

# Security-aware Resource Optimization in Distributed Service Computing

Kaiqi Xiong and Harry Perros  
Department of Computer Science  
North Carolina State University  
Raleigh, NC 27965-7534, USA  
{xiong, hp}@csc.ncsu.edu

## Abstract

This chapter considers a set of computer resources used by a service provider to host enterprise applications for customer services subject to a service level agreement (SLA). The SLA defines three QoS metrics, namely, trustworthiness, percentile response time and availability. We first give an overview of current approaches, solutions and challenges in the security-aware resource optimization problem. Then, we present a framework for solving the problem. We further propose an approach for resource optimization in such an environment that minimizes the total cost of computer resources used by a service provider for such an application while satisfying all these three QoS metrics in a security-aware resource optimization problem that typically arises in distributed service computing. We formulate the security-aware resource optimization problem as an optimization problem under the SLA constraints, and solve it using an efficient numerical procedure. Finally, we conclude our discussion and provide the research directions for future study.

## Introduction to Secure-aware resource optimization

With the number of e-Business applications is dramatically increasing, service level agreement (SLA) will play an important part in distributed service computing. An SLA is a combination of several qualities of services (QoS), such as security, performance, availability, and reliability, agreed between a customer and a service provider. The service provider may be a telecommunications carrier, an Internet service provider, or any company that provides outsourcing services. The services provided may include dedicated leased lines, shared packet-oriented services, Web hosting services, off-site application management, and off-site network management. With the ubiquity of mobile devices such as smartphones and PDAs, mobile devices will generate a large percentage of Web service requests.

An SLA defines all aspects of the service being provided. It generally consists of security, performance and availability. *Security* can be categorized as *identity security* and *behavior security*. Identity security includes the authentication and authorization between a customer and a service provider, data confidentiality and data integrity. Behavior security describes the trustworthiness among multiple resource sites, and the trustworthiness of these resource sites by customers, including the trustworthiness of computing results provided by these sites. *Performance* includes the two following aspects.

- *Response time* is the time for a service request to be satisfied. That is, this is the time it takes for a service request to be executed on the service provider's multiple resource sites.
- *Throughput* is the service rate that a service provider can offer. It is defined by the maximum throughput or by the undergoing change of throughput with service intensity.

Finally, *availability* is the percentage of time that a service provider can offer services.

In this chapter, we shall discuss secure-aware resource optimization in Web service application under SLA guarantees. We first give an overview of existing models and techniques in the research that addresses only one of these QoS metrics but is reluctant to deal with all of them together due to the complexity of these metrics. In the case of the response time as a performance metric, the average time to process and complete a job is typically used in the literature. Then, we present our model and its challenge. For example, an average time may not be of real interest to a customer. A statistically bounded metric, that is, percentile response time, is more realistic than the average response time. Specifically, we define and solve a security-aware resource optimization problem that occurs in Web service applications, subject to the constraints of trustworthiness, a percentile response time and availability.

Figure 1 depicts a framework for this security-aware resource optimization problem.

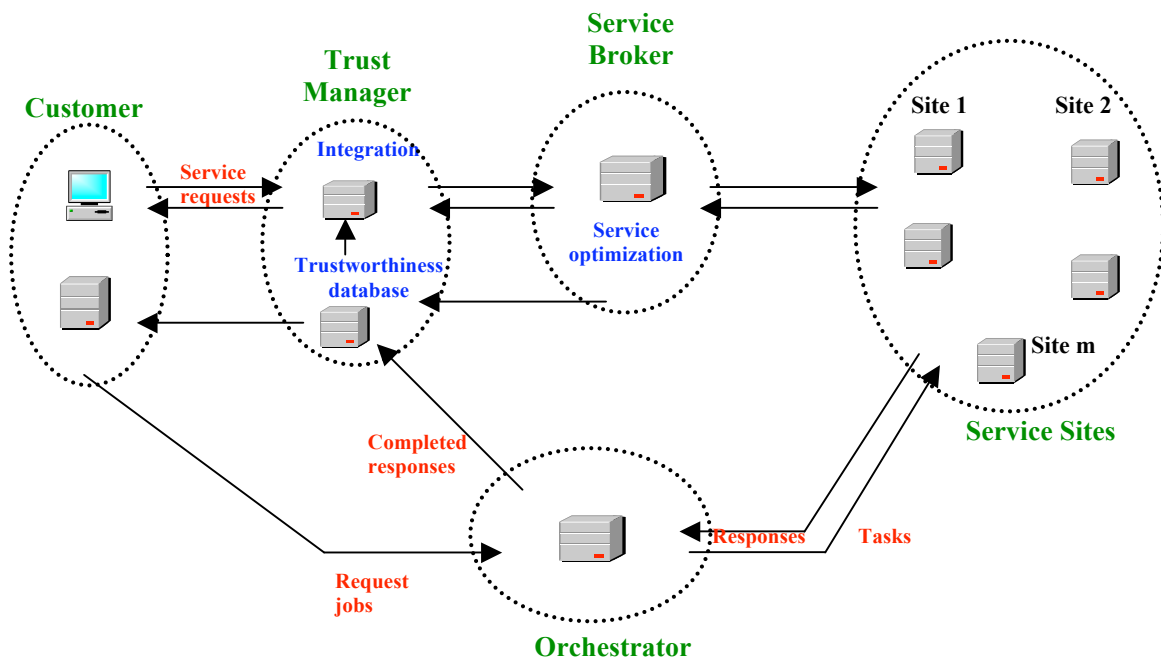


Figure 1: An SLA-based Web Service Model

A customer represents a business that generates a stream of service jobs issued at a specified rate, which require a certain quality of service. A trust manager, who represents

the customer, negotiates a SLA with a service broker who represents resource sites (alternatively called service sites)  $S_1, S_2, \dots, S_M$  where  $M > 0$ . The trust manager checks the trustworthy information of the resource sites and selects  $m$  ( $0 < m \leq M$ ) say sites 1, 2, ...,  $m$  of those sites which meet pre-defined trustworthy requirements for serving the service jobs. These  $m$  service sites are then optimized by the service broker in terms of service costs at these sites so that the service jobs can receive the requested quality of service. The trust manager monitors the trustworthiness of these service sites and if necessary, it may replace one or more if their trustworthiness index drops below a pre-defined level. Specifically, service jobs are sent directly to the orchestrator who is responsible for initiating a sequence of tasks necessary for the execution of a service job. The result is sent back to the trust manager who forwards it to the customer, after it checks its trustworthiness and updates its database. The customer also sends feedback to the trust manager who uses it to update its trustworthiness information. This framework will be further described in detail in this chapter later. We will formulate the security-aware resource allocation problem as an optimization problem under SLA constraints in which we calculate the number of servers in each service site that minimize a cost function that reflects operational costs for customer services. We will provide a solution to this problem using an efficient procedure and an illustrative example to briefly demonstrate how to use the procedure.

The rest of this chapter will mainly focus on the following three aspects: (1) Current approaches, solutions and challenges; (2) Our proposed solution and practical implication; and (3) Conclusions and future directions.

## **Current Approaches, Solutions and Challenges**

The issue of service quality has been extensively investigated. Characterizing QoS metrics is essential but challenging in the security-aware optimization problem.

The first QoS metric is trustworthiness. "Trust" is used to deal with the notion of the trustworthiness in this research. In this chapter, trust is defined as a firm belief in the competence of a resource site that acts as expected. Many researchers have studied the trustworthiness of service entities, and suggested several different trust metrics. Zhang et al. (2004), and Ziegler & Lausen (2004) classified various trust metrics. Vu et al. (2005) proposed a new QoS-based semantic Web service selection and ranking solution using a trust and reputation management and assuming known QoS qualities. Azzedin & Maheswaran (2002) studied trust management in Grid computing.

Furthermore, availability is a critical metric in today's computer design. A computer system is unavailable due to a variety of causes, such as network, hardware, software, and operational failures, security attacks, and so on. Detecting and preventing these failures and attacks is beyond the scope of our discussion in this chapter. Cisco has asserted that the operational failure causes 80% of non-availability (see CISCO (2004)). Hence, increasing network availability is becoming a key priority for enterprise and service provider organizations. Brown & Patterson (2000) defined a new availability metric to capture the variations of the system quality of service over time. It is defined by the

number of requests satisfied per second (or the latency of a request service) and the number of server failures that can be tolerated by a system.

Among the QoS metrics, the response time is the most important one. The computation of the response time has been extensively studied for a variety of computing systems. However, only the average response time is calculated rather than a percentile of the response time. Levy et al. (2003) presented a performance management system for cluster-based Web service where the average response time is used as a metric. Chandra (2003) employed an online measurement method, and considered a resource optimization problem based on measured response times. In order to compute a percentile of the response time one has to first find the probability distribution of the response time. This is not an easy task in a complex computing environment involving many computing nodes. Xiong & Perros (2006b) developed the efficient and accurate solutions of the percentile of the response time for both a tandem service model and a feedback service model.

Web service is often contracted through SLAs that typically specify a certain QoS in return for a certain price. Although QoS was not defined in the initial UDDI standard for Web services, many studies have been carried out to extend the initial UDDI, such as WSLA (Keller & Ludwig (2003)), WSOL (Tosic et al. (2002)), and QML (Dobson (2002)). The issue of a reputation-based SLA has been studied by Jurca & Faltings (2005) where the cost is determined by the QoS that was actually delivered. Menasce & Almeida (2002) discussed the basic metrics and models in Web service resource management.

Resource provisioning problems in distributed systems have also been widely studied based on either trustworthiness or response time. For example, Menasce & Casalicchio (2004) used the average response time as their QoS metric in studying Grid resource management. A collection of such papers can be found in Nabrzysbi et al. (2004).

Our challenges are that we need to consider a resource optimization problem subject to all three QoS metrics, expressed by trustworthiness, percentile response time and service availability, and to provide the accurate calculation of these three QoS metrics in a variety of distributed service computing models.

## **Our Proposed Solution and Practical Implications**

This part discusses a framework for solving the security-aware resource optimization problem in detail, our algorithm for solving the optimization problem and its practical implications.

### **A Framework for Solving the Security-aware Resource Provisioning Problem**

Services components from several universal service providers can be flexibly integrated into a composite service with cross-language and cross-platform regardless of their location, platform, execution speed, and process. Delivering quality services to meet

customer's requirement under SLA is very important and challenging due to the dynamical and unpredictable nature of Web service applications. In the part, we propose a Web service framework for solving the Security-aware resource optimization problem as shown in Figure 2. The framework consists of a customer, a trust manager, a service broker and a service processor. The functions of these service entities are explained as follows.

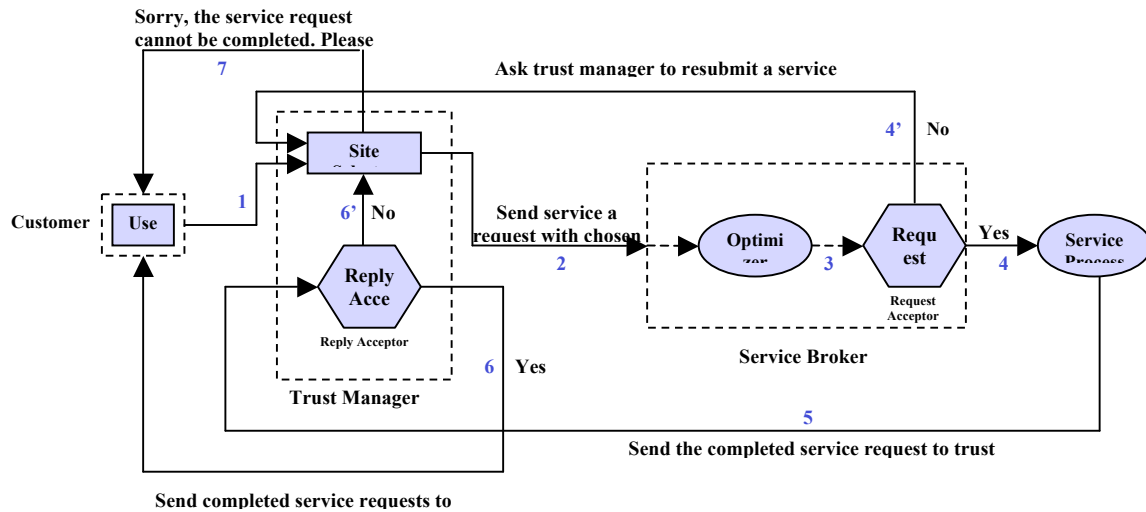


Figure 2: A Framework for Solving the Security-aware Resource Provisioning Problem

- The customer represents a business that negotiates a contract for particular Web services with a service broker and submits a service request to be processed by a number of resource sites.
- The customer consists of a number of business end-users (simply called users), and the service request defines service jobs generated by the users at a given rate, and QoS requirements with a fee.
- The trust manager is an entity that plays in the following roles:
  - **Service integration.** According to the customer's request, it integrates services chosen from a number of resource sites.
  - **Trust monitoring and determination.** It monitors and determines the trustworthiness of service sites.
  - **Service selection.** It chooses service sites that meet trust level requirements predefined by the customer.
  - **Service orchestration.** It defines the sequence and conditions in which one Web service invokes other Web services in order to realize some useful function. That is, it defines the pattern of interactions that service components must follow in order to process the customer's requests.

Based the customer request, the trust manager generates a new service request that contains a set of service sites selected by the trust manager.

- The service broker is an entity that represents service sites. After it receives the trust manager's service request, the service broker optimizes the number of service resources in each service site to ensure the customer's QoS requirements. The service resource could be a piece of software, hardware, or both and it plays a key role in completing the customer's request. In this chapter, we consider it as a hardware device, such as a blade, a CPU, a storage device, a disk, a router and so on.
- The service processor is an entity that manages and instructs service sites to process the customer request based on the trust manager's service request. The service sites are entities that provide specific services. A single or multiple service providers may own these service sites. Each site consists of a number of service resources managed by a service manager.

Furthermore, we outline the workflow for an approach to solving the security-aware resource optimization problem in the framework as shown in Figure 2.

The workflow can be presented in the following steps.

**Step 1:** A customer submits a service request to the trust manager who represents the customer. Let us recall that the customer represents a business that consists of a number of users within this business, and the service request consists of a number of service jobs generated by users.

**Step 2:** The site selector who is part of the trust manager selects service sites with the trust indices that meet the customer's trust requirement. Then, the trust manager submits the customer request to the service broker.

**Step 3:** An optimizer who is part of the service broker runs an optimization algorithm to find the number of servers required at each resource site to ensure the customer's SLA guarantee. Then, it will decide whether to accept the service request based on the profitability of the service broker.

**Step 4:** If the service broker can make an acceptable profit with a service provider who owns these chosen service sites, then the service request is accepted. Subsequently, the service request is submitted to a service processor whereby these chosen service sites process the service request based on a rule defined in the trust manager's request. If the service request is not accepted, go to Step 4'. The service processor is an entity that manages service sites.

**Step 5:** After these chosen service sites finish the service request, the service processor sends the completed service request back to the trust manager. Then, the reply acceptor who is part of the trust manager will decide whether to accept the service reply based on the trust indices of these chosen sites at the time when it receives the response from the service processor.

**Step 6:** If the trust indices of these chosen sites meet the customer's requirement, then the service reply is accepted. Subsequently, the completed service request is forwarded to the customer.

**Step 6':** If the trust indices of these chosen sites do not meet the customer's requirement, then the service reply cannot be accepted. Subsequently, the service acceptor forwards the service request to the site selector and asks it to resubmit the service request.

**Step 4'**: If the service broker cannot make an acceptable profit with a service provider who owns these chosen service sites, then the service request cannot be accepted. Thus, the service broker notifies the trust manager to resubmit the service request by re-selecting service sites.

**Step 7**: After the number of resubmissions surpasses a threshold predefined by the trust manager or the service broker, the trust manager will notify the customer that its request cannot be completed. Then, the customer needs to either abort the service request or modify the QoS requirement with a fee accordingly.

**Remarks:**

- The trust manager and the service broker represent the customer and the chosen service sites respectively. Hence, we assume that they only receive service commissions from the customer and the service processor respectively based on the number of completed services. In general, the trust manager may be a business entity that not only receive a service commission from the customer, but also make money since it may work with the service broker to find service sites that meet predefined trust requirements but also the chosen service sites have an overall lower fee.
- The trust manager selects service sites on behalf of its customer. Hence, it does not care about how much cost is needed to process the customer's request. But, if the number of service resubmissions surpasses a threshold predefined by the trust manager or the service broker, then the trust manager will notify the customer that her/his request cannot be completed, as mentioned in Step 7.
- If the trust manager constantly resubmits a service request to the service broker, then the service request submissions constantly consume the service broker's resource. Thus, a denial of service attack may occur when an attacker impersonates the trust manager to keep submitting bogus service requests to the service broker. We do not analyze such an attack here since it is beyond the scope of our discussion in this chapter.
- The above steps can be regarded as an SLA negotiation process between the trust manager and the service broker. The process is often finished before service delivery. That is, the SLA is static. However, the SLA can be dynamically changed as well. In this case, the SLA will be periodically verified and/or negotiated. The length of the period of time for the verification and/or negotiation depends on individual service types. It may be a week, or a month.

**An Approach for Solving the Security-aware Resource Optimization Problem**

As we saw above, we have proposed a security-aware Web service model, and provided a Web service framework for studying the model. The framework consists of a customer, a trust manager, a service broker, and a service processor. In this framework, an optimizer within the service broker is an important key component. It calculates the number of service resources required to ensure that the response time of a service request meets the requirement of a predefined percentile response time under a given fee. The security-aware resource optimization problem can be constructed by minimizing the total cost of

service providers while satisfying SLA guarantees. It will be formulated as a mathematical minimization problem later.

In this part we first use a queueing network to model  $m$  service sites, formulate the secure-aware optimization problem as three sub-problems. Then, we demonstrate how to develop an algorithm to solve the security-aware resource optimization problem.

Without any confusion, we reuse  $m$  ( $0 < m \ll M$ ) as the number of resource sites necessary for processing a customer's service job. Assume that the trust manager first selects  $m$  resource sites from the  $M$  ones. We only consider the case of single customer services in this chapter. Our discussion below can be easily extended to the case of multiple priority customer services.

We can model those  $m$  resource sites as a queueing network, for example, the one as shown in Figure 3. Without loss of generality, we assume that the first  $m$  sites are chosen. In Figure 3, the tandem queueing network consists of a trust server and  $m$  stations numbered sequentially from 1 to  $m$ . The trust server represents the trust manager, and each station represents a resource site. Each station carries out a particular function, such as a database server, a computing server, a file server, or a web server. The execution procedure of a service request may be complicated in the real world, but the main idea of our proposed modeling approach can be extended to describe any collaborative relationship among resource sites as long as the relationship can be quantified.

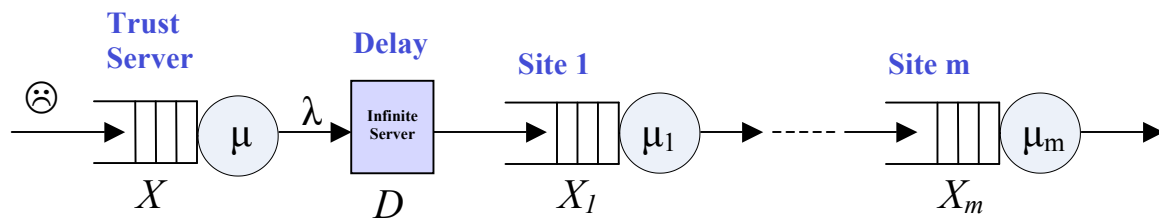


Figure 3: A Web Service Computing System for Single-class Customer

In Figure 3, each station  $j$  is modeled as a single FIFO queue served by  $n_j$  identical servers, each providing a service at the rate  $\mu$ . Let  $\lambda$  be the external arrival rate to the infinite Server, and let  $\lambda$  and  $\lambda_j$  be the effective arrival rates to the infinite server and station  $j$  ( $j=1, 2, \dots, m$ ). The notion of server here is defined as a service resource at each site that processes users' jobs. For example, as we mentioned early, it could be a blade, a CPU, a disk, a storage device, and so on. We assume that all service times are exponentially distributed and the external arrival to the trust server occurs in a Poisson fashion. The trust server provides a service at the rate  $\mu$ .

In Figure 3, the infinite server represents the total propagation delay from the user to the service provider and back and also from station 1 to  $m$ . We only consider a single class of customer in this research.

In this research we are interested in minimizing the overall cost of the above Web service computing system so that the desired SLA is guaranteed. Before presenting an algorithm for solving the minimization problem, we first need to calculate the response time of a customer's service request. Recall that the response time refers to the time elapsed from the moment a customer's service job joins the computing system to the time it departs. That is, the response time of a service request is equal to a sum of the processing time of the trust manager and the execution time of chosen service sites for the customer request. It is the most important QoS metric. The response time also reflects security behavior and the availability of services in some degree.

In our approach we considered the percentile response time that may better address a customer's concern as opposed to the average response time that is commonly used in the literature. Let  $T$  be a random variable that represents the response time of a service job. Also, assume that  $f_T(t)$  and  $F_T(t)$  be the probability and cumulative distributions of  $T$  respectively. Denote the overall service cost of processing a service request by

$$g(n_1, n_2, \dots, n_m) = \sum_{j=1}^m n_j c_j \quad (1)$$

where  $n_1, n_2, \dots, n_m$  are the number of servers allocated to a specific stream of jobs with arrival rate  $\lambda$ , each server with the costs of  $c_1, c_2, \dots, c_m$ .

Then, the security-aware resource optimization problem can be formulated as the following three sub-problems:

**Sub-problem 1:** Select  $m$  resource sites within a predefined trust index at time  $t=t_k$

**Sub-problem 2:** Solve for  $n_j$  in the  $m$ -dimensional integer optimization problem:

$$\min_{n_1, n_2, \dots, n_m} g(n_1, n_2, \dots, n_m) \quad (2)$$

Under the constraint of a percentile response time:

$$F_T(t = T^D) \geq \gamma\% \quad (3)$$

and the constraint of service availability for service sites  $j=1,2,\dots,m$ :

$$P_j(n_j, N_j) \geq \delta_j\% \quad (4)$$

where  $T^D$  is a desired response time defined by a customer,  $\gamma\%$  is a desired percentage of the response time, and  $\delta_j\%$  is a desired percentage of service availability at site  $j$ .  $P_j(n_j, N_j)$  is the probability that there are at least  $n_j$  servers available among  $N_j$  servers at service site  $j$ .

**Sub-problem 3:** Update the trust indices of all  $M$  sites based on the activity during the time interval  $[t_k, t_k + T^D]$ . Then, the trust manager decides if a completed job is accepted. If each trust index at those selected sites that complete the service job meets

a redefined index value, then the completed job is accepted. Otherwise, then the completed job is discarded and the trust manager needs to resubmit the job.

Note that in the model the verifying time of a trust index is ignored since we are interested in the processing time of a job at resource sites. While Sub-problems 1 and 3 are solved at the customer side to ensure a customer service received from reliable resource sites, Sub-problem 2 is solved within the domain of the service broker who represents these resource sites. Hence, in the current model the trustworthiness of resource sites in Sub-problems 1 and 3 are not considered as a constraint for the optimization problem in Sub-problem 2.

To solve Sub-problems 1 and 3, we need to constantly update the trust index of each service site. We consider the discrete time  $t_1, t_2, \dots, t_k, \dots$  in an increasing order ( $k=1,2,\dots$ ). For each site  $j$ , its trust index at time  $t_k$  is defines as a weighted sum of its trust index at time  $t_{k-1}$  and a satisfactory ratio for services completed at time between  $t_{k-1}$  and  $t_k$ . The satisfactory ratio is equal to the number of satisfactory services assessed by customers (i.e., through customer's feedback) divided by the total number of service submissions from time  $t_{k-1}$  to  $t_k$ . Its detailed discussion can be found in Xiong & Perros (2006a).

The above resource optimization problem as described in Sub-problem 2 is to find a minimal cost presented in (2) such that  $\gamma\%$  of the time a customer's response time is less than a predefined value  $T^D$ , and  $\delta_j\%$  of time the selected resource sites have  $n_j$  servers available for the job.

Furthermore, to solve the minimization problem as described in Sub-problem 2, we need to derive the calculation of the cumulative distribution of response time  $T$ ,  $F_T(t= T^D)$ , for inequality (3) and the probability  $P_j(n_j, N_j)$  for inequatlity (4) respectively.

First, a two-state Markov chain with the states ``up" and ``down" is used to study the service availability at each service site. Assume that each server fails at a rate of  $a_j$ , and recovers (i.e., it is put back into operation) at a rate of  $b_j$  at site  $j$ . That is, the failure rate  $a_j$  is the state of transition from ``up" to ``down," and the recovery rate  $b_j$  represents the rate of transition from ``down" to ``up." Thus, the probability that  $i$  servers are down at site  $j$  is given by

$$p_j^i = \frac{N_j!}{i!(N_j - i)!} \eta_j^i p_j^0 \quad i=1, \dots, N_j$$

where  $\eta_j = \frac{a_j}{a_j + b_j}$  is the server unavailability rate, and  $p_j^0$  is given by

$$p_j^0 = [N_j! \sum_{i=0}^{N_j} \frac{\eta_j^i}{i!(N_j - i)!}]^{-1}$$

Thus, the probability that at least  $n_j$  servers are available at site  $j$  is:

$$P_j(n_j, N_j) = p_j^0 \sum_{i=0}^{N_j - n_j} \frac{N_j \eta_j^i}{i!(N_j - i)!} \quad (5)$$

Second, the complexity of the calculation for  $F_T(t = T^D)$  is determined by the workload of service requests. When the workload can be modeled as the queueing network depicted in Figure 3, the Laplace transform of the probability distribution of the total response time among  $m$  service sites is a product of each Laplace transform of such a probability distribution at each service site. Subsequently, we can derive the calculation of the probability and cumulative distributions of the total response time among  $m$  service sites. Their detailed calculation can be found in Xiong & Perros (2006b).

To ensure the fairness of all service sites, we assume that each service site as depicted in Figure 3 is equally utilized. Then, Sub-problem 2 in the security-based resource optimization problem is solved using the following iterative algorithm.

### Algorithm

1. Find the minimal number of servers at each service site such that the constraint of a percentile response time given in (3) is satisfied, denoted as  $n_j(T)$ .
2. Find the minimal number of servers at service site such that the constraint of service availability given in (4) is satisfied, denoted as  $n_j(a)$ .

Then, the minimal number of servers at each service site is  $n_j = \max \{n_j(T), n_j(a)\}$  required to ensure that both constraints of a percentile response time and service availability are guaranteed, where  $j=1,2,\dots,m$ .

**Algorithm's practical implications:** The key advantage of the algorithm is that we only need to seek for the minimal number of servers required to ensure one of two constraints at each service site. The algorithm reduces the  $m$ -dimensional optimization into two one-dimensional minimization problems, which greatly reduces the computational cost of solving the security-aware optimization problem. Thus, our proposed solution can be easily realized in real-time applications.

### Example

As derived in Xiong and Perros (2006b), the cumulative distribution of the response time  $T$  for the service model described as a queueing network depicted in Figure 3 is

$$F_T(t) = L^{-1}\left(\frac{\mu(1-\rho)}{s + \mu(1-\rho)} \cdot \frac{\lambda}{s(s + \lambda)} \cdot \frac{\hat{a}}{(s + \hat{a})^m}\right) \quad (6)$$

where  $\hat{a} = \psi(n_j)\mu_j(1 - \rho_j)$ ,  $\rho = \frac{\lambda}{\mu}$ ,  $\rho_j = \frac{\lambda}{\mu_j}$ ,  $\psi(n_j) = \xi^{\log_2 n_j}$ , and  $\boxtimes$  is a server basic scaling factor from one to two servers for  $j=1,2,\dots,m$ .  $L^{-1}$  represents an inverse Laplace transform.

Let's consider a seven-resource site example. For simplicity, all parameters below are unitless. We choose  $\boxtimes=1.5$ ,  $\bullet=100$ ,  $T^D=0.16$ ,  $\gamma_p=97.5$ ,  $\circ=200$ ,  $\underline{\rho}_j=99.999$ ,  $c_j=2$ , and  $N_j=200$ . The service rates of seven service sites are given by  $\circ_1=18$ ,  $\circ_2=80$ ,  $\circ_3=\circ_7=35$ ,  $\circ_4=41$ ,  $\circ_5=15$ , and  $\circ_6=25$ .

By using Steps 1 and 2 the above algorithm, we can find the minimal numbers of servers for service sites 1 to 7 are 62, 5, 20, 16, 84, 35 and 20 necessary to ensure 97.5% response time guarantee, and 10, 7, 22, 18, 15, 23, 10 necessary to ensure 99.999% service availability guarantee, respectively. Thus, the minimal numbers of servers for service sites 1 to 7 are 62, 7, 22, 18, 84, 35 and 20 necessary to ensure both 97.5% response time and 99.999% service availability guarantees.

When each service site is equally utilized, the accuracy of our proposed algorithm for solving the security-aware optimization problem depends on the quality of the obtained cumulative distribution given in (6). For this purpose, we simulated the above queueing network model in Arena 10.01, and compared the simulation result with the approximate result based on the formula (6). Figure 4 gives their relative error % that was calculated by

$$\text{Relative error \%} = \frac{\text{Approximation Result} - \text{Simulation Result}}{\text{Simulation Result}} \times 100$$

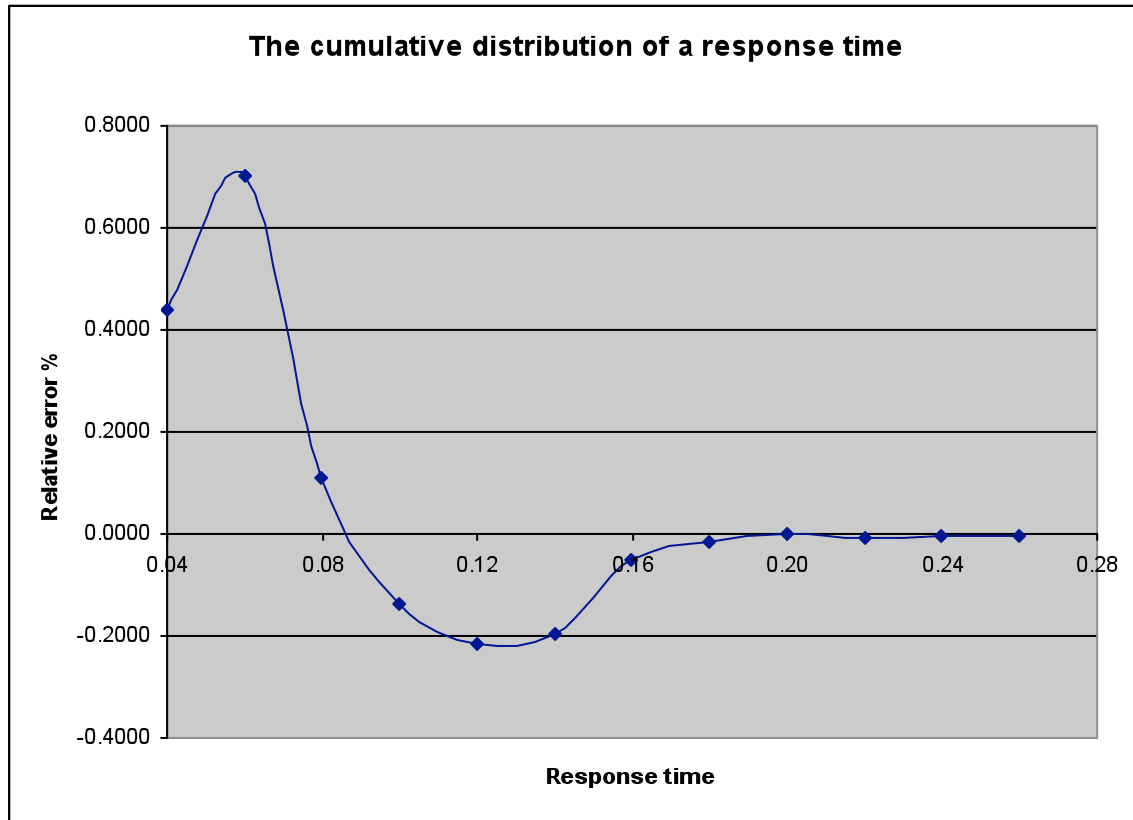


Figure 4 validates that the approximate result is very close to the simulation one. Moreover, the exact optimal number of servers, obtained by exhaustive search using the simulation model, is consistent with the ones obtained as above.

## Conclusions and Future Directions

We considered three QoS metrics, namely, trustworthiness, percentile response time and availability. Then, we studied all these three QoS metrics in a security-aware resource optimization problem that typically arises in Web services, and formulated the security-aware resource optimization problem as an optimization problem under SLA constraints in the case of single class. In our approach we considered the percentile response time that may better address a customer's concern as opposed to the average response time that is commonly used in the literature. By the use of an effective and accurate numerical solution for the calculation of the percentile response time (also refer to Xiong & Perros (2006b)), we solved the optimization problem using an efficient numerical procedure. Our numerical validations showed that our proposed algorithm has provided a good accuracy.

In the chapter, we provide a preliminary study of the security-aware resource optimization in distributed computing. The study raises several interesting questions. We briefly discuss some of them as follows.

We introduce a trust manager who is a trusted agent representing customers, and a service broker who is a trusted agent representing service sites. The failure of a single point on either the trust manager or the service broker could be catastrophic. Hence, a decentralized mechanism should be introduced for the replacement of the centralized design. But, how much performance overhead would be added when a decentralized mechanism is in place? Additionally, if a service provider could not provide QoS as defined in an SLA, what penalty should the service provider encounter and how to implement a penalty mechanism? Furthermore, the trust manager selects service sites by using their trust indices that are partially based on the feedback of customers. Then, the question is how to deal with customer's negative feedback.

## References

Azzedin, F. & Maheswaran, M. (2002). Evolving and managing trust in Grid computing systems. In *Proceedings of the IEEE Canadian Conference on Electrical Computing Engineering (CCECE '02)*.

Brown, A. & Patterson, D. (2000). Towards Availability Benchmarks: A Case Study of Software RAID Systems. In *Proceedings of 2000 USENIX Annual Technical Conference*.

Chandra, A, Gong, W.& Shenoy, P. (2003). Dynamic Resource Allocation for Shared Data Centers Using Online Measurements. In *Proceedings of Eleventh International Conference on Quality of Service (IWQoS 2003)*.

CISCO (2004). Increasing network availability. In <http://www.cisco.com/wrap/public/779/largeent/learn/technologies/ina/IncreasingNetworksAvailability-Abstract.pdf>.

Dobson, G. (2002). Quality of service in service-oriented architectures. In <http://digs.sourceforge.net/papers/qos.html>.

Jurca, R. & Faltings, B. (2005). Reputation-based Service Level Agreements for Web services. In *Proceedings of Third International Conference on Service Oriented Computing (ICSOC 2005)*, Amsterdam, The Netherlands.

Keller, A. & Ludwig, H. (2003). The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web services. *Journal of Network and Systems Management, Special Issue on `E-Business Management*, pp. 27-33.

Menasce, D. & Almeida, V. (2002). *Capacity planning for Web services: Metrics, Models, and Methods*. Prentice Hall PTR, Upper Saddle River, NJ.

Menasce, D. & Casalicchio, E. (2004). A Framework for Resource Allocation in Grid Computing. In *Proceedings of the 12th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*, pp. 259-267.

Levy, R., Nagarajarao, J., Pacifici, G., Spreitzer, M., Tantawi, A. & Youssef, A. (2003). Performance Management for Cluster Based Web services. In *Proceedings of the 8th IFIP/IEEE International Symposium on Integrated Network Management (IM2003)*.

Nabrzysbi, J., Schopf, J. & Weglarz, J. (2004). *Grid Resource Management*. Kluwer Academic Publication, Boston, MA.

Tosic, V., Patel, K. & Pagurek, B. (2002). WSOL - *Web Service Offerings Language*, Lecture Notes In Computer Science, Vol. 2512, Revised Papers from the International Workshop on Web services, E-Business, and the Semantic Web, pp. 57-67.

Vu, L, Hauswirth, M., & Aberer, K. (2005). QoS-based Service Selection and Ranking with Trust and Reputation Management. *Infoscience's Technical Report*.

Wickramage, N. & Weerawarana, S. (2005). A Benchmark for Web Service Frameworks. In *Proceedings of the 2005 IEEE international conference on service computing*, pp.233-242.

Xiong, K. & Perros, H. (2006a). Security-aware resource allocation in Web services. In *Proceedings of the International Conference on Web Services*, Chicago, IL.

Xiong, K. & Perros, H. (2006b). Resource optimization subject to a percentile response time SLA for enterprise computing. In *Proceedings of the IEEE Globecom: Quality Reliability and Performance Modeling for Emerging Network Services (Globecom-QRPM)*, San Francisco, CA.

Zhang, Q., Yu, T., & Irwin, K. (2004). A classification scheme for trust functions in reputation-based trust management. In *Proceedings of the International Workshop on Trust, Security, and Reputation on the Semantic Web*, Hiroshima, Japan.

Ziegler, C. & Lausen, G. (2004). Spreading Activation Models for Trust Propagation. In *Proceedings of the IEEE International Conference on e-Technology, e-commerce, and e-Service (EEE '04)*.