

# Trustworthy Web Services Provisioning for Differentiated Customer Services

Kaiqi Xiong

Department of Computer Science  
North Carolina State University  
Raleigh, NC 27965-7534, USA  
xiong@csc.ncsu.edu

Harry Perros

Department of Computer Science  
North Carolina State University  
Raleigh, NC 27965-7534, USA  
hp@csc.ncsu.edu

## Abstract

*With the number of e-Business applications dramatically increasing, a service level agreement (SLA) will play an important part in Web services. The SLA is a combination of several quality of services (QoS), such as security, performance, and availability, agreed between a customer and a service provider. Most existing research addresses only one of these QoS metrics. Furthermore, in the case of the response time defined as one of QoS metrics for performance, only the average time to process and complete a job is typically used. Moreover, customer requests often need to be distinguished, with different request characteristics and customer's different service requirements.*

*In this paper, we consider a set of computer resources used by a service broker to host enterprise applications for two classes of differentiated customer services subject to a service level agreement. We study three QoS metrics, namely, trustworthiness, a percentile response time, and availability. The percentile response time metric defines a value below which the end-to-end response time has to be for a given percent of time. We present an approach for resource optimization in such an environment that minimizes the total cost of computer resources while satisfying all these three QoS metrics in a trust-based resource provisioning problem which typically arises in Web services. We formulate the trust-based resource provisioning problem as an optimization problem under SLA constraints, and then solve it using an efficient numerical procedure.*

## 1 Introduction

Web services technology was introduced as a major component of .NET technology by Microsoft in June of 2000, and it is widely considered as an emerging paradigm for the next generation of Internet computing. Web services technology has become the most popular computing paradigm for e-business applications and distributed environments. Many companies such as Google and Amazon are boosting their traffic using Web services APIs (Menasce 2004 [17]). By adopting service-oriented architectures (SOAs), services components from several universal service providers can be flexibly integrated into a composite service regardless of their location, platform, and execution speed.

The main standard in Web services is Extensible Markup Language, XML. XML provides a foundation for many core standards including Web Services Description Language (WSDL), Universal Description, Discovery and Integration specification (UDDI), and Simple Object Access Protocol (SOAP). WSDL allows developers to describe what services are offered, and it helps Web services of e-business to be accessed in public. UDDI defines XML-based registries in which businesses can upload information about themselves and the services they offer. SOAP gives us a way to move XML messages between locations, and it enables programs on separate computers to interact across any network. Thus, Web services allows us to exchange data with other applications on different computers by using Internet protocols.

However, most existing Web services products do not support service level agreements (SLAs) that guarantee a level of service delivered to a customer for a given price. An SLA is a formal contract between a customer and a service provider that 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 (Menasce [16]).

1. *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.
2. *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 the paper, we consider a resource management problem for Web services under SLA guarantees. Specifically, we define and solve a trust-based resource provisioning problem that occurs in Web services applications, subject to the constraints of trustworthiness, a percentile response time and availability. Figure 1 depicts a framework for this trust-based resource provisioning problem. 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  $1, 2, \dots, m$  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 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 Section 3.

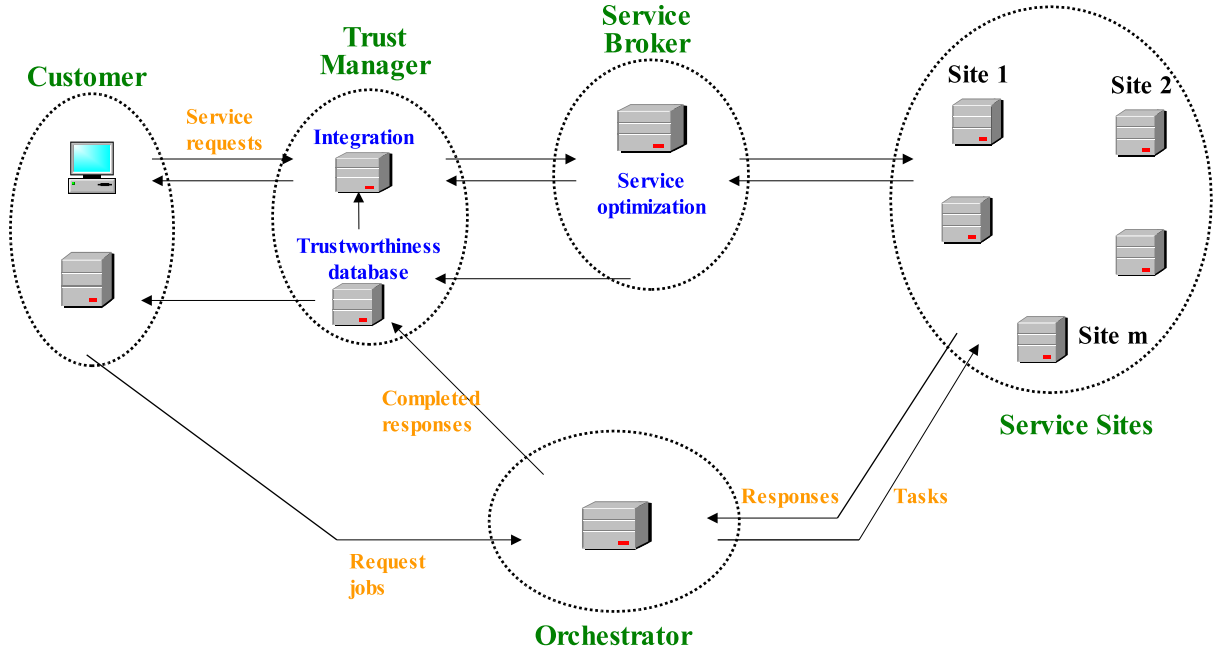


Figure 1. An SLA-based Web Services Model

The trust-based resource provisioning problem is to minimize the overall cost of the trusted computing resources required while satisfying SLA requirements. For presentation purpose, we assume that each resource site  $j$  consists of multiple servers of the same type, each with cost  $c_j$ . Otherwise, if a resource site consists of more than one type of multiple servers, then it is decomposed into several sites so that each one only contains one set of multiple servers of the same type and cost. Let  $N_j$  be the number of servers at site  $j$  ( $j = 1, 2, \dots, m$ ). Thus, the trust-based resource provisioning is quantified by solving for  $n_j$  ( $1 \leq n_j \leq N_j$ ) in the following optimization problem:

$$\min_{n_1, \dots, n_m} (n_1 c_1 + \dots + n_m c_m) \quad (1)$$

subject to constraints by an SLA. We discuss these constraints in Section 4.

Customer jobs often need to be distinguished in terms of service times and QoS requirements. In this paper, we consider two classes of customer jobs, namely, class 1 and class 2, with class 1 having a higher priority over class 2. We assume a *preemptive-resume* priority discipline at each service site. That is, a class 2 service job can be interrupted if a higher-priority service job of class 1 arrives during its service. The interrupted service job resumes its service from where it stopped after the class 1 service job and any class 1 service jobs that may arrive during its service, complete their service. As will be seen, the two-class preemptive-resume discipline can be generalized to any number of classes.

The rest of the paper is organized as follows. Related work is reviewed in Section 2. In Section 3, we describe in detail the framework for the trust-based resource provisioning problem, and in Section

4, we describe the SLA metrics considered in this paper. In Section 5, we propose an approach for the trust-based resource provisioning problem. A numerical example is given in Section 6 that demonstrates the validity of this approach. Finally, the conclusions are given in Section 7.

## 2 Related Work

Web services is often contracted through SLAs which 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, WSOL, and QML. The issue of a reputation-based SLA has been studied by Jurca and Faltings [12] in which the cost is determined by the QoS that was actually delivered.

The issue of service quality has been extensively investigated. Zhang et al. [26], and Ziegler and Lausen [27] classified various trust metrics, including a rank-based metric. Vu et al. [22] proposed a new QoS-based semantic Web services selection and ranking solution using a trust and reputation management and assuming known QoS qualities. Wickramage and Weerawarana [24] introduced a performance model to analyze the round trip time of SOAP messages using measurements and a curve fitting technique.

Availability is a critical metric in today's computer design (Hennessy [11]). 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 study in this paper. Cisco has asserted that the operational failure causes 80% of non-availability [8]. Hence, increasing network availability is becoming a key priority for enterprise and service provider organizations, as discussed in [7]. Martin and Nilsson [15] gives an example of how network service availability is defined in Sprint's SLA and WorldCom's SLA.

Brown and Patterson [3] 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. Menasce and Almeida [18] designed self-managing systems to control QoS. Levy et al. [14] presented a performance management system for cluster-based Web services. In both studies the average response time is used as a metric. Chandra [4] employed an online measurement method, and considered a resource provisioning problem based on measured response times.

In order to compute a percentile of the response time one has to first find the probability distribution function (pdf) of the response time. This is not an easy task in a complex computing environment involving many computing nodes. The calculation of the pdf of the response time is relatively simple for overtake-free paths in Jackson and Gordon-Newell queueing networks (Walrand and Varaiya [23] and Daduna [9]). Walrand and Varaiya [23] showed that in any open Jackson network, the response times of a customer at the various nodes of overtake-free path are all mutually independent. Daduna [9] further proved that the same result is valid for overtake-free paths in Gordon-Newell queueing networks. In this paper, we develop efficient and accurate solutions of the percentile of the response time for a service model under our study.

"Trust" is used to deal with the notion of the trustworthiness in this research. In this paper, 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. [26], and Ziegler and Lausen [27] classified various trust metrics. In this research, we use a rank and a threshold-based approach. The rank-based approach is to rank the trustworthiness of all sites who provide the same type of services, such as EigenRep in Kamvar et al. [13]. A threshold-based approach is to choose any of those service sites who can meet trust requirements predefined by customers. For example, if the trustworthiness of a service site is over 0.9 and the predefined trust requirement of a customer is 0.9, then the trust manager can select the site to serve the customer's service request. The relationship between the rank and the threshold based approaches are discussed in Zhang et al. [26].

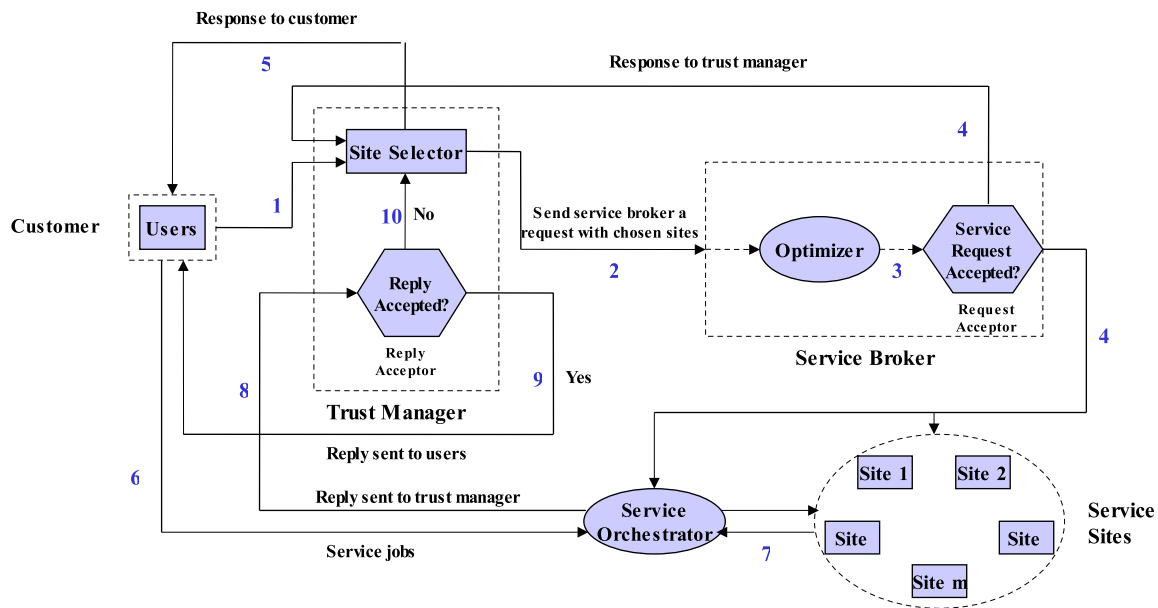
Resource optimization problems subject to performance metrics, such as response time, bandwidth, and link utilization have been extensively studied. It was studied for multiple packet networks in Bouillet, et al. [2], different classes of flows at network nodes in [6], IP networks in Martin and Nilsson [15], and grid computing in Menasce and Casalicchio in [19], Bouillet, et al. [2] considered a routing and resource management problem subject to requirements on aggregate bandwidth from ingress to egress nodes. In [6], Chassot et al. dealt with a communication architecture with guaranteed end-to-end QoS in an IPv6 environment. The end-to-end QoS includes an end-to-end delay (i.e., response time). Chassot et al. only discussed and measured the maximal, minimal and average values of the response time. Moreover, Martin and Nilsson [15] and Menasce and Casalicchio [19] considered only the average response time. In [21], Osogami and Wierman analyzed an *M/GI/k* system with two priority classes and a general phase-type distribution and evaluated the optimal number of servers based on the overall *mean* response time. Resource provisioning problems in distributed systems have also been widely studied based on trustworthiness or response time (e.g., see [20]).

In the paper, we consider a resource provisioning problem in Web services subject to all three QoS metrics, expressed by trustworthiness, percentile response time and service availability. To the best of our knowledge, our study is the first attempt to consider the resource provisioning under all these three constraints.

### 3 A Framework for the Trust-based 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 an SLA is very important and challenging due to the dynamic and unpredictable nature of Web services applications. In the paper, we propose a Web services framework shown in Figure 2 for solving the trust-based resource provisioning problem. The framework consists of a customer, a trust manager, a service broker, a service orchestrator, and service sites. The roles of these stakeholders are explained below.

- 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 is associated with service jobs generated by the users at a given rate with specific QoS requirements.
- The trust manager is an entity that plays the following roles:



**Figure 2. A Framework for the Trust-based Resource Provisioning Problem**

- 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.
- The service broker is an entity that represents service sites. After it receives the trust manager's selection of resource sites, and it 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 paper, we consider it as a hardware device, such as a blade, a CPU, and a storage device.
- The service orchestrator initiates the execution of a sequence of tasks associated with the workflow of a service request.

In this paper, we will assume that the tasks executed sequentially, i.e., we will not consider join and fork patterns.

Below, we describe in detail the steps involved in negotiating a contract.

**Step 1.** A customer submits a service request to the trust manager, who represents the customer. We recall that the customer represents a business associated with a number of users who generate service jobs to be executed at selected resource sites with a given QoS.

**Step 2.** The site selector, who is part of the trust manager, selects service sites with trust indices that

meet the customer's trust requirement. Then, the trust manager submits the customer's service 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 that the customer's SLAs are guaranteed. The optimization algorithm uses a cost structure provided by the service provider(s) who own the selected resource sites.

**Steps 4 and 5.** The service request is accepted if the optimization solution is feasible and the cost is acceptable. In this case, the service broker notifies the trust manager, who in turn notifies the customer, the service orchestrator and the service providers who own the service sites. Renegotiation is possible if the service request is rejected.

**Steps 6 and 7.** Once the service request has been negotiated, the users begin to submit service jobs to the orchestrator, who makes sure that all the tasks in the workflow are executed by the service sites.

**Steps 8 and 9.** The result obtained after a service job is executed is sent back to the trust manager. The reply acceptor, who is part of the trust manager, decides whether to accept the reply based on the trust indices of the resource sites at the time when it receives the response from the orchestrator. If the trust indices of these chosen sites meet the customer's requirement, then the service reply is accepted and forwarded to the customer.

**Step 10.** If the trust indices of these chosen sites do not meet the customer's requirement, then the service reply is not accepted, and the customer is notified to that effect. A notification is also sent to the site selector. When the number of notifications exceeds a predefined threshold, the service broker appropriately selects new service sites and then the flow proceeds as in Step 3. If the number of times the service broker selects new service sites surpasses a threshold, then the trust manager notifies the customer that its service request cannot be completed.

## 4 The SLA Metrics

The SLA metrics considered in this paper are trustworthiness, percentile response time, and service availability. Below, we discuss these three SLA metrics in detail.

### 4.1 The Trustworthiness of Resource Sites

Security can be classified into *identity security* and *behavior security*, similar to identity trust and behavior trust defined in a grid computing system (Azzedin et al. [1]). Identity security includes the authentication and authorization between a customer and a service provider, data confidentiality and data integrity. It has been widely studied in the literature. The identity security is beyond the scope of our study in this paper.

In this paper, "trust" is used to deal with the notion of the trustworthiness in behavior security. Its definition is varied in the literature. We define trust as a firm belief in the competence of a resource site that acts as expected. The trustworthiness of resource sites is an indicator of the quality of services provided by these sites based on previous and current job completion experience. It often predicates

the future behavior of the quality of services at these sites. Moreover, we are only interested in the trustworthiness of resource sites from a customer's perspective. Hence, in this paper we simply assume that these resource sites trust each other.

Furthermore, we assume that the trust manager is a trusted agent who represents customers. The trust manager uses the collected trustworthy information regarding the resource sites to evaluate their security behavior. We consider security behavior by modeling the behavior trusts of all sites, and quantify the trustworthiness of these sites using a rank and a threshold-based approach. This approach is based on previous job completion experience assessed by the trust manager and customers. The assessment may also include the opinion of other trust managers besides its own customer's feedback. The domain of feedback is assumed to be  $[0, 1]$ .

The feedback is a statement issued by the trust manager's customers about the quality of service provided by the service sites chosen for each service request or for a set of service requests during a certain period of time. These customers only provide feedback about the quality of the received service and not about each service site involved in this service, since a customer is usually not aware of which service sites process its service request. In addition, the trust manager may have its own feedback. The trust indices of those chosen service sites are updated by aggregating both the trust manager's feedback and its customers' feedbacks.

The opinion is defined as the information of trustworthiness provided by those trust managers who are *neighbors* of the trust manager. These neighbors are a small subset of the trust manager's acquaintances, adaptively selected based on their usefulness. The opinion aggregates the overall impression of those neighborhood trust managers for the service sites.

We first discuss the case in which the trust indices of service sites are only determined by the trust manager's and its customers' feedbacks. Let us consider the discrete times  $t_1, t_2, \dots, t_k, \dots$  in an increasing order ( $k = 1, 2, \dots$ ). Let  $I_j^{t_k}$  be the trust index of site  $j$  at time  $t_k$ . Then, a *trust function* is defined by

$$I_j^{t_{k+1}} = \xi \frac{R_j^s(k+1)}{R_j^a(k+1)} + (1 - \xi)I_j^{t_k} \quad (2)$$

for each site  $j$  ( $j = 1, 2, \dots, M$ ), where  $R_j^s(k+1)$  is the number of service jobs that are completed at site  $j$  and satisfied by customers, provided that the trust manager itself does not assess each completed job, or both the trust manager and its customers provided that both of them assess each completed job.  $R_j^a(k+1)$  is the total number of service jobs submitted to site  $j$  during the time period  $[t_k, t_{k+1}]$ . The trust manager's satisfaction is assessed by the validation of a customer's SLA requirement after a service is completed, and a customer's satisfaction is based on the feedback assessed by the customer through a comparison of the job completion experience with its expectation.

For presentation purpose, we simply assume that the trust manager has the same feedback as its customers. (Otherwise, the following discussion and notation needs to be adjusted accordingly that can be easily done.) Denote  $r_j(k+1)$  by

$$r_j(k+1) = \frac{R_j^s(k+1)}{R_j^a(k+1)} \quad (3)$$

Moreover,  $r_j(k+1)$  is the satisfactory rate at site  $j$  from time  $t_k$  to  $t_{k+1}$  and it obviously ranges from 0 to 1. Clearly, when a set of the chosen sites is unchanged during this period of time, the number of completion jobs is the same for all chosen sites. Thereby,  $r_j(k+1)$  is the same for these chosen sites as

well.  $\xi$  is a parameter determined by the trust manager. It is chosen depending on the type of resource sites. If a critical service job is processed at site  $j$  and site  $j$ 's security is sensitive with the change of time, then  $\xi$  should be close to 1 (e.g., bigger than 0.8). Otherwise, it should be close to 0 (e.g., smaller than 0.2). Thus,  $I_j^{t_k}$  ranges from 0 to 1, and it is called *a trust index*. It gives the percent of a time that a resource site completes jobs to the satisfaction of the trust manager and its customers.

Now, we discuss the case in which the trust indices of service sites are determined by not only the trust manager's and its customers' feedbacks, but also the opinions of other trust managers. In today's large scale complex computer system, a central trust manager would not be able to represent all customers, and would not know or would not be able to keep up with the trustworthiness information of all service sites in the system. This implies that there would exist multiple or many trust managers in the large scale complex computer system. In this case, the trust index function in (2) should be modified to consider the opinions of the trust manager's neighbors besides its own customer's feedback. Thus, the satisfactory rate  $r_j(k+1)$  consists of the following two components: the first component is based on the feedbacks from the trust manager and its own customers that is denoted by  $r_j^b(k+1)$ . Thus,  $r_j^b(k+1)$  is given by (3). The second component is determined by aggregating all the opinions of the trust manager's neighbors and it is denoted by  $r_j^o(k+1)$ . The satisfactory rate  $r_j(k+1)$  is therefore given by

$$r_j(k+1) = \xi^b \times r_j^b(k+1) + \xi^o \times r_j^o(k+1)$$

where  $\xi^b + \xi^o = 1$ . The two parameters are used to adjust the balance between the feedbacks and the opinion, and they are determined by the trust manager. Consequently, the trust index function in (2) is rewritten as

$$\begin{aligned} I_j^{t_{k+1}} &= \xi r_j(k+1) + (1-\xi)I_j^{t_k} \\ &= \xi [\xi^b \times r_j^b(k+1) \\ &\quad + \xi^o \times r_j^o(k+1)] + (1-\xi)I_j^{t_k} \end{aligned}$$

The trust function describes the trustworthiness of resource sites monitored and updated by the trust manager. Therefore, the trust function reflects a probabilistic security behavior of the resource sites from a customer's perspective.

## 4.2 The Percentile Response Time

The SLA metric for performance, defined in Section 1, includes *throughput* and *response time*. As an end user, a customer is in general concerned about response time rather than throughput. So, in the study we only consider *the response time*. In the trust-based resource provisioning problem for Web services, a response time is the time it takes for a job to be executed and completed in a distributed computing environment consisting of a trust manager, an orchestrator and multiple resource sites.

In the literature, typically the average response time (or the average execution time) is used (e.g., see Menasce [16] and [17]). The average response time is heavily influenced by "outliers," which occur in almost all measurements. Therefore, although the average response time is relatively easy to calculate, it may not address the concerns of a customer. Typically, a customer is more inclined to request a statistical bound on its response time than an average response time. For instance, a customer can request that at least 95% of the time the response time should be less than a desired value. Hence, in this paper we are

concerned with finding an optimal resource provisioning from trusted resource sites that meet a desired percentile response time.

Assume that  $f_T(t)$  is the probability distribution function of a response time  $T$  of a certain priority class. For example, in the case of two priority classes, for the high-priority class  $T = T^{(1)}$  and for the low-priority class  $T = T^{(2)}$ .  $T_D^{(r)}$  is a desired target response time for priority class  $r$  ( $r = 1, 2$ ) that a customer requests and agrees upon with its service provider based on a fee paid by the customer. The SLA performance metric used in this paper can be expressed as follow:

$$\int_0^{T_D^{(r)}} f_{T^{(r)}}(t) dt \geq \gamma^{(r)}\%, \quad (r = 1, 2) \quad (4)$$

That is,  $\gamma^{(r)}\%$  of the time a customer will receive its service in less than  $T_D^{(r)}$  ( $r = 1, 2$ ).

As an example let us consider an  $M/M/1$  queue with an arrival rate  $\lambda^{(r)}$  and a service rate  $\mu^{(r)}$  ( $r = 1, 2$ ). The service discipline is preemptive-resume. We want to describe the SLA performance metric (4) for the high-priority class. As discussed in Section 1, in this case, the steady-state probability of the system as far the high-priority class is concerned is  $p_0 = 1 - \rho^{(1)}$ , and  $p_k = (1 - \rho^{(1)})(\rho^{(1)})^k$ ,  $k > 0$ , where  $\rho^{(1)} = \frac{\lambda^{(1)}}{\mu^{(1)}}$ . The response time  $T^{(1)}$  is exponentially distributed with the parameter  $\mu^{(1)}(1 - \rho^{(1)})$ , i.e., the probability distribution of the high-priority response time is given by

$$f_{T^{(1)}}(t) = \mu^{(1)}(1 - \rho^{(1)})e^{-\mu^{(1)}(1 - \rho^{(1)})t} \quad (5)$$

Using the definition given in (4), we have that

$$\int_0^{T_D^{(1)}} f_{T^{(1)}}(t) dt = 1 - e^{-\mu^{(1)}(1 - \rho^{(1)})T_D} \geq \gamma^{(1)}\% \quad (6)$$

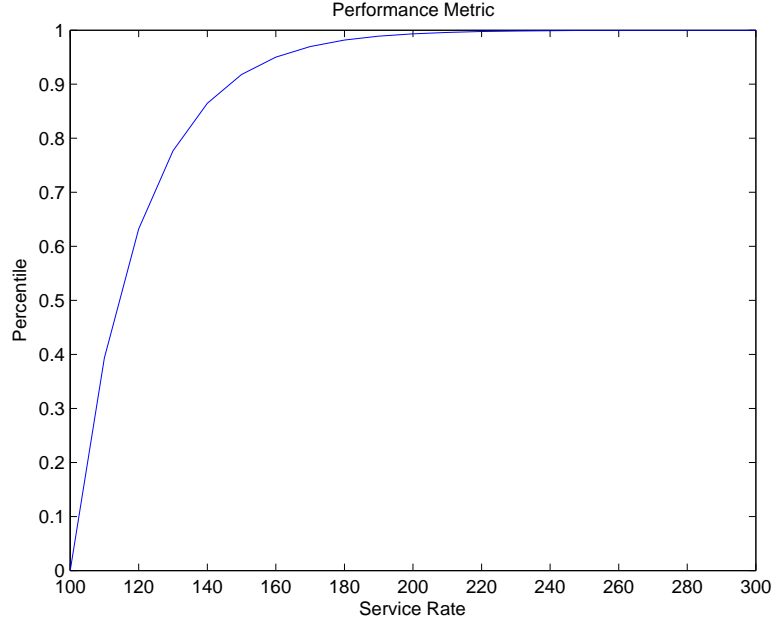
or  $\mu^{(1)} \geq \frac{-\ln(1 - \gamma^{(1)}\%)}{T_D^{(1)}} + \lambda^{(1)}$ . This means that in order to guarantee a higher SLA service level,  $\mu^{(1)}$  increases when  $T_D^{(1)}$  decreases. Similarly, for any given arrival rate  $\lambda^{(1)}$  and service rate  $\mu^{(1)}$ , we can use (6) to find the percentile of  $\gamma^{(1)}$ .

For example, Figure 3 gives the cumulative distribution of the response time when  $\lambda^{(1)} = 100$  and  $T_D^{(1)} = 0.05$  as a function of  $\gamma^{(1)}$ . As shown in Figure 3, the percentile of response time will increase as the service rate increases. From the figure we see that the service rate has to be bigger than 150 in order that 90% of the response time is less than 0.05.

### 4.3 The Service Availability

The metric for service availability is the percentage of time that a resource is “up” or “down”. Let  $MMTF_j$  (Mean Time to Failure) be the average time of a server failure, and  $MTTR_j$  (Mean Time to Recover) be the average time for recovering a server at the resource site  $j$  ( $j = 1, 2, \dots, m$ ).  $MMTF_j$  information can be obtained from a hardware provider at site  $j$ , see for instance Cisco [8].  $MTTR_j$  is determined by evaluating how quickly the owner of site  $j$  can repair a broken server.

Each server fails at resource site  $j$  at a rate of  $\frac{1}{MMTF_j}$ , denoted by  $a_j$ , and recovers (i.e., it is put back into operation) at a rate of  $\frac{1}{MTTR_j}$ , denoted by  $b_j$  ( $j = 1, \dots, m$ ). Thus, a two-state Markov chain with the states “up” and “down” can be used to study the service availability at site  $j$ . The failure rate  $a_j$  is



**Figure 3. The Cumulative Distribution vs. Service Rate**

the state of transition from “up” to “down,” and the recovery rate  $b_j$  represents the rate of transition from “down” to “up.” Then, the probability  $p_j^i$  that  $i$  servers of a total  $N_j$  servers are down is given by

$$p_j^i = \frac{N_j!}{i!(N_j - i)!} \eta_j^i p_j^0, \quad \text{for } i = 1, \dots, N_j \quad (7)$$

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

$$p_j^0 = [N_j! \sum_{i=0}^{N_j} \frac{\eta_j^i}{i!(N_j - i)!}]^{-1} \quad (8)$$

The probability that no more than  $N_j - n_j$  servers at site  $j$  are down is:

$$P_j(n_j, N_j) = p_j^0 \sum_{i=0}^{N_j - n_j} \frac{N_j!}{i!(N_j - i)!} \eta_j^i \quad (9)$$

Namely, the expression given in (9) can also be seen as the probability that at least  $n_j$  servers in site  $j$  of a total  $N_j$  servers are available.

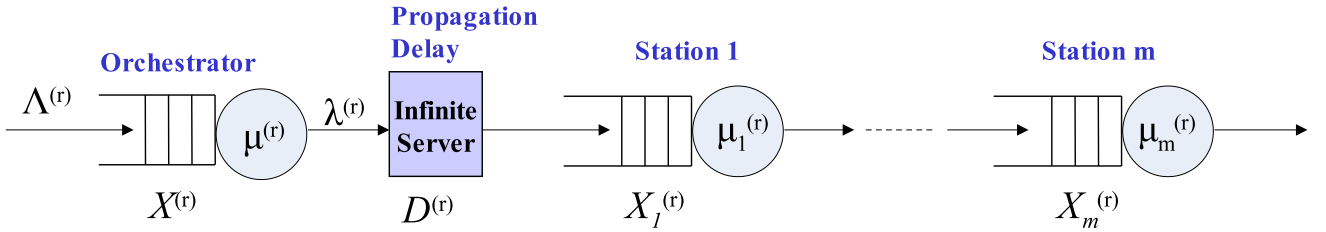
## 5 Trust-based Resource Optimization

As we see in Section 3, a Web services framework consists of a customer, a trust manager, a service broker, and a service orchestrator. In this framework, an optimizer within the service broker is used to calculate the number of servers in each resource site required to ensure that the response time of a

service job meets a predefined percentile response time. The response time consists of the processing time at the service orchestrator, the processing time at each service site, and the propagation delay.

The execution of a request job (hereafter referred to as a job) involves the sequential execution of a number of tasks. (As mentioned previously, we do not consider a fork and join patterns in this paper.) Each task is initiated by the orchestrator on the appropriate resource site. The orchestrator waits until it receives the result from the resource site, and it then initiates the next task in the workflow on another resource site. This continues until all the tasks in the workflow have been completed. At that time, the final result is returned to the trust manager for verification.

We assume that each task is executed on a different resource site, and that there are  $m$  tasks, i.e.,  $m$  resource sites. Each resource site can be seen as carrying out a particular function, such as, a database server, a computing server, a file server, and a Web server. Then, the time it takes for a job to be executed is the time it takes for a job to traverse the tandem queueing network shown in Figure 4. This first node represents the orchestrator, the infinite server represents the propagation delay, and stations 1 to  $m$  represents each of the  $m$  resource sites. Jobs arrive at the orchestrator node, and then they proceed



**Figure 4. The Tandem Queueing Network**

through the tandem queueing network until they depart from node  $m$ . The reply from each resource site back to orchestrator is not modelled explicitly. Rather, the infinite server represents all the propagation delays involved in the execution of the workflow associated with a job. Also, the service time at the orchestrator is assumed to be equal to the sum of all the services included for initiating the  $m$  tasks and other overheads associated with a job.

Each station  $j$ ,  $j = 1, 2, \dots, m$ , is represented by a single queue of infinite capacity. Two classes of customers are considered, classes 1 and 2, with a preemptive-resume priority. Each station  $j$  is served by  $n_j$  identical servers at the rate  $\mu_j^{(r)}$ ,  $r = 1, 2$ . Also, the class-dependent service rate at the orchestrator and infinite server is  $\mu^{(r)}$  and  $D^{(r)}$  respectively. Let  $\Lambda^{(r)}$ ,  $\lambda^{(r)}$  and  $\lambda_j^{(r)}$ ,  $j = 1, 2, \dots, m$ , be respectively the external arrival rate to the orchestrator, the arrival rates to the infinite server, and the arrival rates to the  $j$  station. We have that  $\Lambda^{(r)} = \lambda^{(r)} = \lambda_j^{(r)}$ ,  $j = 1, 2, \dots, m$ . All service times are assumed exponentially distributed, and the external arrival process to the orchestrator occurs is Poisson distributed.

Now, denote the overall service cost (1) by

$$g(n_1^{(r)}, n_2^{(r)}, \dots, n_m^{(r)}) = \sum_{j=1}^m n_j^{(r)} c_j \quad (10)$$

where  $n_1^{(r)}, n_2^{(r)}, \dots, n_m^{(r)}$  are the number of servers allocated to a customer who submits jobs at the rate  $\Lambda^{(r)}$  ( $r = 1, 2$ ). We are interested in minimizing this cost function subject to a percentile response time and availability constraints.

Specifically, the trust-based resource provisioning problem can be formulated as follows.

1. Select  $m$  resource sites within the predefined trust indices of all class customers at time  $t = t_k$ .
2. Solve for  $n_j^{(r)}$  in the  $m$ -dimensional integer optimization problem:

$$\min_{n_1^{(r)}, \dots, n_m^{(r)}} g(n_1^{(r)}, n_2^{(r)}, \dots, n_m^{(r)}) \quad (11)$$

Under the constraints:

- (a) The percentile response time

$$\int_0^{T_D^{(r)}} f_{T^{(r)}}(t) dt \geq \gamma^{(r)}\%, \quad r = 1, 2 \quad (12)$$

where  $T^{(r)}$  be a random variable that represents the response time of a class  $r$  job,  $f_{T^{(r)}}(t)$  is the probability distribution (pdf) of this response time, and  $\gamma^{(r)}$  is the percent of time that  $T^{(r)}$  should be less than a target upper bound denoted by  $T_D^{(r)}$ .

- (b) The service availability constraint:

$$P_j(n_j^{(r)}, N_j) \geq \zeta_j^{(r)}\% \quad (13)$$

where  $P_j(n_j^{(r)}, N_j)$  is given by (9).  $\zeta_j^{(r)}\%$  is a desired percentage of service availability at site  $j$ .

3. Calculate

$$n_j = \max\{n_j^{(1)}, n_j^{(2)}\} \quad (j = 1, 2, \dots, m)$$

4. As explained in Section 3, the orchestrator forwards the reply of a completed job to the trust manager. The reply is accepted by the trust manager (and forwarded to the appropriate user) if the trust index of the resource sites meet a predefined index value. Otherwise, the reply is rejected.

At the time  $t = t_k + T^D$ , the trust indices of all the  $M$  resource sites are updated. At this moment, the trust manager may replace one or more resource sites associated with the execution of job submitted by a customer if their must index has fallen below the predefined threshold.

Sub-problems 1 and 4 are solved at the customer side (i.e., the trust manager) to ensure a customer service received from reliable resource sites. Sub-problems 2 and 3 are solved by the service broker who represents the resource sites. Trustworthiness of the resource sites in Sub-problems 1 and 4 are not considered as a constraint for the optimization problem in Sub-problem 2.

### 5.0.1 The pdf of the Response Time

As mentioned above, station  $j$ ,  $j = 1, 2, \dots, m$ , consists of an infinite capacity queue served by  $n_j$  servers, where  $n_j = 1, 2, \dots, N_j$ , is to be determined using the optimization described in Step 2. In order to simplify the calculation of the pdf of the response time of a job, we will assume that station  $j$ ,  $j = 1, 2, \dots, m$ , is served by a single server with a class-dependent service rate  $\psi^{(r)}(n_j)\mu_j^{(r)}$ , where  $\psi^{(r)}(n_j)$  is a known function of  $n_j$  ( $r = 1, 2$ ), and depends on the configuration of servers and the type of customers at each station. It is non-decreasing and can be inverted, i.e.,  $(\psi^{(r)})^{-1}$  exists ( $r = 1, 2$ ). For instance, suppose that a station represents a group of CPUs. Then,  $\psi^{(r)}(n)$  can be seen as a CPU scaling factor for the number of CPUs from 1 to  $n$ . According to [5],  $\psi^{(r)}(n) = (\epsilon^{(r)})^{\log_2 n}$ , where  $\epsilon^{(r)}$  is a basic scaling factor from 1 CPU to 2, and it ranges from 1 to 2 ( $r = 1, 2$ ). So,  $(\psi^{(r)})^{-1}(n) = (\epsilon^{(r)})^{-\log_2 n}$ . As another example, let us consider a domain that represents a group of routers in a network model. Then,  $\psi^{(r)}(n) = 1$  when these  $n$  routers are serially placed, and  $\psi^{(r)}(n) = n$  when they are in parallel. Since only a high-priority or a low priority customer is served at time, we have that  $\psi^{(1)}(n_j)\mu_j^{(1)}$  is considered as the same as  $\psi^{(2)}(n_j)\mu_j^{(2)}$ .

Since the queueing network is overtake-free, the waiting time of a customer at a station is independent of its waiting times at other stations (see [9] and [23]). Let  $X^{(r)}$  be the time elapsed from the moment a customer of class  $r$  arriving at the trust manager server to the moment it departs from it,  $D^{(r)}$  be the service time at the infinite server, and  $X_j^{(r)}$  be the time elapsed from the moment a customer of class  $r$  arriving at station  $j$  to the moment it departs from the station. Then, the total response time is

$$T^{(r)} = X^{(r)} + D^{(r)} + X_1^{(r)} + X_2^{(r)} + \dots + X_m^{(r)},$$

for the customer of class  $r$  and hence the LST (Laplace-Stieltjes transform) of the response time  $T^{(r)}$  is

$$L_{T^{(r)}}(s) = L_{X^{(r)}}(s)L_{D^{(r)}}(s)L_{X_1^{(r)}}(s)\cdots L_{X_m^{(r)}}(s) \quad (14)$$

where  $L_{D^{(r)}}(s)$  is the LST of the service time  $D^{(r)}$  given by

$$L_{D^{(r)}}(s) = \frac{\lambda^{(r)}}{s + \lambda^{(r)}} \quad (15)$$

Also, let  $L_{X^{(r)}}(s)$  be the LST of the waiting times  $X^{(r)}$  at the orchestrator, and  $L_{X_j^{(r)}}(s)$  the LST of the waiting time  $X_j^{(r)}$  at the  $j$ -th station, where  $r = 1, 2, j = 1, 2, \dots, m$ .

Due to the preemptive-resume priority, the waiting time for the high-priority jobs is the same as in the single class FIFO  $M/M/1$ . Thus,  $L_{X^{(1)}}(s)$  is given by

$$L_{X^{(1)}}(s) = \frac{\mu(1 - \rho^{(1)})}{s + \mu(1 - \rho^{(1)})}, \quad j = 1, 2, \dots, m \quad (16)$$

The LST  $L_{X^{(2)}}(s)$  of the waiting time of the low-priority jobs is given by

$$L_{X^{(2)}}(s) = \frac{(1 - \rho)\delta^{(1)}}{1 - \rho\delta^{(1)}}, \quad j = 1, 2, \dots, m \quad (17)$$

due to (25) in [25], where  $\mu \triangleq \mu^{(1)} = \mu^{(2)}$ ,  $\rho^{(r)} = \frac{\lambda^{(r)}}{\mu}$ ,  $r = 1, 2$ ,  $\rho = \frac{\lambda^{(1)} + \lambda^{(2)}}{\mu}$ , and  $\delta^{(1)}$  is given by

$$\delta^{(1)} = \frac{\eta - \sqrt{\eta^2 - 4\Lambda^{(1)}\mu^{(1)}}}{2\Lambda^{(1)}}, \quad \text{and} \quad \eta = s + \Lambda^{(1)} + \mu^{(1)}$$

Furthermore,  $L_{X_j^{(1)}}(s)$  is given by

$$L_{X_j^{(1)}}(s) = \frac{\psi^{(1)}(n_j)\mu_j(1 - \rho_j^{(1)})}{s + \psi^{(1)}(n_j)\mu_j(1 - \rho_j^{(1)})}, \quad (j = 1, 2, \dots, m) \quad (18)$$

and  $L_{X_j^{(2)}}(s)$  is the LST of the low-priority response time given by

$$L_{X_j^{(2)}}(s) = \frac{(1 - \rho_j)\delta_j^{(1)}}{1 - \rho_j\delta_j^{(1)}}, \quad j = 1, 2, \dots, m \quad (19)$$

due to (25) in [25], where  $\rho_j^{(r)} = \frac{\lambda_j^{(r)}}{\psi^{(r)}(n_j)\mu_j}$ ,  $\rho_j = \frac{\lambda_j^{(1)} + \lambda_j^{(2)}}{\psi^{(r)}(n_j)\mu_j}$ , and  $\delta_j^{(1)}$  is given by

$$\delta_j^{(1)} = \frac{\eta_j - \sqrt{\eta_j^2 - 4\psi^{(1)}(n_j)\lambda_j^{(1)}\mu_j^{(1)}}}{2\psi^{(1)}(n_j)\lambda_j^{(1)}}, \quad \text{and} \quad \eta_j = s + \lambda_j^{(1)} + \psi^{(1)}(n_j)\mu_j^{(1)}$$

for  $j = 1, 2, \dots, m$ .

From (14)-(19) we have that

$$L_{T^{(1)}}(s) = \frac{\lambda^{(1)}}{s + \lambda^{(1)}} \frac{\mu(1 - \rho^{(1)})}{s + \mu(1 - \rho^{(1)})} \prod_{j=1}^m \frac{\psi^{(1)}(n_j)\mu_j(1 - \rho_j^{(1)})}{s + \psi^{(1)}(n_j)\mu_j(1 - \rho_j^{(1)})}$$

and

$$L_{T^{(2)}}(s) = \frac{\lambda^{(2)}}{s + \lambda^{(2)}} \frac{(1 - \rho)\delta^{(1)}}{1 - \rho\delta^{(1)}} \prod_{j=1}^m \frac{(1 - \rho_j)\delta_j^{(1)}}{1 - \rho_j\delta_j^{(1)}}$$

We observe that  $f_{T^{(r)}}(t)$  and  $F_{T^{(r)}}(t)$ ,  $r = 1, 2$ , are usually nonlinear functions of  $t$  and  $n_j$ . Hence, the resource optimization problem is an  $m$ -dimensional linear optimization problem subject to nonlinear constraints. In general, it is not easy to solve this problem. However, the complexity of the problem can be significantly reduced by requiring that the service rates of the queues in the queueing network in Figure 4 are all equal. That is, we find the optimum value of  $n_1, \dots, n_m$  such that  $\psi^{(r)}(n_1)\mu_1^{(r)} = \dots = \psi^{(r)}(n_m)\mu_m^{(r)}$ ,  $r = 1, 2$ , (*balanced utilization*).

From the traffic equations:  $\lambda^{(r)} = \lambda_j^{(r)} = \Lambda^{(r)}$  ( $j = 1, 2, \dots, m$ ), we have that the utilization of each station  $\rho_j^{(r)} = \frac{\lambda_j^{(r)}}{\psi^{(r)}(n_j)\mu_j^{(1)}} = \frac{\Lambda^{(r)}}{\psi^{(r)}(n_j)\mu_j^{(1)}}$ . Thus we have that for the high-priority queue,  $\hat{a}_i = \psi^{(1)}(n_i)\mu_i^{(1)}(1 - \rho_i^{(1)}) = \psi^{(1)}(n_j)\mu_j(1 - \rho_j^{(1)}) = \hat{a}_j \triangleq \hat{a}$  that implies  $n_j = (\psi^{(1)})^{-1}(\frac{\hat{a} + \lambda_j^{(1)}}{\mu_j})$ ,  $i, j = 1, 2, \dots, m$ . Hence, from (14) we have

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

and subsequently obtain that

$$F_{T^{(1)}}(t) = L^{-1} \left\{ \frac{\lambda^{(1)}}{s(s + \lambda^{(1)})} \cdot \frac{\mu(1 - \rho^{(1)})}{s + \mu(1 - \rho^{(1)})} \cdot \frac{\hat{a}^m}{(s + \hat{a})^m} \right\} \quad (20)$$

Moreover, since  $\hat{a}_i = \hat{a}_j = \hat{a}$  and  $\psi^{(1)}(n_i)\mu_i^{(1)} = \psi^{(2)}(n_i)\mu_i^{(2)}$ ,  $i, j = 1, 2, \dots, m$ , we have that  $\psi^{(2)}(n_1)\mu_1^{(2)} = \dots = \psi^{(2)}(n_m)\mu_m^{(2)} \triangleq \hat{b}$ . It is easy to see that  $\hat{b} = \hat{a} + \lambda_j^{(1)}$ . Thus, for the low-priority queue, we obtain that  $\rho_{j_1} = \rho_{j_2} \triangleq \hat{\rho}$ , and  $\delta_{j_1}^{(1)} = \delta_{j_2}^{(1)} \triangleq \hat{\delta}^{(1)}$ , for  $j_1, j_2 = 1, 2, \dots, m$ . Furthermore,  $L_{T^{(2)}}(s)$  reduces to

$$L_{T^{(2)}}(s) = \frac{\lambda^{(2)}}{s + \lambda^{(2)}} \cdot \frac{(1 - \rho)\delta^{(1)}}{1 - \rho\delta^{(1)}} \cdot \frac{(1 - \hat{\rho})^m(\hat{\delta}^{(1)})^m}{(1 - \hat{\rho}\hat{\delta}^{(1)})^m} \quad (21)$$

which is a function of only one variable  $\hat{b}$ , i.e., variable  $\hat{a}$ . This is because  $\hat{\rho}$  and  $\hat{\delta}^{(1)}$  are considered as functions of only one variable  $\hat{b}$ , i.e., variable  $\hat{a}$ . Hence, we obtain the cumulative distribution of the response time for a low-priority customer

$$F_{T^{(2)}}(s) = L^{-1} \left\{ \frac{\lambda^{(2)}}{s(s + \lambda^{(2)})} \cdot \frac{(1 - \rho)\delta^{(1)}}{1 - \rho\delta^{(1)}} \cdot \frac{(1 - \hat{\rho})^m(\hat{\delta}^{(1)})^m}{(1 - \hat{\rho}\hat{\delta}^{(1)})^m} \right\} \quad (22)$$

Since  $n_j^{(1)} = \lceil (\psi^{(1)})^{-1}(\frac{\hat{a} + \lambda_j^{(1)}}{\mu_j^{(1)}}) \rceil$  for a high-priority customer and  $n_j^{(2)} = \lceil (\psi^{(2)})^{-1}(\frac{\hat{b}}{\mu_j^{(2)}}) \rceil$  for a low-priority customer, the constraint of service availability at each resource site in (13) for a high-priority customer rewritten as

$$G_j^{(1)}(\hat{a}) \stackrel{def}{=} P_j^{(1)}(\lceil (\psi^{(1)})^{-1}(\frac{\hat{a} + \lambda_j^{(1)}}{\mu_j^{(1)}}) \rceil, N_j) \quad (23)$$

and the constraint of service availability at each resource site in (13) for a low-priority customer rewritten as

$$G_j^{(2)}(\hat{a}) \stackrel{def}{=} P_j^{(2)}(\lceil (\psi^{(2)})^{-1}(\frac{\hat{b}}{\mu_j^{(2)}}) \rceil, N_j) = P_j^{(2)}(\lceil (\psi^{(2)})^{-1}(\frac{\hat{a} + \lambda_j^{(1)}}{\mu_j^{(2)}}) \rceil, N_j) \quad (24)$$

Consequently,  $\sum_{j=1}^m n_j c_j$  reduces to a function of variable  $\hat{a}$  due to  $n_j^{(1)} = \lceil (\psi^{(1)})^{-1}(\frac{\hat{a} + \lambda_j^{(1)}}{\mu_j^{(1)}}) \rceil$  for a high-priority customer and to a function of variable  $\hat{b}$  due to  $n_j^{(2)} = \lceil (\psi^{(2)})^{-1}(\frac{\hat{b}}{\mu_j^{(2)}}) \rceil$  for a low-priority customer. That is,  $g(n_1, \dots, n_m)$  in (10) reduces to a function of one variable. Thus, the resource optimization problem is reduced into the one-dimensional resource optimization problems for both high-priority and low-priority customers. Hence, the trust-based resource provisioning problem is solved using the following algorithm.

### Algorithm

1. Find  $\hat{a}$  in the following two one-dimensional optimization problems.

(a) Minimization problem of a percentile response time for class  $r$  customers:

$$\hat{a}^{(r)}[1] \leftarrow \arg \min_{\hat{a}} F_{T^{(r)}}(t) \Big|_{t=T_D^{(r)}}$$

**Table 1. The Initial Trust Index of The Ten Stations**

Station	1	2	3	4	5	6	7	8	9	10
$I^{t_1}$	0.8	0.5	0.2	0.9	0.6	0.4	0.8	0.7	0.3	0.6

subject to the constraint  $F_{T^{(r)}}(t)|_{t=T_D^{(r)}} \geq \gamma^{(r)}\%$  at  $\hat{a} = \hat{a}^{(r)}[1]$ , where  $F_{T^{(r)}}(t)$  are given by (20) and (22) respectively. Then, calculate the number of servers  $n_j^{(r)}[1]$  required for a percentile response time guarantee by using

$$n_j^{(r)}[1] = \lceil (\psi^{(r)})^{-1} \left( \frac{\hat{a}_j^{(r)}[1] + \lambda_j^{(1)}}{\mu_j^{(r)}} \right) \rceil$$

for  $r = 1, 2$ , and  $j = 1, 2, \dots, m$ .

(b) Minimization problem of service availability:

$$\hat{a}_j^{(r)}[2] \leftarrow \arg \min_{\hat{a}} G_j^{(r)}(\hat{a})$$

subject to the constraint  $G_j^{(r)}(\hat{a}^{(r)}[2]) \geq \zeta_j^{(r)}\%$ , where  $G^{(r)}(\hat{a})$  are given by (23) and (24) respectively. Then, calculate the number of servers  $n_j^{(r)}[2]$  required for availability guarantee by using

$$n_j^{(r)}[2] = \lceil (\psi^{(r)})^{-1} \left( \frac{\hat{a}_j^{(r)}[2] + \lambda_j^{(1)}}{\mu_j^{(r)}} \right) \rceil$$

for  $r = 1, 2$ , and  $j = 1, 2, \dots, m$ .

2. Compute integers  $n_j^{(r)}$  by using

$$n_j^{(r)} = \max\{n_j^{(r)}[1], n_j^{(r)}[2]\}$$

Thus, the number of servers required is  $n_j = \max\{n_j^{(1)}, n_j^{(2)}\}$  for station  $j, j = 1, 2, \dots, m$ .

## 6 A Numerical Example

We note that the algorithm described in the section above is exact, under the assumption of balanced utilizations. The only source of error is due to the numerical implementation of a procedure for inverting the Laplace transform of the response time. In this section we demonstrate how the algorithm can be used to solve the trust-based resource provisioning problem.

Let us consider a ten resource site example modelled by the tandem queueing network presented in Section 5. We choose  $m = 7, \xi = 0.6$ , and the trust index at time  $t_1$  is listed in Table 1.

Assume that  $r_j(k)$  is uniformly distributed in  $[0.75, 1]$ .  $r_i$  and  $r_j$  are independent for any  $i \neq j, i, j = 1, \dots, 10$ . Furthermore, we choose  $\Lambda^{(1)} = 100, \Lambda^{(2)} = 50, T_D^{(1)} = 0.16, T_D^{(2)} = 0.8, \gamma^{(1)} = 97.5, \gamma^{(2)} = 99, \mu = 200$ , and the service rates of these seven stations are given in Table 2. We also

**Table 2. The Service Rates of The Ten Stations**

Service Rates	$\mu_1$	$\mu_2$	$\mu_3$	$\mu_4$	$\mu_5$	$\mu_6$	$\mu_7$	$\mu_8$	$\mu_9$	$\mu_{10}$
$\mu_j^{(1)}$	88	18	85	32	31	49	24	28	21	38
$\mu_j^{(2)}$	96	15	60	25	55	41	18	26	31	35

**Table 3. The Server Unavailability Rates of The Ten Stations**

Unavailable Rates	$\eta_1$	$\eta_2$	$\eta_3$	$\eta_4$	$\eta_5$	$\eta_6$	$\eta_7$	$\eta_8$	$\eta_9$	$\eta_{10}$
Values	0.012	0.01	0.005	0.04	0.015	0.03	0.02	0.045	0.008	0.01

choose  $c_j = 1$ ,  $N_j = 200$ ,  $\psi(n_j) = 1.55^{\log_2 n_j}$ ,  $\xi_j^{(1)} = 99.999$ ,  $\xi_j^{(2)} = 99.9$ ,  $\hat{I}_j^{(1)} = 0.9$ ,  $\hat{I}_j^{(2)} = 0.91$ ,  $j = 1, \dots, 10$ , and the server unavailability rates of these ten stations are listed in Table 3.

A customer submits a service job at time  $t_5$  with  $t_{k+1} - t_k = 0.01$ ,  $k = 1, 2, \dots$  and it requires two of these 5 resource sites satisfying the predefined  $\hat{I}_j$  for processing the job.

First, we generate  $I^{t_k}$ ,  $k = 2, \dots, 5, \dots, 21, \dots, 84, 85$  in Matlab. As is seen, while sites 1, 2, 3, 4, 6, 7, 8, and 10 meet the trust requirement for high-priority customers at  $t = t_5$ , only sites 2, 3, 4, 6, 7, 8, and 10 meet the trust requirement for low-priority customers at  $t = t_5$ . Thus, sites 2, 3, 4, 6, 7, 8, and 10 are selected because they meet both trust requirements.

We implemented (20) by using the inverse Laplace transform method in Graf [10]. This method is approximate because  $F_{T(r)}(t)$ ,  $r = 1, 2$ , given in (20) and 22) can usually be solved numerically. The error of this procedure is typically very small, and it was verified by comparing the numerical results against simulation. The simulation program was written in Arena 10.01, and simulated the queueing network shown in Figure 4. An example of the simulated and numerically calculated cumulative distribution of the high-priority and low-priority customers is given in Tables 5 and 6. The relative error in these two tables was calculated as follows:

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

**Table 4. The Trust Indices of The Ten Stations at Times  $t = t_2, \dots, t_5, \dots, t_{20}, t_{21}, \dots, t_{84}, t_{85}$** 

Station	1	2	3	4	5	6	7	8	9	10
$I^{t_2}$	0.8625	0.6412	0.5709	0.9156	0.7908	0.6651	0.8317	0.7892	0.3205	0.7428
$I^{t_3}$	0.8761	0.8190	0.7925	0.8501	0.8335	0.7698	0.8918	0.8593	0.4846	0.8865
$I^{t_4}$	0.8420	0.8532	0.8209	0.9014	0.7961	0.8725	0.9016	0.8946	0.4592	0.9132
$I^{t_5}$	0.9005	0.9129	0.9314	0.9248	0.8552	0.9153	0.9421	0.9189	0.5828	0.9588
...	...	...	...	...	...	...	...	...	...	...
$I^{t_{20}}$	0.8728	0.9215	0.9338	0.9182	0.8736	0.9326	0.9419	0.9546	0.6223	0.9235
$I^{t_{21}}$	0.8812	0.9513	0.9602	0.9278	0.9046	0.9462	0.9392	0.9698	0.8588	0.9351
...	...	...	...	...	...	...	...	...	...	...
$I^{t_{84}}$	0.9126	0.9318	0.9458	0.9288	0.8538	0.9225	0.9518	0.9449	0.7256	0.9338
$I^{t_{85}}$	0.8918	0.9815	0.9731	0.9388	0.9126	0.9565	0.9291	0.9588	0.8528	0.9588

**Table 5. The Cumulative Distribution of The High-priority Response Time**

Response Time	Simul	Approx	R-Err %
0.02	0.0002	0.0002	0.0000
0.04	0.0214	0.0214	0.0000
0.06	0.1542	0.1528	-0.9079
0.08	0.4085	0.4075	-0.2448
0.10	0.6691	0.6672	-0.2840
0.12	0.8459	0.8450	-0.1064
0.14	0.9385	0.9379	-0.0639
0.16	0.9781	0.9780	-0.0102
0.18	0.9931	0.9929	-0.0201
0.20	0.9980	0.9979	-0.0100
0.22	0.9995	0.9994	-0.0100
0.24	0.9998	0.9998	0.0000
0.26	0.9999	1.0000	0.0100
0.28	1.0000	1.0000	0.0000

(As shown in Table 6, the relative error is -16.6384% when  $t = 0.2$ . This is because the numerical computation of an inverse Laplace transform is an ill-posed problem and its accuracy heavily depends on the numerical calculation for the singularities of its image function at a point near the origin.)

Subsequently, we calculated the optimal number of servers required for satisfying both the high-priority and the low-priority response times. We assumed that 97.5% of the time the high-priority response time has to be less than  $T_D^{(1)} = 0.16$ , and 99% of the time the low-priority response time has to be less than  $T_D^{(2)} = 0.8$ . The results are shown in Table 7. We also calculated the exact optimal number of servers by exhaustive search using the simulation model, under the assumption of balanced utilizations. The results are identical to those shown in Table 7.

Furthermore, we obtain the number of servers required for 99.999% service availability guarantee for high-priority customers and for 99.9% service availability guarantee for low-priority customers in these seven stations using Step 2 (b) of our algorithm, as shown in Table 7. By doing the calculation in Step 3 of our algorithm, we obtained the number of servers required for the response time and service availability guarantees at these seven stations respectively.

Finally, the trust manager needs to determine whether to accept the job completed by sites 2, 3, 4, 6, 7, 8, and 10 when it receives the completed job at  $t = t_5 + T_D^{(1)} = t_5 + 0.16 = t_{21}$  for high-priority customers and at  $t = t_5 + T_D^{(2)} = t_5 + 0.8 = t_{85}$  for low-priority customers respectively. As is seen, these stations meet the trust requirements at  $t = t_{21}$  for high-priority customers and  $t = t_{85}$  for low-priority customers, and consequently the completed jobs are accepted.

## 7 Conclusions

In this paper, we have proposed a trust-based Web services model, and provided a framework for studying the model. We have discussed a trust-based resource provisioning problem that arises in these

**Table 6. The Cumulative Distribution of The Low-priority Response Time**

Response Time	Simul	Approx	R-Err %
0.2	0.1767	0.1473	-16.6384
0.3	0.4538	0.4396	-3.1291
0.4	0.6982	0.7082	1.4323
0.5	0.8541	0.8719	2.0841
0.6	0.9389	0.9503	1.2142
0.7	0.9774	0.9824	0.5116
0.8	0.9925	0.9942	0.1713
0.9	0.9978	0.9982	0.0401
1	0.9993	0.9995	0.0200
1.1	0.9998	0.9999	0.0100
1.2	1.0000	1.0000	0.0000

**Table 7. The Optimal Number of Servers**

The number of Servers	Stations						
Metrics	2	3	4	6	7	8	10
97.5% response time guarantee for high-priority customers	62	5	23	12	38	29	18
99% response time guarantee for low-priority customers	61	7	27	13	46	26	16
99.999% service availability guarantee for high-priority customers	10	7	22	18	15	23	10
99.9% service availability guarantee for low-priority customers	7	5	14	15	11	19	7
#Servers necessary to ensure all these SLA metrics	62	7	27	18	46	29	18

typical Web services applications. The problem has been constructed by minimizing the total cost of service providers while satisfying SLA guarantees. We have further formulated it as an optimization problem under the constraints of percentile response time and service availability.

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. We obtained an efficient and accurate numerical solution for calculating the percentile response time. Then, we proposed an efficient approach for solving the trust-based resource provisioning problem. We used a numerical example to illustrate the use of our proposed approach. Our validation tests showed that this approach has a good accuracy.

In this paper, we used a rank and a threshold-based approach to deal with the trust indices of resource sites, and assumed that the service discipline was FIFO and each resource site could repair its own servers. In our future work we will extend our work by evaluating different trust approaches and using other service disciplines with a general repairing mechanism for secure resource management in Web services applications and other distributed computing environments.

## References

- [1] F. Azzedin and M. Maheswaran. Evolving and managing trust in Grid computing systems. In *Proc. of the IEEE Canadian Conf. on Electrical Computing Engineering (CCECE '02)*. IEEE, May 2002.
- [2] E. Bouillet, D. Mitra, and K. Ramakrishnan. The structure and management of service level agreements in networks. *IEEE Journal on Selected Areas in Communications*, 20(4):691–699, 2002.
- [3] A. Brown and D. Patterson. Towards availability benchmarks: A case study of software RAID systems. In *Proceedings of 2000 USENIX Annual Technical Conference*. USENIX, June 2000.
- [4] A. Chandra, W. Gong, and P. Shenoy. Dynamic resource allocation for shared data centers using online measurements. In *Proceedings of Eleventh International Conference on Quality of Service (IWQoS 2003)*, June 2003.
- [5] J. Chang. Processor performance, Update 1. In *SQL-Server-Performance.com*.
- [6] C. Chassot, F. Garcia, G. Auriol, A. Lozes, E. Lochin, and P. Anelli. Performance analysis for an IP differentiated services network. In *Proceedings of IEEE International Conference on Communication (ICC'02)*, pages 976–980, 2002.
- [7] CISCO. Increasing network availability. In <http://www.cisco.com/warp/public/779/largeent/learn/technologies/ina/IncreasingNetworkAvailability-Abstract.pdf>.
- [8] CISCO. Service level management: Best practice. In <http://www.cisco.com/en/US/tech/tk869/tk769>.
- [9] H. Daduna. Burke's theorem on passage times in Gordon-Newell networks. *Adv. Appl. Prob.*, 16, 1984.
- [10] U. Graf. *Applied Laplace Transforms and z-Transforms for Scientists and Engineers*. Birkhauser Verlag, Basel-Boston-Berlin, 2004.
- [11] J. Hennessy. The future of systems research. *IEEE Computer*, 32(8):27–33, August 1999.
- [12] R. Jurca and B. Faltings. Reputation-based service level agreements for Web services. In *Third International Conference on Service Oriented Computing (ICSOC 2005)*. Amsterdam, The Netherlands, December 2005.
- [13] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. EignRep: Reputation management in P2P networks. In *Proceedings of the World-Wide Web Conference*, 2003.
- [14] R. Levy, J. Nagarajarao, G. Pacifici, M. Spreitzer, A. Tantawi, and A. Youssef. Performance management for cluster based Web services. In *The 8th IFIP/IEEE International Symposium on Integrated Network Management (IM2003)*. IEEE, 2003.
- [15] J. Martin and A. Nilsson. On service level agreements for IP networks. In *IEEE INFOCOM*, June 2002.
- [16] D. Menasce. QoS issues in Web services. *IEEE Internet Computing*, 6(4):72–75, November-December 2002.
- [17] D. Menasce. Response-time analysis of composite Web services. *IEEE Internet Computing*, 8(1):90–92, January-February 2004.
- [18] D. Menasce and M. Bennani. On the use of performance models to design self-managing computer systems. In *Proceedings of the 2003 Computer measurement group conference*. IEEE, 2003.
- [19] D. Menasce and E. Casalicchio. 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)*, pages 259–267. IEEE, October 2004.
- [20] J. Nabrzysbi, J. Schopf, and J. Weglarz. *Grid Resource Management*. Kluwer Academic Publication, Boston, MA, 2004.
- [21] T. Osogami, A. Wierman, M. Harchol-Balter, and A. Scheller-Wolf. How many servers are best in a dual-priority fcfs system? In *CMU Technical Report: CMU-CS-03-201*. Carnegie Mellon University, November 2003.
- [22] L. Vu, M. Hauswirth, and K. Aberer. QoS-based service selection and ranking with trust and reputation management. In *Infoscience's Technical Report*.
- [23] J. Walrand and P. Varaiya. Sojourn times and the overtaking condition in Jacksonian networks. *Adv. Appl. Prob.*, 12, 1980.

- [24] N. Wickramage and S. Weerawarana. A benchmark for Web service frameworks. In *Proceedings of the 2005 IEEE international conference on service computing*, pages 233–242. IEEE, July 2005.
- [25] K. Xiong and H. Perros. Computer resource optimization for differential customer services. In *Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2006)*. IEEE,, September 2006.
- [26] Q. Zhang, T. Yu, and K. Irwin. A classification scheme for trust functions in reputation-based trust management. In *International Workshop on Trust, Security, and Reputation on the Semantic Web*. Hiroshima,, November 2004.
- [27] C. Ziegler and G. Lausen. Spreading activation models for trust propagation. In *IEEE International Conference on e-Technology, e-commerce, and e-Service (EEE '04)*, April 2002.