

**Original citation:**

He, Ligang, Chaudhary, Nadeem and Jarvis, Stephen A.. (2014) Developing security-aware resource management strategies for workflows. *Future Generation Computer Systems*, 38 (9). pp. 61-68.

<http://dx.doi.org/10.1016/j.future.2013.09.030>

**Permanent WRAP url:**

<http://wrap.warwick.ac.uk/65088>

**Copyright and reuse:**

The Warwick Research Archive Portal (WRAP) makes this work of researchers of the University of Warwick available open access under the following conditions. Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.


**Publisher statement:**

© 2014 Elsevier, Licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International <http://creativecommons.org/licenses/by-nc-nd/4.0/>

**A note on versions:**

The version presented here may differ from the published version or, version of record, if you wish to cite this item you are advised to consult the publisher's version. Please see the 'permanent WRAP url' above for details on accessing the published version and note that access may require a subscription.

For more information, please contact the WRAP Team at: [publications@warwick.ac.uk](mailto:publications@warwick.ac.uk)

warwick**publications**wrap  
  
highlight your research

<http://wrap.warwick.ac.uk/>

# Developing Security-aware Resource Management Strategies for Workflows

Ligang He

Department of Computer Science  
University of Warwick  
Coventry, United Kingdom, CV4 7AL  
Email: liganghe@dcs.warwick.ac.uk

Nadeem Chaudhary

Department of Computer Science  
University of Warwick  
Coventry, United Kingdom, CV4 7AL  
Email: nadeem@dcs.warwick.ac.uk

Stephen A. Jarvis

Department of Computer Science  
University of Warwick  
Coventry, United Kingdom, CV4 7AL  
Email: saj@dcs.warwick.ac.uk

**Abstract**—This paper investigates the resource allocation problem for a type of workflows in pervasive computing. These workflows are abstracted from the enterprise-level applications in the business or commerce area. The activities in these workflows require not only computing resources, but also human resources. Human involvement introduces additional security concerns. When we plan/allocate resource capacities, we often assume that when a task is allocated to a resource, the resource will accept the task and start the execution once the processor becomes available. However, the security policies impose further constraints on task executions, and therefore may affect both application- and system-oriented performance. Authorization is an important aspect in security. This paper investigates the issue of allocating resources for running workflows under the role-based authorization control, which is one of the most popular authorization mechanisms. By taking into account the authorization constraints, the resource allocation strategies are developed in this paper for both human resources and computing resources. In the allocation strategy for human resources, the optimization equation is constructed subject to the constraint of the budget available to hire human resources. Then the optimization equation is solved to obtain the number of human resources allocated to each authorization role. The allocation strategy for computing resources calculates not only the number of computing resources, but also the proportion of processing capacity in each resource allocated to serve the tasks assuming each role. The simulation experiments have been conducted to verify the effectiveness of the developed allocation strategies. The experimental results show that the allocation strategy developed in this paper outperforms the traditional allocation strategies, which do not consider authorization constraints, in terms of both average response time and resource utilization.

## I. INTRODUCTION

This work considers a type of workflow in pervasive computing. These workflows are abstracted from the enterprise-level application in the business or commerce area. In a workflow of this type, some activities (or tasks) are run on computing resources, which are called Computing Tasks (CT) in this paper, and some need to be processed by human resources (i.e., the employees in a bank), which are called Human Tasks (HT). Nowadays, an employee in an enterprise that is involved in processing the human tasks in certain business processes is often equipped with a wireless device (e.g., iPad or smart phone), so that the activities can be

processed wherever the relevant employees are. When a human task needs the employee's attention, an alert or the task itself is sent to his or her wireless device and then the employee spends a certain amount of time to process it, for example, to read the relevant information related to the task and then make corresponding decisions. Human involvement introduces additional authorization concerns. Research has been conducted on the topic of security and authorization constraints in the workflow context [1][2][3][4]. Role-Based Authorization Control (RBAC), under which the users are assigned to certain roles while the roles are associated with prescribed permissions, is a popular authorisation control scheme applied to workflow executions [5].

Many workflow management strategies have been developed to enhance the performance of workflow executions [6]. When we design workflow management/scheduling strategies or plan resource capacities, it is often assumed that when a task is allocated to a resource, the resource will accept the task and start the execution once the processor becomes available [7][8][9]. However, the authorization policies impose further constraints on task executions and therefore, may incur performance penalty and affect both application- and system-oriented performance. The following example illustrates such a situation.

A bank will need both human tasks and computing tasks to support its business. A human task may involve a person (i.e., a user in the RBAC terminology) with an official position (i.e., a role in RBAC, e.g., a branch manager) signing a document; a computing task may involve running an application (often deployed as a service) on a computing resource to assess risk for an investment. Further, the computing applications may be hosted in a central resource pool (e.g. a cluster or a Cloud) [10], and the invocation of an application may be automated without human intervention, which we term an Automated Computing Task (ACT), or for security reasons, can only be initiated by a user with a certain role and be executed under that role/user, which we term a Human-aided Computing Task (HCT). The following authorization constraints are often encountered in such scenarios [11]: 1) Role constraints: A human task may only be performed by a particular role; a computing application may only be invoked by assuming a particular role; 2) Temporal constraints: A role

<sup>0</sup>Ligang He is the corresponding author

or a user is only activated during certain time intervals (e.g., a staff member only works morning hours); 3) Cardinality constraints: The maximum number of tasks (computing or other) running simultaneously under a role is  $N$ . It is common to find such authorization constraints and interaction between human and automated activities in other application domains such as healthcare systems [12], the manufacturing community [13][14], and so on. Human intervention and associated authorization clearly affects the processing of tasks and impacts on both application-oriented performance (e.g. mean response time of workflows) and system-oriented performance (e.g. utilization of the computing resource pool). For example, a task may have to wait in the waiting queue due to the temporal constraint and/or cardinality constraint of the role that the task is assuming, even if there are free resources in the system. Therefore, when planning resource capacities for workloads in such situations, we need to take into account the impact of authorization constraints.

The security-aware scheduling issues are also investigated in literature. The work in [15] developed a security overhead model for workflows and focussed on three security aspects: i) confidentiality, ii) integrity and iii) authentication. However, the work presented in this paper focuses on the security aspect of authorization. The work in [16] developed the security-aware resource allocation strategies for real-time DAG jobs in homogeneous clusters and heterogeneous cluster. The work in [17] proposed a Security-Aware Task (SEAT) graph model and based on this model, further developed the methods to achieve maximum security strength while satisfying the real-time constraints. Again, the security issues investigated in the above work do not include authorization. Moreover, those work do not consider human resources.

This paper investigates the issue of allocating both human resources and computing resources for running workflows under the role-based authorization control, so as to mitigate negative impact of authorization constraints on execution performance of workflows.

In the application domains of interest, the allocations of human resources and computing resources have different considerations. In the role-based authorization control, a human resource is affiliated with a role. The human resources with different roles will incur different salary costs (e.g., hiring a branch manager is more expensive than hiring a cashier). The budget is often a major factor of determining the allocation of human resources in enterprise applications. Therefore, this paper takes authorization constraints into account and develop an optimization method to allocate the proper amount of human resources for each role, so that the human tasks can achieve optimized performance subject to the budget limit for human resources.

Due to relatively low costs of computing resources, the cost is typically not a major concern for deploying low- or middle-end computing resources. When the workflows are running under authorization control, authorization constraints may incur performance penalty as discussed in the above workflow example in banks. Therefore, minimizing the overhead caused

by the authorization constraints should be a main objective. In order to address this issue, this paper develops a strategy of allocating computing resources. The strategy is able to calculate 1) a proper number of computing resources allocated to host each service, and 2) the processor sharing proportion in each resource allocated to run the tasks assuming a certain role.

In this paper, a workflow consist of human tasks and computing tasks. A computing task involves invoking a computing service hosted in a central resource pool (e.g., a cluster or a Cloud). It is assumed that the invocation of computing services can only be initiated by a user with a certain role. A human task is executed by a human resource with a certain role. A human task can also be regarded as invoking a human service provided by a user with a certain role. Therefore, we will discuss human tasks and computing tasks in a consistent manner in this paper.

It is assumed that a set of services (human service or computing service) is hosted by the resources (human resources or computing resources). A task (human task or computing task) in a workflow invokes one of the hosted services.

The rest of this paper is organized as follows. Section II presents the methods to calculate the arrival rate of the requests assigned to a role. Section III presents the method to allocate human resources, while Section IV develops the method to allocate computing resources for hosting computing services. The experimental studies are presented in Section V. Finally Section VI concludes the paper.

## II. CALCULATING THE ARRIVAL RATE UNDER AUTHORIZATION

In the workflow context in this paper, a task in a workflow invokes one of the services running on the resources. In order to determine the amount of resources allocated to host services, this section first calculates the arrival rate of tasks for each service, which is the invocation rate of each service when there is no authorization control. However, under the authorization control, the tasks have to be assigned to a role before they can invoke the services, and the roles may have temporal and cardinality constraints. Consequently, the services' invocation rates may be different from those when there is no authorization. This section derives the arrival rate of tasks for each role, i.e., the rate at which the tasks are assigned to each role under the authorization constraints. Table.I lists the notations used in the paper.

### A. Calculating the arrival rates for services

$\mathbb{S} = \{s_1, \dots, s_L\}$  denotes the set of services running on the resource pool.

$\mathbb{F} = \{f_1, \dots, f_N\}$  denotes the set of workflows, which has  $N$  types of workflow. Different types of workflow may have different topologies of tasks. A task in a workflow invokes one of the services in  $\mathbb{S}$ . A service invocation matrix, denoted as  $C_{L \times N}$ , can be used to represent which services are invoked by a workflow in  $\mathbb{F}$ . The matrix has  $L$  rows and  $N$  columns. Row  $i$  represents service  $s_i$ , while column  $j$  represents workflow

TABLE I  
NOTATIONS

notations	Explanation
$r_i$	role $i$
$e_i$	the execution time of the tasks assigned to $r_i$ .
$w_i$	the waiting time of the tasks assigned to role $r_i$
$np(r_i)$	the number of resources used to serve the tasks running under $r_i$
$rp_i$	the mean response time of the tasks running under role $r_i$
$\mathcal{C}^c(r_i)$	the cardinality constraint of $r_i$
$\mathcal{C}^t(r_i)$	the temporal constraint of $r_i$
$\mathcal{C}^r(s_i)$	the role constraint of service $s_i$
$\lambda^x(r_i)$	the arrival rate of the tasks that are assigned to $r_i$ when $x$ constraints are considered.
$\mathcal{C}^s(r_i)$	the set of services that role $r_i$ can invoke
$rp(r_i, s_j)$	the mean response time of the tasks that assume $r_i$ to invoke $s_j$

$f_j$ . An element  $c_{ij}$  represents how many times service  $s_i$  is invoked by workflow  $f_j$  (different tasks in a workflow may invoke the same service).  $\lambda_i$  denotes the arrival rate of Workflow  $f_i$ .

The arrival rate of the requests for service  $s_i$ , denoted as  $\lambda(s_i)$ , can be calculated from the service calling matrix,  $C_{L \times N}$ , as in Eq.1.

$$\lambda(s_i) = \sum_{j=1}^N (c_{ij} \times \lambda_i) \quad (1)$$

### B. Calculating the arrival rates for roles

This subsection analyse how to calculate the arrival rates for the roles under three types of authorisation constraints: role constraints, temporal constraints and cardinality constraints [5].

1) *Arrival rates under role constraints:*  $\mathbb{R} = \{r_1, \dots, r_M\}$  denotes the set of roles defined in the authorisation control system. The role constraint specifies the set of roles that are permitted to run a particular service.  $\mathcal{C}^r(s_i)$  denotes the role constraint applied to service  $s_i$ .

A role constraint matrix, denoted as  $O_{L \times M}$ , is used to represent which roles are permitted to invoke a particular service. The matrix has  $L$  rows and  $M$  columns. Row  $i$  represents service  $s_i$ , while column  $j$  represents role  $r_j$ . An element  $o_{ij}$  is 0 or 1, representing whether role  $r_j$  is permitted to run service  $s_i$ .

If only role constraints are considered and multiple roles are permitted to run a service, a role is randomly selected. In the requests for service  $s_i$ , the arrival rate of the requests allocated to role  $r_j$ , denoted as  $\lambda^r(s_i, r_j)$ , can be calculated using Eq.2. Further, the arrival rate of all service requests allocated to  $r_j$ , denoted as  $\lambda^r(r_j)$  can be calculated using Eq.3.

$$\lambda^r(s_i, r_j) = \begin{cases} \frac{\lambda(s_i)}{\sum_{j=1}^M o_{ij}} & \text{if } \sum_{j=1}^M o_{ij} \neq 0 \\ 0 & \text{if } \sum_{j=1}^M o_{ij} = 0 \end{cases} \quad (2)$$

$$\lambda^r(r_j) = \sum_{i=1}^L \lambda^r(s_i, r_j) \quad (3)$$

2) *Arrival rates under both role constraints and temporal constraints:* In most cases, a role is activated periodically. For example, the role of bank manager is only activated from 9am to 12pm in a day. Therefore, the temporal constraint of role  $r_i$ , denoted as  $\mathcal{C}^t(r_i)$  can be expressed as Eq.4, where  $P_i$  is the period,  $D_i$  is the time duration when  $r_i$  is activated in the period  $P_i$ , and  $S_i$  and  $E_i$  are the start and end time points when this period pattern begins and ends.  $E_i$  can be  $\infty$ , meaning the periodic pattern continues indefinitely. A temporal function for role  $r_i$  is defined in Eq.5. The value of the temporal function is 1 if the role is activated at the current time point  $t$ . Otherwise, the value of the function is 0.

$$\mathcal{C}^t(r_i) = (P_i, D_i, S_i, E_i) \quad (4)$$

$$ft(r_i, t) = \begin{cases} 1 & \text{if } t - \lfloor \frac{t-S_i}{P_i} \rfloor \times P_i \leq D_i \\ 0 & \text{if } t - \lfloor \frac{t-S_i}{P_i} \rfloor \times P_i > D_i \end{cases} \quad (5)$$

The function  $nr(s_i, t)$  defines the number of roles which are activated at time point  $t$  and are permitted to run service  $s_i$ .  $nr(s_i, t)$  can be calculated using Eq.6, which is based on the roles' temporal functions.

$$nr(s_i, t) = \sum_{r_j \in \mathcal{C}^r(s_i)} ft(r_j, t) \quad (6)$$

$\lambda^{rt}(s_i, r_j)$  denotes the arrival rate of the tasks that are requesting service  $s_i$  and are assigned to role  $r_j$  when both role constraints and temporal constraints are considered.  $\lambda^{rt}(s_i, r_j, t)$  denotes at time  $t$ , the arrival rate of the requests that assume  $r_i$  and invoke  $s_i$ .  $\lambda^{rt}(s_i, r_j, t)$  can be calculated as Eq.7. Then  $\lambda^{rt}(s_i, r_j)$  (i.e., the average arrival rate of the requests that assume  $r_i$  and invoke  $s_i$ ) can be calculated as Eq.8, where  $P$  is the minimal common multiple of the periods of all roles that can run  $s_i$ .

$\lambda^{rt}(r_j)$  denotes the arrival rate of all tasks that are assigned to role  $r_j$  when both role constraints and temporal constraints are considered.  $\lambda^{rt}(r_j)$  can be calculated as Eq.9.

$$\lambda^{rt}(s_i, r_j, t) = \frac{\lambda(s_i)}{nr(s_i, t)} \quad (7)$$

$$\lambda^{rt}(s_i, r_j) = \frac{\int_0^P \lambda^{rt}(s_i, r_j, t) dt}{P} \quad (8)$$

$$\lambda^{rt}(r_j) = \sum_{i=1}^L \lambda^{rt}(s_i, r_j) \quad (9)$$

Fig. 1 illustrates the temporal constraints of three roles,  $r_1$ ,  $r_2$ ,  $r_3$ , in which  $t(r_1) = (6, 4, 0, \infty)$ ,  $t(r_2) = (4, 2, 0, \infty)$ , and  $t(r_3) = (3, 1, 0, \infty)$ .

Fig. 2 illustrates  $nr(s_i, t)$  for the three roles in Fig.1. As can be seen from this figure, the number of activated roles that can run  $s_i$  varies over time. Note that since the minimal

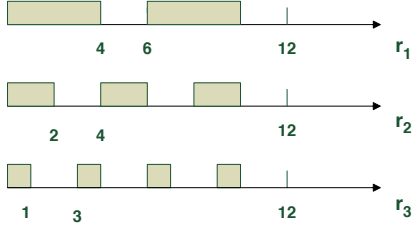


Fig. 1. An example of the temporal constraints of roles

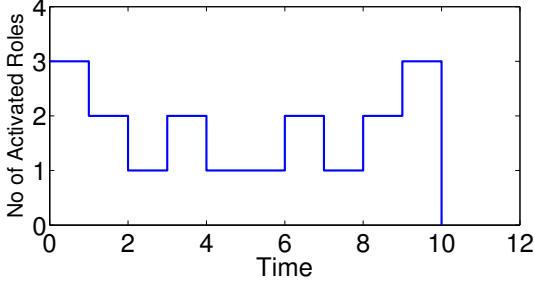


Fig. 2. The function of the number of activated roles for the example in Fig.1

common multiple of the periods of  $r_1, r_2, r_3$  is 12, the pattern of  $nr(s_i, t)$  will repeat in every time duration of 12.

According to Eq.8,  $\lambda^{rt}(s_i, r_j)$  is  $\frac{\lambda(s_i)}{3}$  and  $\lambda(s_i)$  at time point 0 and 12, respectively.

The analysis can be easily extended to the case where the temporal constraint of a role consists of multiple different periodic patterns, each of which is specified by Eq.4. The analysis for multiple periodic patterns is omitted in this paper.

3) *Arrival rates under both role constraints and cardinality constraints:* The cardinality constraint of a role is defined as the maximum number of tasks that the role can run simultaneously.  $C^c(r_i)$  denotes the cardinality constraint of  $r_i$ .

In order to avoid the execution delay caused by  $r_i$ 's cardinality constraint, the number of the tasks running under role  $r_i$  should be less than  $C^c(r_i)$  when a new task arrives requesting role  $r_i$ .  $\lambda^{rc}(s_i, r_j)$  denotes the arrival rate of the tasks that are requesting service  $s_i$  and are assigned to role  $r_j$  when both role constraints and cardinality constraints are considered.  $\lambda^{rc}(r_i)$  denotes the arrival rate of all tasks that are assigned to role  $r_j$  when both role constraints and cardinality constraints are considered.

According to Little's Law [18], we have Eq.10, where  $rp_i$  is the mean response time of the tasks running under role  $r_i$ .

$$C^c(r_i) = \lambda^{rc}(r_i) \times rp_i \quad (10)$$

$np(r_i)$  denotes the number of resources used to serve the

tasks running under  $r_i$ , and these resources are modelled as a  $M/M/np(r_i)$  queuing model.  $w_i$  denotes the waiting time of the tasks assigned to role  $r_i$ . According to the queuing theory [18],  $w_i$  can be calculated by Eq.11, where  $e_i$  is the execution time of the tasks assigned to  $r_i$ .

$$w_i = \frac{\lambda^{rc}(r_i) \times e_i^2}{np(r_i)^2 - e_i \times np(r_i) \times \lambda^{rc}(r_i)} \quad (11)$$

Since Eq.12 holds, Eq.10 can be transformed to Eq.13

$$rp_i = w_i + e_i \quad (12)$$

$$C^c(r_i) = \lambda^{rc}(r_i) \times \left( \frac{\lambda^{rc}(r_i) \times e_i^2}{np(r_i)^2 - e_i \times np(r_i) \times \lambda^{rc}(r_i)} + e_i \right) \quad (13)$$

$\lambda^{rc}(r_i)$  can then be calculated by transforming Eq.13 to Eq. 14 in page 5, which is the maximum task arrival rate that  $r_i$  can tolerate in order to avoid the overhead caused by its cardinality constraint.

4) *Arrival rates under role, temporal and cardinality constraints:*  $\lambda^{rtc}(r_i)$  denotes the arrival rate of the tasks that are assigned to  $r_i$  when the role constraints, temporal constraints and cardinality constraints are considered.  $\lambda^{rtc}(r_i)$  can be calculated as Eq.15.

$$\lambda^{rtc}(r_i) = \min(\lambda^{rt}(r_i), \lambda^{rc}(r_i)) \quad (15)$$

### III. ALLOCATING RESOURCES FOR HUMAN TASKS

Since a human task in a workflow invokes a human service provided by a user with a certain role, we need to allocate an appropriate amount of human resources for each role, so that the desired performance can be achieved for human tasks. In Section II, we have derived the tasks' arrival rates for roles under the authorization constraints. This section models the problem of allocating human resources for roles, aiming to optimizing the average response time of the human tasks. Since the budget is often a major factor in hiring human resources, the allocation of human resources is subject to a budget constraint.

$B$  denotes the budget that can be spent for human resources.  $b_i$  denotes the cost of hiring a human resource assuming role  $r_i$  (e.g., the salary for a staff taking the manager role).  $h_i$  denotes the number of the human resources allocated for role  $r_i$ . The budget constraint can be expressed as Eq.16, where  $h_i$  is an integer.

$$\sum_{i=1}^M (b_i \times h_i) \leq B \quad (16)$$

We model the human resources allocated for role  $r_i$  as an  $M/M/h_i$  queuing model. According to the queuing theory [18], the average response time of human tasks over all roles, denoted as  $RH$ , can be calculated by Eq.17.

$$RH = \sum_{i=1}^M \left( rp_i \times \frac{\lambda_i}{\sum_{j=1}^M \lambda_j} \right) \quad (17)$$

$$\lambda^{rc}(r_i) = \frac{np(r_i) \times \sqrt{(np(r_i) + \mathcal{C}^c(r_i))^2 - 4 \times \mathcal{C}^c(r_i) \times (np(r_i) - 1)} + np(r_i) \times (np(r_i) + \mathcal{C}^c(r_i))}{2 \times e_i \times (np(r_i) - 1)} \quad (14)$$

Following the similar derivation as in Eq.11 and Eq.12, Eq.17 can be transformed to Eq.18.

$$RH = \sum_{i=1}^M \left( \left( \frac{\lambda_i \times e_i^2}{h_i^2 - e_i \times h_i \times \lambda_i} + e_i \right) \times \frac{\lambda_i}{\sum_{j=1}^M \lambda_j} \right) \quad (18)$$

From the analysis in Subsection II-B3, we know that in order to reduce the performance penalty caused cardinality constraints, the tasks assigned to a role with a tighter cardinality constraint (i.e., less value of  $\mathcal{C}^c(r_i)$ ) should have a shorter average response time so that they can be turned around faster in the system. This relation can be represented in Eq.19.

$$rp_i \leq rp_j, \quad \text{if } \mathcal{C}^c(r_i) \leq \mathcal{C}^c(r_j) \quad (19)$$

The objective is to find  $h_i$  ( $1 \leq i \leq M$ ) subject to Eq.16 and Eq.19, such that  $RH$  in Eq.18 is minimized. This is a constrained-minimum problem, and there are existing solvers to find its solution [19].

#### IV. ALLOCATING RESOURCES FOR COMPUTING TASKS

A computing task in the workflow invokes a service hosted in the central computing resource pool (e.g., a Cluster or a Cloud [10]). This section aims to determine the suitable amount of computing resources allocated for hosting each service and for processing the tasks assuming each role, so that the overhead caused by the authorization constraints can be minimized.

$n_i$  denotes the number of homogeneous nodes used to host service  $s_i$ . According to the role constraints, we know which roles can invoke the services. Using Eq.8, we can calculate the arrival rate of the requests that assume  $r_j$  to invoke  $s_i$ . Applying Little's law, the desired average response time for a request assuming  $r_j$  (i.e.,  $rp_j$ ) can be calculated as Eq.20. In order to satisfy  $rp_j$ , we need to find a minimal number of nodes for hosting each service (i.e., the minimal value of  $n_i, 1 \leq i \leq L$ ), and to find the proportion of processing capability (in a node hosting  $s_i$ ) allocated to run the requests that assume role  $r_j$ , which is denoted as  $\alpha_{ij}$ .

$$rp_j = \frac{\mathcal{C}^c(r_j)}{\lambda^{rtc}(r_j)} \quad (20)$$

We first calculate the desired response time for the requests that assume  $r_j$  to invoke  $s_i$ .  $es_i$  denotes the mean execution time of the requests invoking service  $s_i$ , which can be obtained by benchmarking the executions of service  $s_i$ .  $rp(r_j, s_i)$  denotes the desired mean response time of the requests that assume role  $r_j$  to invoke service  $s_j$ . Then  $rp(r_j, s_i)$  can be calculated from Eq.21, where Eq.21.ii expresses that the ratio among  $rp(r_j, s_i)$  should be equal to the ratio among  $es_i$  ( $s_i \in \mathcal{C}^s(r_j)$ ).

$$\begin{cases} \sum_{s_i \in \mathcal{C}^s(r_j)} \left( \frac{\lambda^{rtc}(s_i, r_j)}{\lambda^{rtc}(r_j)} \times rp(r_j, s_i) \right) = rp_j & (i) \\ \forall s_i, s_k \in \mathcal{C}^s(r_j), rp(r_j, s_i) = \frac{es_i}{es_k} \times rp(r_j, s_k) & (ii) \end{cases} \quad (21)$$

The problem of finding  $\alpha_{ij}$  in a node hosting service  $s_i$  relies on the analysis of multiclass queueing systems with Generalized Processor Sharing, which is notoriously difficult [20]. The analysis of the multiclass single-server queue can be approximated by decomposing it into multiple single-class single-server queues with the capacity equal to  $\alpha_{ij}\mu_i$  [20], where  $\mu_i$  is the processing rate of a node for serving service  $s_i$  (i.e.,  $\frac{1}{es_i}$ ). Finding  $\alpha_{ij}$  and  $n_i$  can then be modelled as Eq.22, where Eq.22.i is constructed based on the equation of calculating average response time of the tasks in an M/M/1 queue [18]. In Eq.22, the number of unknown variables (i.e.,  $n_i$  and  $\alpha_{ij}, r_j \in \mathcal{C}^r(s_i)$ ) is the same as the number of equations in Eq.22. Therefore,  $n_i$  and  $\alpha_{ij}$  can be calculated.

$$\begin{cases} \forall r_j \in \mathcal{C}^r(s_i), \frac{\alpha_{ij}}{es_i} - \frac{\lambda^{rtc}(r_j, s_i)}{n_i} = \frac{1}{rp(r_j, s_i)} & (i) \\ \sum_{r_j \in \mathcal{C}^r(s_i)} \alpha_{ij} = 1 & (ii) \end{cases} \quad (22)$$

#### V. EXPERIMENTAL STUDIES

##### A. Experimental settings

This section presents the simulation experiments to demonstrate the effectiveness of the resource allocation strategies developed in this paper. The metrics used to measure the performance obtained by resource allocation strategies are mean response time of workflows and resource utilization.

The simulation program has the following components: capacity planner, workflow generator, authorization controller, workflow server.

The workflow generator generates the workflows in the following way. The workflows are randomly generated, each workflow containing TNUM tasks and each task in a workflow having the maximum of  $MAX\_DG$  children. A workflow contains two types of task, Human Task (HT) and Computing Task (CT), following a certain ratio of the number of tasks in each type (denoted as  $|HT| : |CT|$ ). A HT is processed by a user, while a CT involves one of  $A$  applications. The tasks execution times follow an exponential distribution. The human tasks have the average execution time of  $EX\_H$  time units, while the computing tasks, including HCT and ACT, have the average execution time of  $EX\_C$  units. Assume that all computing tasks can only be initiated by a user with a certain role (i.e., all computing tasks are human-aided computing tasks). RNUM roles and UNUM users are assumed to be involved in processing the workflows. The generated workflow instances will be issued following the Poisson process.

The authorization controller generates the authorization constraints as follows. The role constraints (i.e., the set of roles

that a task can assume) for each HT and CT are set in the following fashion. The simulation sets a maximum number of roles that any task can assume in the role constraints, denoted as  $MAX\_RCST$ , which represents the level of restrictions imposed on the role assignment for tasks. When setting the role constraint for task  $t_i$ , the number of roles that can run  $t_i$  is randomly selected from  $[1, MAX\_RCST]$ , and then that number of roles are randomly selected from the role set. A similar scheme is used to associate users to roles. The maximum number of users a role can be associated to is denoted as  $MAX\_U2R$ . The number of users belonging to role  $r_i$  is randomly selected from  $[1, MAX\_U2R]$ ; and these users are then randomly selected for  $r_i$  from the user set.

The temporal constraints on roles are set in the following way. For each role, a time duration is selected from a period of TD time units. The selected time duration occupies the specified percentage of the TD time units, which is denoted as TEMP. The starting time of the selected duration is chosen randomly from the range of  $[0, TD \times (1 - TEMP)]$ . For example, if TD=200 and TEMP=70%, the starting point is randomly selected from 0 to  $30\% \times 200$ .

CARD denotes the cardinality constraint, i.e., the maximum number of the tasks that can be run simultaneously in the system by a role.

Besides generating the authorization constraints, the authorization controller also enforces the authorization policy as the workflow instances arrive and request services.

Given the generated workflows and the authorization constraints generated by the authorization control component, the capacity planner calculates the capacity of human resources and calculate the capacity of computing resources (i.e., the number of computing resources) and the allocation strategy of computing resources (i.e., the processing sharing fraction for each role).

The workflow server run the generated workflows in the resources in the aforementioned fashion. The obtained performance is recorded. In the experiments, we also compare the performance obtained by our strategies with the performance by conventional strategies. Conventional capacity planning and resource allocation strategies do not take authorization constraints into account, and allocate the amount of resources proportional to the arrival rate.

### B. Experimental results

In order to demonstrate the effectiveness of the allocation strategy for human resources, we conduct the experiments using the traditional allocation strategy for human resources. In the traditional strategy, we don't impose authorization constraints, and assume a particular type of human tasks are handled by a particular user. Based on the arrival rate of workflows, we can obtain the arrival rate of the requests for each human service. The number of human resources allocated for handling each human service is proportional to the arrival rate of requests for each service, subject to the constraint that the total cost of hiring all human resources is no more than the budget  $B$ . With the same budget constraint, we conduct the

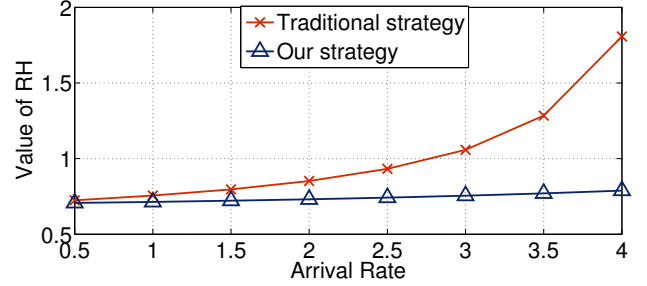


Fig. 3. Comparing average response time of human tasks between our strategy and the traditional allocation strategy for human resources; TNUM=15, MAX\_DG=10, EX\_H=7, RNUM=5, UNUM=15, A=15, MAX\_U2R=5, MAX\_RCST=4, CARD=4, TEMP=70%, TD=200, B=200,  $b_1, \dots, b_{RNUM} = 10, 8, 2, 5, 9$

experiments using the allocation strategy for human resources developed in this paper. Then we run the workflows consisting of only human tasks under authorization constraints on both resource allocation settings. Fig. 3 shows their performance in terms of mean response time (i.e., RH) as the arrival rate of the workflow increases.

As can be seen from Fig. 3, our strategy outperforms the traditional strategy in all cases and the trend becomes more prominent as the arrival rate of workflows increases. This is because our strategy takes into account authorization constraints and the arrival rate of requests, and establishes the optimization equations to calculate the allocation of human resources that can minimize the mean response time of human tasks. In the traditional allocation strategy, the resources are allocated only based on the arrival rate of the requests for services, not considering authorization constraints. Due to the existence of authorization constraints, the incoming requests need to be first assigned to roles and then invoke the corresponding services. Consequently, the rate at which the services are invoked under authorization may be different from that without authorization. Therefore, the resources allocated by the traditional strategy may not be in line with the resource demands, and consequently the performance may be impaired. Further, as the arrival rate of workflows increases, it becomes more likely that the following situation may occur under the traditional strategy due to the fact that the amounts of resources allocated for different services have to maintain the proportion: the resources allocated for some services become saturated while the resources are over-provisioned for other services due to the extra authorization constraints. In our strategy, however, the authorization constraints are taken into account, and the amount of resources for each role are calculated accordingly. The effect is that the cost spent for allocating over-provisioned resources is now used to allocate more resources that are saturated under the traditional strategy.

Fig. 4 compares resource utilizations between our strategy and the traditional strategy in the same experimental settings as in Fig. 3. As can be seen from Fig. 4, our strategy achieves



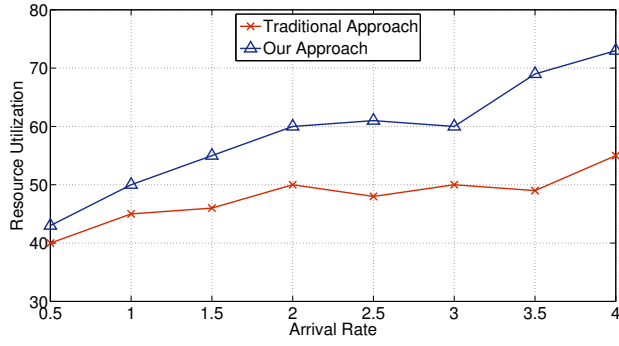


Fig. 4. Comparing resource utilization between our strategy and the traditional allocation strategy for human resources; the experimental settings are the same as in Fig.3

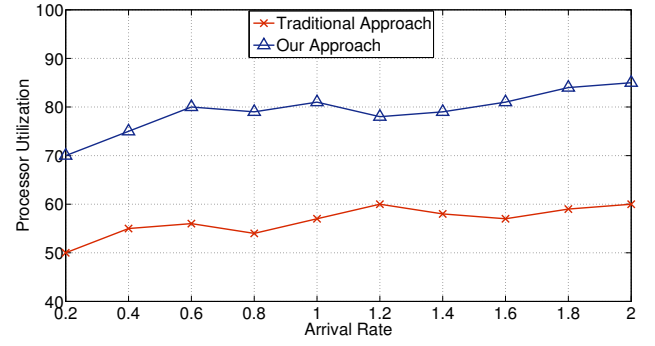


Fig. 6. Comparing resource utilization between our strategy and the traditional allocation strategy for computing resources; the experimental settings are the same as in the last figure

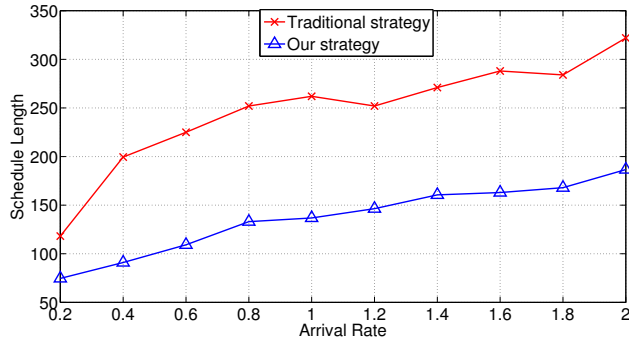


Fig. 5. Comparison of performance in terms of average response time between our allocation strategy and traditional strategy for computing resources; NUM=15, MAX\_DG=10, EX\_C=7, RNUM=5, UNUM=15, MAX\_U2R=5, MAX\_RCST=4, CARD=4, TEMP=70%, TD=200

higher utilization than the traditional strategy. This is still because the traditional strategy allocates resources based on the arrival rate of the requests for services, which causes the over-provisioned resources for some services after imposing authorization constraints.

In order to demonstrate the effectiveness of the allocation strategy for computing resources, we conduct the experiments using the traditional allocation strategy for computing resources. In our strategy, the authorization constraints are taken into account, and the proportion of processing capability allocated for each role is calculated accordingly. In the traditional strategy, all tasks are treated equally and are put into the central waiting queue in the cluster of computing resources. When a computing resource is free and the authorization constraints are satisfied, the task at the head of the waiting queue is put into execution in the free resource.

Fig. 5 compares average response time of computing tasks between our strategy and the traditional allocation strategy for computing resources. In these experiments, all tasks in a workflow are computing tasks. In the traditional resource allocation strategy, authorization constraints are not taken into account, and the amount of resources allocated for a service is proportional to the arrival rate of the requests for the service.

The allocation strategy developed in this paper calculates the arrival rate for each role and then further calculate the amount of resources allocated to serve the requests assigned to each role.

As can be seen from Fig. 5, our strategy performs better than the traditional strategy. This can be explained as follows. In our allocation strategy, the authorization constraints are taken into account. For example, if role  $r_i$  has the tighter cardinality constraint (i.e., smaller value of  $C^c(r_i)$ ), more proportion of processing capability will be allocated to serve the tasks assuming  $r_i$ , so that the number of those tasks in the system will be less and the performance penalty imposed by  $r_i$ 's cardinality constraint can be reduced. In the traditional allocation strategy, the tasks assuming different roles are treated equally, and therefore cannot prioritize the tasks that are assuming the roles with tight cardinality constraint and therefore should be turned around faster.

Fig. 6 compares the resource utilization between our strategy and the traditional allocation strategy for computing resources. It can be seen from this figure that our strategy can achieve higher resource utilization than the traditional strategy. This can be explained as follows. In the traditional strategy, it is more likely that the tasks have to wait in the waiting queue even if there are free resources in the system, because the tasks assuming the roles with tight cardinality constraints can be turned around faster in our strategy. This causes lower resource utilization.

Fig. 7 compares the schedule lengths of workflows achieved by our strategy and the traditional strategy. In these experiments, a workflow contains both human tasks and computing tasks. Then we run the workflows on human resources and computing resources allocated by our strategy as well as by the traditional strategy. Fig. 7 shows that our strategy achieves shorter schedule length than the traditional strategy. Again, this is because our strategy takes authorization constraints into account and allocate suitable amount of resources for both human resources and computing resources.

Fig. 8 compares the resource utilization achieved by our strategy and the traditional strategy. The depicted utilization is averaged over the entire system consisting of both human



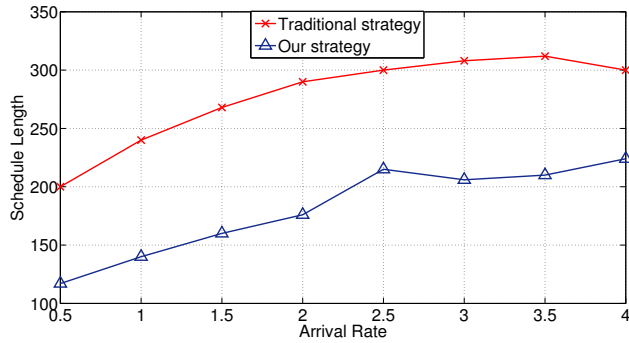


Fig. 7. Comparing the schedule lengths of workflows achieved by our strategy and the traditional strategy; NUM=15, MAX\_DG=10, EX\_C=7, EX\_H=7, RNUM=5, UNUM=15, MAX\_U2R=5, MAX\_RCST=4, CARD=4, TEMP=70%, TD=200,  $|HT| : |CT| = 4 : 6$ , B=200,  $b_1, \dots, b_{RNUM} = 10, 8, 2, 5, 9$

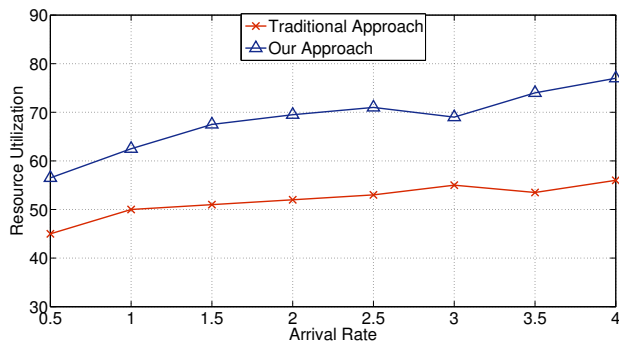


Fig. 8. Comparing average resource utilization achieved by our strategy and the traditional strategy; the experimental settings are the same as in last figure

resources and computing resources. The figure shows that our strategy can achieve higher system utilization than the traditional strategy. The reason for this is similar as explained in Fig. 6 and Fig. 4.

## VI. CONCLUSION

This paper investigates the issue of allocating resources for a type of workflows in pervasive computing. These workflows comprise both human and computing tasks. Human involvement introduces the authorization concerns. The authorization policies may incur performance penalty and affect both application- and system-oriented performance. The resource allocation strategies are developed in this paper for both human and computing resources. In the allocation strategy for human resources, the optimization equation is established by taking into account the authorization constraints, and also subject to the constraint of the budget available to hire human resources. Then the optimization equation is solved to obtain the number of human resources allocated to each authorization role. The allocation strategy for computing resources also takes into account authorization constraints, calculating not only the number of computing resources, but also the proportion of processing capacity in each resource allocated to serve the tasks assuming each role. The simulation experiments have

been conducted to verify the effectiveness of the developed allocation strategies. The experimental results show that the allocation strategy developed in this paper outperforms the traditional allocation strategies, which do not consider authorization constraints, in terms of both average response time and resource utilization.

## ACKNOWLEDGMENT

This work is supported by the Leverhulme Trust (grant number RPG-101).

## REFERENCES

- [1] V. Atluri and W. Kuang Huang, "A petri net based safety analysis of workflow authorization models," 1999.
- [2] L. He, K. Duan, X. Chen, D. Zou, Z. Han, A. Fadavina, and S. Jarvis, "Modelling workflow executions under role-based authorisation control," in *Services Computing (SCC), 2011 IEEE International Conference on*, July 2011, pp. 200–208.
- [3] Q. Wang and N. Li, "Satisfiability and resiliency in workflow authorization systems," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 4, pp. 40:1–40:35, Dec. 2010. [Online]. Available: <http://doi.acm.org/10.1145/1880022.1880034>
- [4] Y. Lu, L. Zhang, and J. Sun, "Using colored petri nets to model and analyze workflow with separation of duty constraints," *The International Journal of Advanced Manufacturing Technology*, vol. 40, pp. 179–192, 2009, 10.1007/s00170-007-1316-1. [Online]. Available: <http://dx.doi.org/10.1007/s00170-007-1316-1>
- [5] D. Zou, L. He, H. Jin, and X. Chen, "Crbac: Imposing multi-grained constraints on the rbac model in the multi-application environment," *Journal of Network and Computer Applications*, vol. 32, no. 2, pp. 402–411, 2009. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1084804508000520>
- [6] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," 2008.
- [7] C.-C. Hsu, K.-C. Huang, and F.-J. Wang, "Online scheduling of workflow applications in grid environments," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 860–870, 2011. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167739X10002086>
- [8] P. Delias, A. Doulamis, N. Doulamis, and N. Matsatsinis, "Optimizing resource conflicts in workflow management systems," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 23, no. 3, pp. 417–432, March 2011.
- [9] D. Chakraborty, V. Mankar, and A. Nanavati, "Enabling runtime adaptation of workflows to external events in enterprise environments," in *Web Services, 2007. ICWS 2007. IEEE International Conference on*, July 2007, pp. 1112–1119.
- [10] L. He, D. Zou, Z. Zhang, K. Yang, H. Jin, and S. A. Jarvis, "Optimizing resource consumptions in clouds," in *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, ser. GRID '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 42–49. [Online]. Available: <http://dx.doi.org/10.1109/Grid.2011.15>
- [11] X. Zhao, Z. Qiu, C. Cai, and H. Yang, "A formal model of human workflow," in *Proceedings of the 2008 IEEE International Conference on Web Services*, ser. ICWS '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 195–202. [Online]. Available: <http://dx.doi.org/10.1109/ICWS.2008.14>
- [12] M. Stuit, H. Wortmann, N. Szirbik, and J. Roodenburg, "Multi-view interaction modelling of human collaboration processes: A business process study of head and neck cancer care in a dutch academic hospital," *J. of Biomedical Informatics*, vol. 44, no. 6, pp. 1039–1055, Dec. 2011. [Online]. Available: <http://dx.doi.org/10.1016/j.jbi.2011.08.007>
- [13] T. Hara, T. Arai, Y. Shimomura, and T. Sakao, "Service cad system to integrate product and human activity for total value," *CIRP Journal of Manufacturing Science and Technology*, vol. 1, no. 4, pp. 262–271, 2009, jce:title;Life Cycle Engineering;fce:title;. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1755581709000078>

- [14] J. Y. Choi and S. Reveliotis, "A generalized stochastic petri net model for performance analysis and control of capacitated reentrant lines," *Robotics and Automation, IEEE Transactions on*, vol. 19, no. 3, pp. 474 – 480, June 2003.
- [15] T. Xie and X. Qin, "Scheduling security-critical real-time applications on clusters," *Computers, IEEE Transactions on*, vol. 55, no. 7, pp. 864–879, 2006.
- [16] —, "Security-aware resource allocation for real-time parallel jobs on homogeneous and heterogeneous clusters," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, no. 5, pp. 682–697, 2008.
- [17] M. Qiu, L. Zhang, Z. Ming, Z. Chen, X. Qin, and L. T. Yang, "Security-aware optimization for ubiquitous computing systems with seat graph approach," *Journal of Computer and System Sciences*, 2012.
- [18] L. Kleinrock, *Queueing Systems*. Wiley Interscience, 1976, vol. II: Computer Applications.
- [19] E. Cuervo, A. Balasubramanian, D. ki Cho, A. Wolman, S. Saroiu, R. Ch, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *In Proceedings of ACM MobiSys*, 2010.
- [20] Z. Liu, M. S. Squillante, and J. L. Wolf, "On maximizing service-level-agreement profits," *SIGMETRICS Perform. Eval. Rev.*, vol. 29, no. 3, pp. 43–44, Dec. 2001. [Online]. Available: <http://doi.acm.org/10.1145/507553.507571>