2,3,4} P(j)_{max} - \prod_{j=3,4} P(j)_{max} \tag{A-7}$$

$$= \left[\frac{x_1}{v_1} + \frac{x_2}{v_2} + \frac{x_3}{v_3} + \frac{x_4}{v_4}\right] - \left[\frac{x_1 x_2}{v_1 v_2}\right] - \left[\frac{x_1 x_3}{v_1 v_3}\right] - \left[\frac{x_2 x_3}{v_2 v_3}\right] + \left[\frac{x_1 x_2 x_3}{v_1 v_2 v_3}\right] - \left[\frac{x_1 x_4}{v_1 v_4}\right] - \left[\frac{x_2 x_4}{v_2 v_4}\right] + \left[\frac{x_1 x_2 x_4}{v_1 v_2 v_4}\right] + \left[\frac{x_1 x_3 x_4}{v_1 v_3 v_4}\right] + \left[\frac{x_2 x_3 x_4}{v_2 v_3 v_4}\right] - \left[\frac{x_1 x_2 x_3 x_4}{v_1 v_2 v_3 v_4}\right] - \left[\frac{x_3 x_4}{v_3 v_4}\right]$$

For a given x_1, x_2, x_3, x_4 value, and S_p, v_4 can be solved numerically for a range of v_1, v_2, and v_3 values to find the valid solutions satisfying the minimum *variant richness* (V_R). Where the valid solutions are positive and non imaginary, and must be greater than the number of exploits.

Average Equation:

Variant Richness (average number of variants for an average number of exploits): Assuming the number of exploits and variants are the same in each locus, the equation can be simplified to:

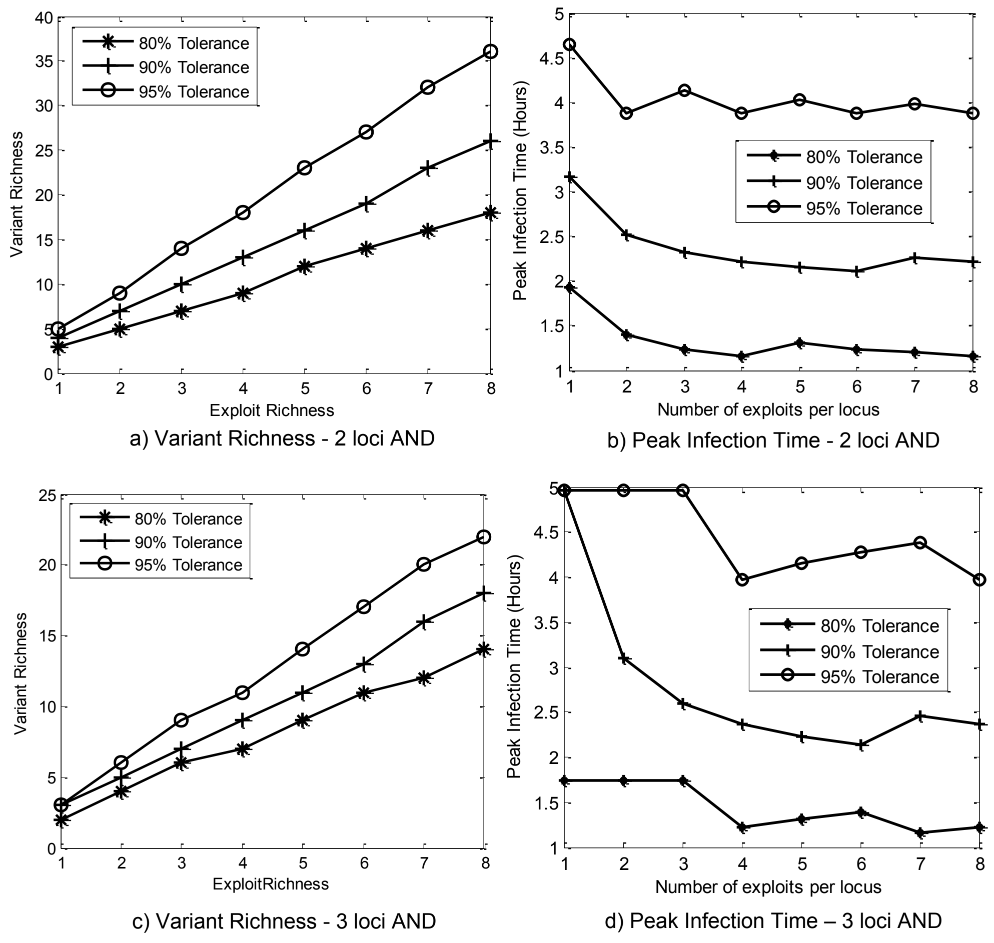
$$S_p \leq \frac{4x}{v} - \frac{6x^2}{v^2} + \frac{4x^3}{v^3} - \frac{x^4}{v^4} \qquad \text{(A-8)}$$

Which can be solved numerically for v.

Appendix B

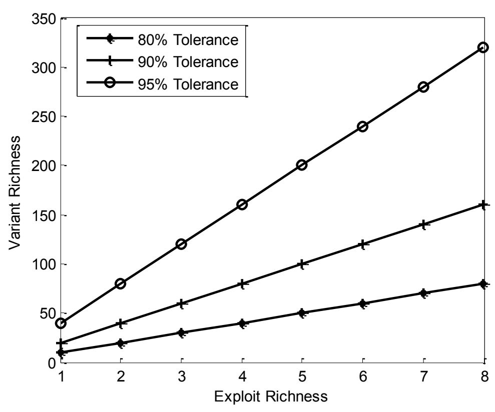
Optimum Diversity and Peak Infection Times for Two and Three Loci

B.1 AND Malware Type

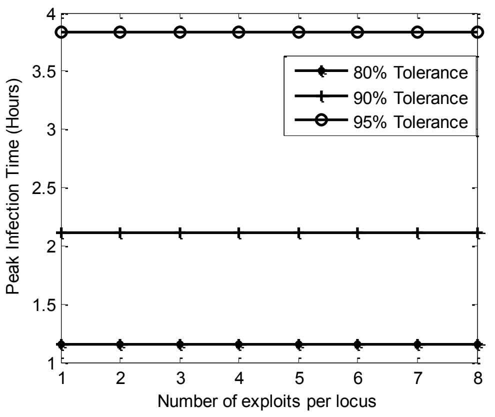


Appendix Figure B-1 – AND malware type

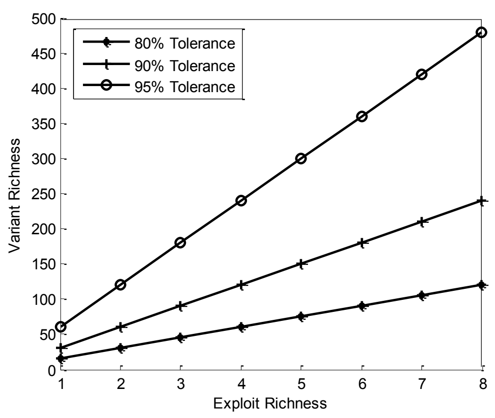
B.2 OR Malware Type



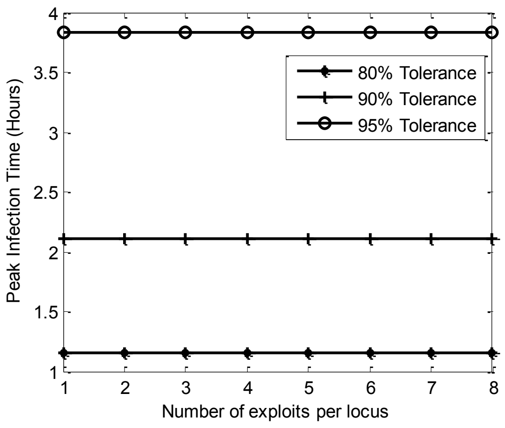
a) Variant Richness - 2 loci OR



b) Peak Infection Time – 2 loci OR



c) Variant Richness - 3 loci OR



d) Peak Infection Time – 3 loci OR type

Appendix Figure B-2 – OR malware type

Abbreviations

3G	Third Generation
4G	Fourth Generation
5G	Fifth Generation
ABM	Agent-Based Models
AMD	Advanced Micro Devices, Inc
API	Application Program Interface
ASLR	Address Space Layout Randomisation
BYOD	Bring Your Own Devices
CODEC	COder-DECoder
COTS	Commercial Off The Shelf
CVE	Common Vulnerabilities and Exposures
D2D	Device-to-Device
DNA	DeoxyriboNucleic Acid
DNS	Domain Name Server
DSR	Data Space Randomisation
DSU	Dynamic Software Updating
FIFO	First-In First-Out
FS	Favourability Score
GIMP	GNU Image Manipulation Program
GNOME	GNU Network Object Model Environment
GNU	GNU's Not Unix!
GPU	Graphics Accelerator Card
GTK+	Object-oriented GIMP ToolKit
GUI	Graphical User Interface
ICMP	Internet Control Message Protocol
IoT	Internet of Things
IP	Internet Protocol
ISR	Instruction Set Randomisation
IT	Information Technology
KDE	K Desktop Environment
LTE	Long Term Evolution
LXDE	Lightweight X11 Desktop Environment
M2M	Machine to Machine
MANETs	Mobile Ad hoc NETworks
MIMO	Multiple Input Multiple Output
MITRE	The Mitre Corporation
MMS	Multimedia Messaging Service
MU	Multi-User
NIST	National Institute of Standards and Technology
NOP	No Operation (computer instruction)
NVD	National Vulnerability Database
OS	Operating System
OSI	Open Systems Interconnection
PAN	Personal Area Network

PC	Personal Computer
PDF	Probability Density Function
PHP	PHP Hypertext Preprocessor
QoS	Quality of Service
RFID	Radio Frequency Identification Device
RE	Random Encounter
RV	Random Variant
RV-T	Random Variant – Time
RV-E	Random Variant – Encounter
RWP	Random WayPoint
SANS	SysAdmin, Audit, Network and Security (Escal Institute of Advanced Technology)
SEIRD	Susceptible Exposed Infected Recovered Dormancy
SEIRS	Susceptible Exposed Infected Recovered Susceptible
SEIS	Susceptible Exposed Infected Susceptible
SEPTICOX	Susceptible Exposed Prevented Treated Infected Contained Offline eXposed-offline
SI	Susceptible Infected (epidemic model)
SIR	Susceptible Infected Recovered (epidemic model)
SMTP	Simple Mail Transfer Protocol (email)
SQL	Structured Query Language
SSH	Secure Socket Shell (secure remote login)
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
VM	Virtual Machine
WiFi	Wireless Fidelity
X11	X window system (protocol version 11)
XFCE	XForms Common Environment
XOR	eXclusive-OR

Trademarks

The following are presented in the text and have, or assert to have, status as (registered) trademarks:

Apache Software Foundation

Apache®

Apple Inc.

Apple®, Cocoa Touch®, iPhone®, Safari®

BitTorrent, Inc.

BITTORRENT®

Bluetooth SIG, Inc.

Bluetooth®

Canonical Limited

UBUNTU™

Cisco Systems, Inc.

iOS® (licensed to Apple Inc.)

Eclipse Foundation, Inc.

Eclipse®

Google Inc.

Google™, Android™, Gmail™

Intel Corporation

Intel™

Kai Kreuzer (openHAB Foundation e.V)

OpenHAB®

Linux Mark Institute

Linux®

MathWorks, Inc.

MATLAB®, ThingSpeak™

Microsoft Corporation

Internet Explorer™, Microsoft™, Outlook™, Silverlight™, Visual Basic™, Windows™, Visio™

MITRE Corporation

MITRE®

Mozilla Foundation

Mozilla®, Firefox®

MySQL AB

MySQL®

Netflix, Inc.

NETFLIX®

Opera Software AS

Opera®

Oracle Corporation

Oracle®, Java®, JavaScript™

SCALABLE Network Technologies, Inc.

QualNet®

Silicon Graphics, Inc.

OpenGL®

Software in the Public Interest, Inc.

Debian®

Symantec Corporation

Symantec™

Riverbed Technology, Inc.

OPNET®

Velcro Industries B.V.

Velcro®

Wi-Fi Alliance

Wi-Fi®

References

[1] J. Jackson, S. creese, and M. S. Leeson, "Biodiversity: A security approach for ad hoc networks," in IEEE Symposium on Computational Intelligence in Cyber Security, Paris, France, 2011.

[2] J. Jackson, *Multi-scale location analysis of vulnerabilities and their link to disturbances within digital ecosystems*, University of Warwick, 2017.

[3] J. T. Jackson, and S. Creese, "Virus propagation in heterogeneous bluetooth networks with human behaviors," *IEEE Transactions on Dependable and Secure Computing,* vol. 9, no. 6, 2012.

[4] Bank of America Merrill Lynch, *Bofaml's transforming world atlas investment themes illustrated by maps*, 2016.

[5] Symantec, *Internet security threat report*, 2016.

[6] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications & Tutorials,* vol. 16, no. 1, 2014.

[7] McAfee Labs, *2016 threats predictions*, 2016.

[8] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communication Surveys & Tutorials,* vol. 17, no. 4, pp. 2347-2376, 2015.

[9] Dell, "Dell security annual threat report," 2016.

[10] Kaspersky, *Kaspersky security bulletin 2015*, 2015.

[11] F-Secure Labs, *Mobile threat report q1 2014*, 2014.

[12] D. Moore, V. Paxson, S. Savage *et al.*, "Inside the slammer worm," *IEEE Security & Privacy,* vol. July/August, pp. P 33-39, 2003.

[13] C. Shannon, and D. Moore, "The spread of the witty worm," *IEEE Security & Privacy,* vol. July/August 2004, 2004.

[14] Symantec, *Internet security threat report*, vol. 20, 2015.

[15] L. Bilge, and T. Dumitras, "Before we knew it, an emperical study of zero-day attacks in the real world," in Proceedings of the 2012 ACM conference on Computer and Communications Security, 2012, pp. 833-844.

[16] MITRE. "Common vulnerabilities and exposures," MITRE, http://cve.mitre.org, https://www.cvedetails.com.

[17] D. Kushner, "The real story of stuxnet," *IEEE Spectrum,* vol. 50, no. 3, pp. 48-53, 2013.

[18] M. Stamp, "Risks of monoculture," *Communications of the ACM,* vol. 47, no. 3, pp. 120, 2004.

[19] J. E. Just, and M. Cornwell, "Review and analysis of synthetic diversity for breaking monocultures," in WORM, 2004.

[20] P. Larsen, S. Brunthaler, and M. Franz, "Security through diversity: Are we there yet?," *IEEE Security & Privacy,* vol. 12, no. 2, pp. 28-35, 2014.

[21] A. M. Jones, *Environmental biology*: Routledge, 1997.

[22] A. Wilby, and A. Hector, "The role of biodiversity," *Encyclopedia of life sciences*: John Wiley & Sons Ltd, 2008.

[23] B. B. Lin, "Resilience in agriculture through crop diversification: Adaptive management for environmental change," *BioScience,* vol. 61, no. 3, 2011.

[24] R. S. Ostfeld, and F. Keesing, "Biodiversity and disease risk: The case of lyme disease," *Conservation Biology,* vol. 14, no. 3, pp. 722-728, 2000.

[25] D. Mackenzie, "Disease runs riot as species disappear," *New Scientist*, no. 2715, 2009.

[26] D. Tilman, and J. A. Downing, "Biodiversity and stability in grasslands," *Nature,* vol. 367, no. January, pp. 363-365, 1994.

[27] F. Isbell, D. Craven, J. Connolly *et al.*, "Biodiversity increases the resistance of ecosystem productivity to climate extremes," *Nature,* vol. 526, pp. 574-577, 2015.

[28] L. Chen, and A. Avizienis, "N-version programming: A fault-tolerance approach to reliability of software operation," in International Symposium on Fault-Tolerant Computing, 1978, pp. 3-9.

[29] J. R. Crandall, R. Ensafi, and S. Forrest, "The ecology of malware," in NSPW, 2008, pp. 99-106.

[30] R. C. Linger, "Systematic generation of stochastic diversity as an intrusion barrier in survivable systems software," in 32nd Hawaii International Conference on System Sciences, 1999.

[31] M. C. Mont, A. Baldwin, Y. Beres *et al.*, "Reducing risks of widespread faults and attacks for commercial software applications: Towards diversity of software components," in 26th Annual International Computer Software and Applications Conference, 2002.

[32] A. Alarifi, and W. Du, "Diversify sensor nodes to improve resilience against node compromise," in The Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks, 2006.

[33] T. Jackson, A. Homescu, S. Crane *et al.*, "Diversifying the software stack using randomized nop insertion," *Moving target defense ii*, pp. 151-173: Springer, 2013.

[34] A. Homescu, S. Neisius, P. Larsen, S. Brunthaler, and M. Franz, "Profile-guided automated software diversity," in International Symposium on Code Generation and Optimization, 2013.

[35] A. J. O'Donnell, and H. Sethu, "On achieving software diversity for improved network security using distributed coloring algorithms," in CCS, 2004.

[36] A. J. O'Donnell, and H. Sethu, "Software diversity as a defense against viral propagation: Models and simulations," in Principles of Advanced and Distributed Simulation, 2005.

[37] Y. Yang, S. Zhu, and G. Cao, "Improving sensor network immunity under worm attacks: A software diversity approach," in MobiHoc, 2008.

[38] Y. Zhang, H. Vin, L. Alvisi, W. Lee, and S. K. Dao, "Heterogeneous networking: A new survivability paradigm," in Proceedings of the 2001 ACM workshop on New Security Paradigms, 2001.

[39] M. G. Bailey, "Malware resistant networking using system diversity," in SIGITE, 2005.

[40] S. Neti, A. Somayaji, and M. Locasto, "Software diversity: Security, entropy and game theory," in HotSec'12, 2012.

[41] B. Bangerter, S. Talwar, R. Arefi, and K. Stewart, “Networks and devices for the 5g era,” *IEEE Communications Magazine*, no. 2, 2014.

[42] M. Nei, “Analysis of gene diversity in subdivided populations,” *Proceedings of the National Academy of Sciences,* vol. 70, no. 12, pp. 3321-3323, 1973.

[43] C. M. Lively, “The effect of host genetic diversity on disease spread ” *The American Naturalist,* vol. 175, no. 6, pp. E149-E152, 2010.

[44] A. R. Hughes, B. D. Inouye, M. T. J. Johnson, N. Underwood, and M. Vellend, “Ecological consequences of genetic diversity,” *Ecology Letters,* vol. 11, pp. 609-623, 2008.

[45] R. J. Whittaker, “Evolution and measurement of species diversity,” *Taxon,* vol. 21, 1972.

[46] Y. Song, P. Wang, G. Li, and D. Zhou, “Relationships between functional diversity and ecosystem functioning: A review,” *Acta Ecological Sinica,* vol. 34, pp. 85-91, 2014.

[47] M. Petter, S. Mooney, S. M. Maynard *et al.*, “A methodology to map ecosystem functions to support ecosystem services assessments,” *Ecology and Society,* vol. 18, no. 1, 2013.

[48] D. U. Hooper, F. S. Chapin III, J. J. Ewel *et al.*, “Esa report: Effects of biodiversity on ecosystem functioning: A consensus of current knowledge,” *Ecological Monographs,* vol. 75, no. 1, pp. 3-35, 2005.

[49] S. Diaz, J. Fargione, F. S. Chapin III, and D. Tilman, “Biodiversity loss threatens human well-being,” *PLoS Biology,* vol. 4, no. 8, 2006.

[50] B. J. Cardinale, J. E. Duffy, A. Gonzalez *et al.*, “Biodiversity loss and its impact on humanity,” *Nature,* vol. 486, pp. 59-67, 2012.

[51] P. A. Harrison, P. M. Berry, G. Simpson *et al.*, “Linkages between biodiversity attributes and ecosystem services: A systematic review,” *Ecosystem Services,* vol. 9, 2014.

[52] S. T. A. Pickett, and P. S. White, *The ecology of natural disturbance and patch dynamics*: Academic Press, 1985.

[53] W. P. Sousa, “The role of disturbance in natural communities,” *Annual Review of Ecology and Systematics,* vol. 15, pp. 353-391, 1984.

[54] P. S. White, J. Harrod, W. H. Romme, and J. Betancourt, "Disturbance and temporal dynamics," *Ecological stewardship*, 1999.

[55] R. Keane, "Disturbance regimes and the historical range of variation in terrestrial ecosystems," *The Encyclopedia of Biodiversity*, Elselvier, 2013.

[56] F. Keesing, L. K. Belden, P. Daszak *et al.*, “Impacts of biodiversity on the emergence and transmission of infectious diseases,” *Nature,* vol. 468, pp. 647-652, 2010.

[57] S. A. Levin, and R. T. Paine, “Disturbance, patch formation, and community structure,” *PNAS,* vol. 71, no. 7, pp. 2744-2747, 1974.

[58] R. L. Mackey, and D. J. Currie, “The diversity-disturbance relationship: Is it generally strong and peaked?,” *Ecology,* vol. 82, no. 12, pp. 3479-3492, 2001.

[59] A. D. Miller, S. H. Roxburgh, and K. Shea, “How frequency and intensity shape diversity-disturbance relationships,” *PNAS,* vol. 108, no. 14, 2011.

[60] K. Bohn, R. Pavlick, B. Reu, and A. Kleidon, “The strengths of r- and k-selection shape diversity-disturbance relationships,” *PLOS ONE*, 2014.

[61] W. R., W. S. Ho, K. S. Lee, and L. C. T., "Impact of disturbance on population and genetic structure of tropical forest trees," *Forest Genetics,* vol. 11, pp. 193-201, 2004.

[62] W. Silke, W. H. H., R. Holderegger, and J. M. Kalwij, "Effect of disturbances on the genetic diversity of an old-forest associated lichen," *Molecular Ecology,* vol. 15, pp. 911-921, 2006.

[63] N. Farwig, C. Braun, and K. Bohning-Gaese, "Human disturbance reduces genetic diversity of an endangered tropical tree, prunus africana (rosaceae)," *Conservation genetics,* vol. 9, no. 2, pp. 317-326, 2008.

[64] A. J. Symstad, F. S. Chapin III, D. H. Wall *et al.*, "Long-term and large-scale perspectives on the relationship between biodiversity and ecosystem functioning," *BioScience,* vol. 53, no. 1, 2003.

[65] N. Bluthgen, and A.-M. Klein, "Functional complementarity and specialisation: The role of biodiversity in plant-pollinator interactions," *Basic and Applied Ecology,* vol. 12, no. 4, 2011.

[66] D. Tilman, "Biodiversity: Population versus ecosystem stability," *Ecology Letters,* vol. 77, pp. 350-363, 1996.

[67] G. Allison, "The influence of species diversity and stress intensity on community resistance and resilience," *Ecological Monographs,* vol. 74, no. 1, pp. 117-134, 2004.

[68] A. R. Hughes, and J. J. Stachowicz, "Genetic diversity enhances the resistance of a seagrass ecosystem to disturbance," *Proceedings of the National Academy of Sciences of the United States of America,* vol. 102, pp. 8998-9002, 2004.

[69] A. R. Hughes, and J. J. Stachowicz, "Seagrass genotypic diversity increases disturbance response via complementarity and dominance," *Journal of Ecology,* vol. 99, pp. 445-453, 2011.

[70] G. E. Hutchinson, "Concluding remarks," *Cold Spring Harbor Symposia on Quantitative Biology,* vol. 22, pp. 415-427, 1957.

[71] G. F. Gause, *The struggle for existence*, Baltimore: The Williams & Wilkins Company, 1934.

[72] P. A. Abrams, C. M. Tucker, and B. Gilbert, "Evolution of the storage effect," *Evolution,* vol. 67, no. 2, pp. 315-327, 2012.

[73] T. B. H. Reusch, A. Ehlers, A. Haemmerli, and B. Worm, "Ecosystem recovery after climatic extremes enhanced by genotypic diversity.," *Proceedings of the National Academy of Sciences of the United States of America,* vol. 102, pp. 2826-2831, 2005.

[74] S. C. Cook-Patton, S. H. McArt, A. L. Parachnowitsch, J. S. Thaler, and A. A. Agrawal, "A direct comparison of the consequences of plant genotypic and species diversity on communities and ecosystem function," *Ecology,* vol. 92, no. 4, pp. 915-923, 2011.

[75] J. Frund, C. F. Dormann, A. Holzschuh, and T. Tscharntke, "Bee diversity effects on pollination depend on functional complementarity and niche shifts," *Ecology,* vol. 94, no. 9, 2013.

[76] R. Kafri, M. Springer, and Y. Pilpel, "Genetic redundancy: New tricks for old genes," *Cell,* vol. 136, no. 3, pp. 389-392, 2009.

[77] A. Wagner, "Redundant gene functions and natural selection," *Journal of Evolutionary Biology,* vol. 12, pp. 1-16, 1999.

[78] J. Ellers, S. Rog, B. C, and B. M. P., “Genotypic richness and phenotypic dissimilarity enhance population performance,” *Ecology,* vol. 92, no. 8, pp. 1605-1615, 2011.

[79] C. E. Proffitt, R. L. Chiasson, A. B. Owens, and S. E. Travis, “Spartina alterniflora genotype influences facilitation and suppression of high marsh species colonizing an early successional salt marsh,” *Journal of Ecology,* vol. 93, no. 2, pp. 404-416, 2005.

[80] J. Vandermeer, *The ecology of intercropping*: Cambridge University Press, Cambridge, UK, 1989.

[81] A. Kahmen, and N. Buchmann, "Chapter 22: Addressing the functional value of biodiversity for ecosystem functioning using stable isotopes," *Stable isotopes as indicators of ecological change*: Elsevier, 2007.

[82] P. Balvanera, A. B. Pfisterer, N. Buchmann *et al.*, “Quantifying the evidence for biodiversity effects on ecosystem functioning and services,” *Ecology Letters,* vol. 9, pp. 1146-1156, 2006.

[83] B. K. Hall, *Evolution: Principles and processes*: Jones and Bartlett Publishers, 2011.

[84] J. Adds, E. Larkcom, and R. Miller, *Genetics, evolution and biodiversity*: Nelson Thornes Ltd, 2004.

[85] A. F. Wright, "Genetic variation: Polymorphisms and mutations," *Encyclopedia of life sciences*: John Wiley & Sons, 2005.

[86] A. R. Hughes, “Disturbance and diversity: An ecological chicken and egg problem,” *Nature Education Knowledge,* vol. 3, no. 10, 2010.

[87] P. J. Morin, *Community ecology (second edition)*: Wiley-Blackwell, 2011.

[88] C. L. Lehman, and D. Tilman, “Biodiversity, stability, and productivity in competitive communities,” *The American Naturalist,* vol. 156, no. 5, 2000.

[89] K. S. McCann, “The diversity-stability debate,” *Nature,* vol. 405, 2000.

[90] D. Tilman, P. B. Reich, and J. M. H. Knops, “Biodiversity and ecosystem stability in a decade-long grassland experiment,” *Nature,* vol. 441, pp. 629-632, 2006.

[91] G. M. Crutsinger, L. Souza, and N. J. Sanders, “Intraspecific diversity and dominant genotypes resist plant invasions,” *Ecology Letters,* vol. 10, 2007.

[92] T. A. Kennedy, S. Naeem, K. M. Howe *et al.*, “Biodiversity as a barrier to ecological invasion,” *Nature - Letter,* vol. 417, pp. 636-638, 2002.

[93] L. Gunderson, C. S. Holling, L. Pritchard, and G. D. Peterson, "Resilience," *Encyclopedia of global environmental change*, 2002.

[94] H. E. Creissen, T. H. Jorgensen, and J. K. Brown, “Impact of disease on diversity and productivity of plant populations,” *Functional Ecology,* vol. 30, no. 4, pp. 649-657, 2016.

[95] D. Spielman, B. W. Brook, D. A. Briscoe, and R. Frankham, “Does inbreeding and loss of genetic diversity increase disease resistance?,” *Conservation Genetics,* vol. 5, pp. 439-448, 2004.

[96] K. A. Schmidt, and R. S. Ostfeld, “Biodiversity and the dilution effect in disease ecology,” *Ecology,* vol. 82, no. 3, 2001.

[97] I. Pagan, P. Gonzalez-Jara, A. Moreno-Letelier *et al.*, “Effect of biodiversity changes in disease risk: Exploring disease emergence in a plant-virus system,” *PLoS Pathogens*, 2012.

[98] U. R. Zargar, M. Z. Chrishti, F. Ahmad, and M. I. Rather, "Does alteration in biodiversity really affect disease outcome? - a debate is brewing," *Saudi Journal of Biological Sciences,* vol. 22, no. 1, pp. 14-18, 2015.

[99] K. C. King, and C. M. Lively, "Does genetic diversity limit disease spread in natural host populations?," *Heredity,* vol. 109, pp. 199-203, 2012.

[100] J. F. Tooker, and S. D. Frank, "Genotypically diverse cultivar mixtures for insect pest management and increased crop yields," *Journal of Applied Ecology,* vol. 49, no. 5, pp. 974-985, 2012.

[101] Y. Zhu, H. Chen, F. Jinghua *et al.*, "Genetic diversity and disease control in rice," *Nature,* vol. 406, no. 718-722, 2000.

[102] F. Altermatt, and D. Ebert, "Genetic diversity of daphnia magna populations enhances resistance to parasites," *Ecology Letters,* vol. 11, pp. 918-928, 2008.

[103] H. H. Ganz, and D. Ebert, "Benefits of host genetic diversity for resistance to infection depend on parasite diversity," *Ecology,* vol. 91, pp. 1263-1268, 2010.

[104] A. E. Magurran, *Measuring biological diversity*: Blackwell, 2003.

[105] K. Van Doninck, I. Schon, K. Martens, and T. Backeljau, "Clonal diversity in the ancient asexual ostracod darwinula stevensoni assessed by rapd-pcr," *Heredity,* vol. 93, pp. 154-160, 2004.

[106] J. A. Stoddart, "A genotypic diversity measure," *Journal of Heredity,* vol. 74, pp. 489-490, 1983.

[107] D. L. Hawksworth, *Biodiversity measurement and estimation*: Springer, 2009.

[108] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal,* vol. 27, no. 3, pp. 379-423, 1948.

[109] A. Lowe, S. Harris, and P. Ashton, *Ecological genetics, design analysis and apllication*: Blackwell Publishing, 2004.

[110] M. Franz, "E unibus pluram: Massive-scale software diversity as a defense mechanism," in Proceedings of the 2010 workshop on New security paradigms, 2010.

[111] K. Makris, and R. A. Bazzi, "Immediate multi-threaded dynamic software updates using stack reconstruction," in USENIX Annual technical conference, 2009.

[112] G. Chen, H. Jin, D. Zou *et al.*, "A framework for practical dynamic software updating," *IEEE Transactions on Parallel and Distributed Systems,* vol. 27, no. 4, 2016.

[113] Wind River, *White paper: Virtualization and the internet of things*, 2014.

[114] T. Sridhar, "Cloud computing: A primer, part 1: Models and technologies," *The Internet Protocol Journal,* vol. 12, no. 3, 2009.

[115] A. Armando, G. Costa, L. Verderame, and A. Merlo, "Securing the "Bring your own device" Paradigm," *IEEE Computer,* vol. 47, no. 6, 2014.

[116] "Horizon mobile secure workplace." www.vmware.com.

[117] "The xen project." http://www.xenproject.org/.

[118] G. Merlino, D. Bruneo, S. Distefano, F. Longo, and A. Puliafito, "Enabling mechanisms for cloud-based network virtualization in iot," in IEEE 2nd World Forum on Internet of Things, 2015, pp. 268-273.

[119] F. Ramalho, and A. Neto, "Virtualization at the network edge: A performance comparison," in IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks, 2016, pp. 1-6.

[120] K. S. Dar, A. Taherkordi, and F. Eliassen, “Enhancing dependability of cloud-based iot services through virtualization,” in IEEE First International Conference on Internet-of-things Design and Implementation, 2016, pp. 106-116.
[121] H. Ko, J. Jin, and S. L. Keoh, “Secure service virtualization in iot by dynamic service dependency verification,” *IEEE Internet of Things Journal*, no. 99, 2016.
[122] D. Kelaidonis, A. Somov, V. Foteinos *et al.*, “Virtualization and cognitive management of real world objects in the internet of things,” in IEEE International Conference on Green Computing and Communications, 2012, pp. 187-194.
[123] N. L. S. da Fonseca, and R. Boutaba, "Virtualization in the cloud," *Cloud services, networking, and management*: Wiley-IEEE Press eBook Chapters, 2015.
[124] *Internet of things - converging technologies for smart environments and integrated ecosystems*: River Publishers, 2013.
[125] C. Greamo, and A. Ghosh, “Sandboxing and virtualization, modern tools for combating malware,” *IEEE Security & Privacy,* vol. March/April, pp. 79-82, 2011.
[126] A. J. Younge, R. Henschel, J. T. Brown *et al.*, “Analysis of virtualization technologies for high performance computing environments,” in IEEE International Conference on Cloud Computing, 2011, pp. 9-16.
[127] E. Ipek, M. Kirman, N. Kirman, and J. F. Martinez, “Core fusion: Accommodating software diversity in chip multiprocessors,” in ISCA, 2007.
[128] E. Ipek, M. Kirman, N. Kirman, and J. F. Martinez, “A reconfigurable chip multiprocessor architecture to accommodate software diversity,” in Parallel and Distributed Processing, 2007.
[129] A. S. Tanenbaum, *Computer networks - fourth edition*: Pearson Education International, 2003.
[130] Google. "Android," https://www.android.com/.
[131] S. Khan, M. Nauman, A. T. Othman, and S. Musa, “How secure is your smartphone: An analysis of smartphone security mechanisms,” in IEEE International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012.
[132] Apple. "Ios," http://www.apple.com/uk/.
[133] N. Smyth, *Ios 7 app development essentials: Developing ios 7 iphone and ipad apps with xcode 5*: eBookFrenzy, 2013.
[134] Microsoft. "Windows 8," https://www.microsoft.com/.
[135] I. Novaik, Z. Arvai, G. Balaissy, and D. Fulop, *Beginning windows 8 application development*: John Wiley & Sons, 2012.
[136] Linux. "Linux," https://www.linux.com/.
[137] M. Helmke, *Ubuntu unleashed 2016 edition: Covering 15.10 and 16.04*: Sams Publishing, 2015.
[138] D. P. Bovet, and M. Cesati, *Understanding the linux kernel*: O'reilly, 2006.
[139] North Bridge and Black Duck, *2015 future of open source survey*, www.slideshare.net/North_Bridge/2015-future-of-open-source-study, 2015.
[140] "Physical web." https://google.github.io/physical-web/.
[141] "Lelylan." http://www.lelylan.com/.
[142] "Thing speak." https://thingspeak.com/.

[143] "Bug labs." http://buglabs.net/.

[144] "The thing system." http://thethingsystem.com/things/index.html.

[145] "Open remote." http://www.openremote.org/display/HOME/OpenRemote.

[146] A. Gupta, and R. Kumar JHA, "A survey of 5g network: Architecture and emerging technologies," *IEEE Access: Special Section on Recent Advances in Software Defined Networking for 5G Networks,* vol. 3, pp. 1206-1232, 2015.

[147] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: The internet of things architecture, possible applications and key challenges," in 10th IEEE International Conference on Frontiers of Information Technology, 2012.

[148] S. K. Sarkar, T. G. Basavaraju, and C. Puttamadappa, *Ad hoc mobile wireless networks: Principles, protocols, and applications, second edition*: CRC Press, 2013.

[149] L. Yan, "Can p2p benefit from manet? Performance evaluation from users' perspective," *Mobile ad-hoc and sensor networks: First international conference, msn 2005*, Lecture notes in computer science 3794, pp. 1026-1035: Springer Berlin Heidelberg, 2005.

[150] J. Chandra, S. Delitzscher, N. Ganguly, and A. Jhunjhunwala, "Optimizing topology in bit torrent based networks," in IEEE Computer Communications Workshops, 2011.

[151] R. Ramanathan, and J. Redi, "A brief overview of ad hoc networks: Challenges and directions," *IEEE Communications Magazine,* vol. 40, no. 5, pp. 20-22, 2002.

[152] R. Schollmeier, I. Gruber, and M. Finkenzeller, "Routing in mobile ad hoc and peer-to-peer networks. A comparison," in International workshop on Peer-to-peer Computing. In networking, 2002.

[153] R. Bruno, M. Conti, and E. Gregori, "Mesh networks: Commodity multihop," *IEEE Communications Magazine,* vol. 43, no. 3, 2005.

[154] D. G. Reina, S. L. Toral, F. Barrero, N. Bessis, and E. Asimakopoulou, "The role of ad hoc networks in the internet of things: A case scenario for smart environments," *Internet of things and inter-cooperative computational technologies for collective intelligence*, Studies in Computational Intelligence, pp. 89-113: Springer-Verlag Berlin Heidelberg, 2013.

[155] J. Loo, J. L. Mauri, and J. H. Ortiz, *Mobile ad hoc networks: Current status and future trends*: CRC Press, 2016.

[156] "Mobile malware evolution: An overview, part 1." Kaspersky, http://www.viruslist.com/en/analysis?pubid=200119916#fams.

[157] Securelist. "Mobile malware evolution 2015," 9th September 2016, https://securelist.com/analysis/kaspersky-security-bulletin/73839/mobile-malware-evolution-2015/.

[158] V. Zhang. "'godless' mobile malware uses multiple exploits to root devices," 1st September 2016, 2016, TrendLabs Security Intelligence Blog, Trend Micro, http://blog.trendmicro.com/trendlabs-security-intelligence/godless-mobile-malware-uses-multiple-exploits-root-devices/.

[159] Lookout and Citizen Lab. "Sophisticated, persistent mobile attack against high-value targets on ios," 1st September 2016, https://blog.lookout.com/blog/2016/08/25/trident-pegasus/.

[160] L. H. Newman. "A hacking group is selling iphone spyware to governments," 2nd September 2016, Wired, https://www.wired.com/2016/08/hacking-group-selling-ios-vulnerabilities-state-actors/.

[161] Kaspersky, *Kaspersky security bulletin 2014*, 2014.

[162] S. Christey, "2010 cwe/sans top 25 most dangerous software errors," MITRE & SANS, 2010.

[163] S. Jain, and S. R. Das, "Exploiting path diversity in the link layer in wireless ad hoc networks," *Ad Hoc Networks,* vol. 6, no. 5, pp. 805-825, 2008.

[164] T. Issariyakul, and V. Krishnamurthy, "Amplify-and-forward cooperative diversity wireless networks: Model, analysis, and monotonicity properties," *IEEE/ACM Transactions on Networking,* vol. 17, no. 1, pp. 225-238, 2009.

[165] H. Lim, C. Lim, and J. C. Hou, "A coordinate-based approach for exploiting temporal-spatial diversity in wireless mesh networks," in MobiCom, 2006.

[166] Y. Zhou, Z.-F. Wu, H. Wang *et al.*, "Breaking monocultures in p2p networks for worm prevention," in Machine Learning and Cybernetics, 2006.

[167] B. Anckaert, B. De Sutter, and K. De Bosschere, "Software piracy prevention through diversity," in DRM, 2004.

[168] H. Shacham, M. Page, B. Pfaff *et al.*, "On the effectiveness of address-space randomization," in 11th ACM Conference on Computing Communication Security, 2004, pp. 298-307.

[169] N. L. Hung, A. R. Jacob, and S. E. Makris, "Alternatives to achieve software diversity in common channel signaling networks," *IEEE Journal on Selected Areas in Communications,* vol. 12, no. 3, pp. 533-538, 1994.

[170] B. Littlewood, P. Popov, and L. Strigini, "Modeling software design diversity - a review," *ACM Computing Surveys,* vol. 33, no. 2, pp. 177-208, 2001.

[171] G. Gaiswinkler, and A. Gerstinger, "Automated software diversity for hardware fault detection," in Emerging Technologies & Factory Automation, 2009.

[172] M. R. Lyu, J.-H. Chen, and A. Avizienis, "Software diversity metrics and measurements," in Computer Software and Applications Conference, 1992.

[173] B. Nikolik, "Test diversity," *Information and Software Technology,* vol. 48, no. 11, pp. 1083-1094, 2006.

[174] S. Forrest, A. Somayaji, and D. H. Ackley, "Building diverse computer systems," in Workshop on Hot Topics in Operating Systems, 1997.

[175] B. Baudry, M. Monperrus, C. Mony, F. Chauvel, and S. Clarke, "Diversify - ecology-inspired software evolution for diversity emergence," in International Conference on Software Maintenance and Reengineering (CSMR), 2014.

[176] S. Shetty, X. Yuchi, and M. Song, "Scalable network diversity modeling for assessing threats in cloud networks," *Moving target defense for distributed systems. Wireless networks*: Springer Verlag, 2016.

[177] C. L. Smith, "Understanding concepts in the defence in depth strategy," in IEEE 37th Annual International Carnahan Conference on Security Technology, 2003.

[178] P. Larsen, A. Homescu, S. Brunthaler, and M. Franz, "Sok: Automated software diversity," in IEEE Symposium on Security and Privacy, 2014, pp. 276-291.

[179] S. Allier, O. Barais, and e. al, "Multi-tier diversification in web-based software applications," *IEEE Software,* vol. 32, no. 1, pp. 83-90, 2015.

[180] B. Baudry, S. Allier, and M. Monperrus, "Tailored source code transformations to synthesize computationally diverse program variants," in International Symposium on Software Testing and Analysis, 2014.

[181] A. Gupta, J. Habibi, M. S. Kirkpatrick, and E. Bertino, "Marlin: Mitigating code reuse attacks using code randomization," *IEEE Transactions on Dependable and Secure Computing,* vol. 12, no. 3, 2015.

[182] M. Murphy, P. Larsen, S. Brunthaler, and M. Franz, "Software profiling options and their effects on security based diversification " in First ACM workshop on Moving Target Defense, 2014.

[183] T. Jackson, B. Salamat, A. Homescu *et al.*, "Compiler-generated software diversity," *Moving target defense*, Advances in information security, pp. 77-98: Springer New York, 2011.

[184] S. Bhatkar, D. C. DuVarney, and R. Sekar, "Address obfuscation: An efficient approach to combat a broad range of memory error exploits," in 12th Conference on USENIX Security Symposium, 2003.

[185] J. C. Knight, "Diversity," *Dependable and historic computing*, Lecture notes in computer science, pp. 298-312: Springer Berlin Heidelberg, 2011.

[186] D. H. Aristizabal, D. M. Rodriguez, and R. Y. Guevara, "Measuring aslr implementations on modern operating systems," in IEEE International Carnahan Conference on Security Technology, 2013, pp. 1-6.

[187] S. Bhatkar, and R. Sekar, "Data space randomization," *Detection of intrusions and malware, and vulnerability assessment*, Lecture notes in computer science, pp. 1-22: Springer Berlin Heidelberg, 2008.

[188] S. W. Boyd, G. S. Kc, M. E. Locasto, A. Keromytis, and V. Prevelakis, "On the general applicability of instruction-set randomization," *IEEE Transactions on Dependable and Secure Computing,* vol. 7, no. 3, pp. 255-270, 2010.

[189] Z. Liang, B. Liang, L. Li *et al.*, "Against code injection with system call randomization," in IEEE International Conference on Networks Security, Wireless Communications and Trusted Computing, 2009.

[190] E. G. Barrantes, D. H. Ackley, S. Forrest, D. Stefanovic, and D. D. Zovi, "Randomized instruction set emulation to disrupt binary code injection attacks," in ACM Transactions on Information and System Security, 2005.

[191] G. S. Kc, A. D. Keromytis, and V. Prevelakis, "Countering code-injection attacks with instruction-set randomization," in 10th ACM Conference on Computer and Communications Security, 2003, pp. 272-280.

[192] A. N. Sovarel, D. Evans, and N. Paul, "Where's the feeb? The effectiveness of instruction set randomization," in 14th Conference on USENIX Security Symposium, 2005, pp. 10.

[193] V. Pappas, M. Polychronakis, and A. D. Keromytis, "Practical software diversification using in-place code randomization," *Moving target defense ii*, pp. 175-202: Springer New York, 2013.

[194] T. R. Jenson, and B. Toft, *Graph coloring problems*: John Wiley & Sons, 1995.

[195] S. Hosseini, M. A. Azgomi, and A. T. Rahmani, "A malware propagation model considering software diversity," in 11th International ISC Conference on Information Security and Cryptology, 2014.

[196] K. Hole, “Diversity reduces the impact of malware,” *IEEE Security & Privacy,* vol. 99, 2013.

[197] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic programming - an introduction*: Morgan Kaufmann Publishers, Inc, 1998.

[198] R. Feldt, “Generating diverse software versions with genetic programming,” in IEEE proceedings-Software, 1998.

[199] N. Holtschulte, and M. Moses, “Diversity and resistance in a model network with adaptive software,” *Security Informatics - a SpringerOpen Journal,* vol. 1, no. 19, Nov 2012.

[200] A. Newell, D. Obenshain, T. Tantillo, Nita-Rotaru, and A. Yair, “Increasing network resiliency by optimally assigning diverse variants to routing nodes,” in IEEE/IFIP International Conference on Dependable Systems and Networks, 2013.

[201] E. Totel, F. Majorczyk, and L. Me, "Cots diversity based intrusion detection and application to web servers," *Recent advances in intrusion detection*, Lecture notes in computer science: Springer Berlin Heidelberg, 2006.

[202] B. Cox, D. Evans, A. Filipi *et al.*, “N-variant systems a secretless framework for security through diversity,” in 15th USENIX security Symposium, 2006.

[203] M. Azab, “Multidimensional diversity employment for software behaviour encryption,” in 6th International Conference on New Technologies, Mobility and Security, 2014.

[204] H. Wang, D. Fang, G. Li *et al.*, “Tdvmp: Improved virtual machine-based software protection with time diversity,” in ACM SIGPLAN on Program Protection and Reverse Engineering, 2014.

[205] K. Kravvaritis, D. Mitropoulos, and D. Spinellis, “Cyberdiversity: Measures and initial results,” in 14th Panhellenic Conference on Informatics, 2010.

[206] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, “Os diversity for intrusion tolerance: Myth or reality?,” in IEEE/IFIP International conference on Dependable Systems and Networks, 2011.

[207] J. Han, D. Gao, and R. H. Deng, "On the effectiveness of software diversity: A systematic study on real-world vulnerabilities," *Detection of intrusions and malware, and vulnerability assessment*, Lecture notes in computer science 5587, pp. 127-146: Springer Berlin Heidelberg, 2009.

[208] R. Khoury, Hamou-Lhadj, and M. Couture, “Towards a formal framework for evaluating the effectiveness of system diversity when applied to security,” in IEEE Symposium on Computational Intelligence for Security and Defence Applications, 2012.

[209] C. Taylor, and J. Alves-Foss, “Diversity as a computer defense mechanism a panel,” in NSPW, 2006.

[210] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, “Understanding individual human mobility patterns,” *Nature,* vol. 453, pp. 779-782, 2008.

[211] T. Camp, J. Boleng, and V. Davies, “A survey of mobility models for ad hoc network research,” *Wireless Communications & Mobile Computing: Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2002.

[212] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, “A performance comparison of multi-hop wireless ad hoc network routing protocols,” in

ACM/IEEE International Conference on Mobile Computing and Networking, 1998, pp. 85-97.

[213] G. Yan, L. Cuellar, S. Eidenbenz *et al.*, “Bluetooth worm propagation: Mobility pattern matters!,” in ACM symposium on Information, computer and communications security, 2007, pp. 32-44.

[214] H. Xiang, J. Liu, and J. Kuang, “Minimum node degree and connectivity of two-dimensional manets under random waypoint mobility model,” in IEEE 10th International Conference on Computer and Information Technology, 2010.

[215] T. Anouari, and A. Haqiq, “Performance analysis of routing protocols in wimax using random waypoint model,” in IEEE 20th International Conference on Telecommunications, 2013, pp. 1-5.

[216] A. Roy, P. Bijan, and S. K. Paul, “Vanet topology based routing protocols & performance of aodv, dsr routing protocols in random waypoint scenarios,” in IEEE 1st International Conference on Computer & Information Engineering, 2015.

[217] L. Irio, and R. Oliveira, “Interference estimation in wireless mobile random waypoint networks,” in IEEE 23rd Telecommunications forum, 2015.

[218] D. B. Johnson, and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile computing*, Tomasz Imielinski and Hank Korth, ed., pp. 153-181: Kluwer Academic Publishers, 1996.

[219] M. Niazi, and A. Hussain, “Agent-based tools for modeling and simulation of self-organization in peer-to-peer, ad hoc, and other complex networks,” *IEEE Communications Magazine,* vol. 47, no. 3, pp. 166-173, 2009.

[220] H. Alharbi, and A. Hussain, “An agent-based approach for modelling peer to peer networks,” in 17th UKSIM-AMSS International Conference on Modelling and Simulation, 2015.

[221] S. Johnson, *Emergence*: Penguin Books, 2002.

[222] A. Pramanik, B. Choudhury, T. S. Choudhury, and W. A. Mehedi, J., “Simulative study of random waypoint mobility model for mobile ad hoc networks,” in IEEE Global Conference on Communication Technologies, 2015, pp. 112-116.

[223] M. Babis, and P. Magula, “Netlogo - an alternative way of simulating mobile ad hoc networks,” in 5th Joint IFIP Wireless and Mobile Networking Conference, 2012, pp. 122-125.

[224] M. E. J. Newman, “The structure and function of complex networks,” *SIAM Review,* vol. 45, no. 2, pp. 167-256, 2003.

[225] T. Tyrakowski, and Z. Palka, “A random graph model of mobile wireless networks,” *Electronic Notes in Discrete Mathematics,* vol. 22, pp. 311-314, 2005.

[226] H. Kawahigashi, Y. Terashima, N. Miyauchi, and T. Nakakawaji, “Modeling ad hoc sensor networks using random graph theory,” in 2nd IEEE Consumer Communications and Networking Conference, 2005.

[227] M. J. Keeling, and P. Rohani, *Modeling infectious diseases in humans and animals*: Princeton University Press, 2008.

[228] D. J. Daley, and J. Gani, *Epidemic modelling: An introduction*, 1999.

[229] H. Zheng, D. Li, and Z. Gao, “An epidemic model of mobile phone virus,” in Symposium on Pervasive Computing and Applications, 2006, pp. 1-5.

[230] R. W. Thommes, and M. J. Coates, “Modeling virus propagation in peer-to-peer networks,” in Conference on Information, Communications and Signal Processing, 2005.

[231] W. Xia, Z. Li, Z. Chen, and Z. Yuan, “Dynamic epidemic model of smart phone virus propagated through bluetooth and mms,” in IET Conference on Wireless, Mobile and Sensor Networks, 2007, pp. 948-953.

[232] W. O. Kermack, and A. G. McKendrick, “A contribution to the mathematical theory of epidemics,” *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character,* vol. 115, no. 772, pp. 700-721, 1927.

[233] J. Jones, Holland, *Notes on r0*, Department of Anthropological Sciences, Stanford University, 2007.

[234] T. Britton, “Stochastic epidemic models: A survey,” *Mathematical Biosciences,* vol. 225, no. 1, pp. 24-35, 2010.

[235] L. J. Allen, “An introduction to stochastic epidemic models,” *Mathematical Epidemiology, Springer Berlin Heidelberg*, pp. 81-130, 2008.

[236] M. Baruch, and P. V. Sasorov, “Wkb theory of epidemic fade-out in stochastic populations,” *Physical Review,* vol. 80, no. 4, 2009.

[237] M. Adeel, and L. N. Tokarchuk, “Analysis of mobile p2p malware detection through cabir & commwarrior families,” in IEEE Third international Conference on Privacy, Security, Risk and Trust, and IEEE Third international Conference on Social Computing, 2011, pp. 1335-1343.

[238] H. H. Flor, “The complementary genetic systems in flax and flax rust,” *Advances in Genetics,* vol. 8, pp. 29-54, 1956.

[239] S. P. Otto, and Y. Michalakis, “The evolution of recombination in changing environments,” *Trends in Ecology and Evolution,* vol. 13, pp. 145-151, 1998.

[240] A. Agrawal, and C. M. Lively, “Infection genetics: Gene-for-gene versus matching-alleles models and all points in between,” *Evolutionary Ecology Research,* vol. 4, pp. 79-90, 2002.

[241] R. K. Grosberg, and M. W. Hart, “Mate selection and the evolution of highly polymorphic self/nonself recognition genes,” *Science,* vol. 289, pp. 2111-2114, 2000.

[242] H. b. Christensen, K. M. Hansen, M. Kyng, and K. Manikas, “Analysis and design of software ecosystem architectures - towards the 4s telemedicine ecosystem,” *Information and Software Technology, Special issue on Software Ecosystems,* vol. 56, no. 11, pp. 1476-1492, 2014.

[243] Kaspersky. "Internet security threats," http://www.kaspersky.co.uk/internet-security-center/threats.

[244] M. Hypponen, “Malware goes mobile,” *Scientific American*, November, 2006.

[245] S. Abu-Nimeh, M. Becher, S. Fogie *et al.*, *Mobile malware attacks and defense*: Syngress Publishing, 2009.

[246] M. Yesilyurt, and Y. Yalman, “Security threats on mobile devices and their effects: Estimations for the future,” *International Journal of Security and Its Applications,* vol. 10, no. 2, pp. 13-26, 2016.

[247] T. Bheemarjuna Reddy, I. Karthigeyan, B. S. Manoj, and C. Siva Ram Murthy, “Quality of service provisioning in ad hoc wireless networks: A survey of issues and solutions,” *Ad Hoc Networks,* vol. 4, pp. 83-124, 2006.

[248] M. Asif, S. Khan, R. Ahmad, and M. Sohail, "Quality of service of routing protocols in wireless sensor networks: A review," *IEEE Access,* vol. 5, 2017.

[249] Apple. "Use the app store on your ios devices, apple tv, or computer," https://support.apple.com/en-gb/HT204266.

[250] Google. "Google play - apps," https://play.google.com/store/apps.

[251] *Blueborne technical report*, Armis Labs, https://www.armis.com/blueborne/, 2017.

[252] *Copycat - an in-depth analysis of the copycat android malware campaign*, Check Point Software Technologies Ltd, https://www.checkpoint.com/downloads/resources/copycat-research-report.pdf, 2017.

[253] G. Geoffrey, and S. David, *Probability and random processes, third edition*: Oxford University Press, 2008.

[254] H. W. Hethcote, "The mathematics of infectious diseases," *SIAM Review,* vol. 42, no. 4, pp. 599-653, 2000.

[255] I. H. Spicknall, B. Foxman, C. F. Marrs, and J. N. S. Eisenberg, "A modelling framework for the evolution and spread of antibiotic resistance: Literature review and model categorization," *American Journal of Epidemiology,* vol. 178, no. 4, pp. 508-520, 2013.

[256] J. C. Detilleux, "Effectiveness analysis of resistance and tolerance to infection," *Genetics Selection Evolution,* vol. 43, no. 9, 2011.

[257] J. C. De Roode, C. L. De Castillejo, Fernandez, T. Faits, and S. Alizon, "Virulence evolution in response to anti-infection resistance: Toxic food plants can select for virulent parasites of monarch butterflies," *Journal of Evolutionary Biology,* vol. 24, pp. 712-722, 2011.

[258] P. Holme, "Extinction times of epidemic outbreaks in networks," *PLOS ONE,* vol. 8, no. 12, 2013.

[259] Q. H. Spencer, C. B. Peel, A. L. Swindlehurst, and M. Haardt, "An introduction to the multi-user mimo downlink," *IEEE Communications Magazine,* vol. 42, no. 10, pp. 60-67, 2004.

[260] *Bluetooth core specification v5.0*, Bluetooth SIG, https://www.bluetooth.com, 2016.

[261] Y. Sun, W. Yu, Z. Han, and K. J. Ray Liu, "Trust modeling and evaluation in ad hoc networks," in IEEE Globecom, 2005.

[262] G. Theodorakopoulos, and J. S. Baras, "On trust models and trust evaluation metrics for ad hoc networks," *IEEE Journal on Selected Areas in Communications,* vol. 24, no. 2, 2006.

[263] J. Su, and K. K. W. Chan, "A preliminary investigation of worm infections in a bluetooth environment," in WORM, 2006.

[264] Gartner. "Worldwide smartphone sales to end users by operating system in 1q16," http://www.gartner.com/newsroom/id/3323017.

[265] *Zero-client computing, white paper*, www.digi.com, 2006.

This page intentionally left blank