Decentralized Market-Based Resource Allocation in a Heterogeneous Computing System

James Smith[1,2], Edwin K. P. Chong[2,4], Anthony A. Maciejewski[2], and Howard Jay Siegel[2,3]

[1]DigitalGlobe
Longmont, CO 80503 USA
Email: jtsmith@digitalglobe.com

Colorado State University
[2]Dept. of Electrical and Computer Engineering
[3]Dept. of Computer Science
[4]Mathematics Department
Fort Collins, CO 80523–1373, USA
Email: {echong, aam, hj}@engr.colostate.edu

Abstract

We present a decentralized market-based approach to resource allocation in a heterogeneous overlay network. The presented resource allocation strategy assigns overlay network resources to traffic dynamically based on current utilization, thus, enabling the system to accommodate fluctuating demand for its resources. We present a mathematical model of our resource allocation environment that treats the allocation of system resources as a constrained optimization problem. Our presented resource allocation strategy is based on solving the dual of this centralized optimization problem. The solution to the dual of our centralized optimization problem suggests a simple decentralized algorithm for resource allocation that is extremely efficient. Our results demonstrate the near optimality of the proposed approach through extensive simulation of a real-world environment. That is, the conducted simulation study utilizes components taken from a real-world middleware application environment and clearly demonstrates the practicality of the approach in a realistic setting.

1 Introduction

Recently, information technology (IT) systems have begun to rely heavily on the concept of Services Oriented Architecture (SOA). SOA is a means of leveraging existing applications as services within a distributed computing environment to develop new applications. One mechanism commonly used to integrate existing applications is known as the enterprise services bus (ESB) [16]. According to the Business Integration Journal [8], "The [ESB] supports the unifying integration infrastructure required for SOA and heterogeneous environments." By relying on an ESB to implement a distributed application, service requesters within the distributed application—using the ESB to communicate with service providers—can remain ignorant of the details of the service providers, e.g., the physical location of the service provider. Instead, service requesters depend on the abstract definition of the service that they are using and trust the ESB to forward their requests to an appropriate service provider. Because the service requester is not dependent on an explicit instance of a service provider, multiple service providers could be deployed within the ESB to provide additional capacity for a particular service.

An important aspect of an ESB implementation is that it can be decentralized both to increase its reliability and to ensure its scalability [8]. A common approach to increasing the reliability of a service is to duplicate that service many times across many hardware deployments—a technique often referred to as replication [7].

Successfully replicating ESB components requires maintaining network transparency [7], i.e., the user of the ESB infrastructure should be shielded from the existence of any redundant components used to provide the replicated ESB or its attached services. To the user, the system should appear as if it were a single highly available service that always has sufficient capacity to process ser-

This research was supported by the NSF under Grants CNS-0615170 and ECCS-0700559, by the Colorado State University Center for Robustness in Computer Systems (funded by the Colorado Commission on Higher Education Technology Advancement Group through the Colorado Institute of Technology), and by the Colorado State University George T. Abell Endowment.

vice requests. Achieving this kind of transparency in service delivery requires a mechanism for allowing the ESB infrastructure to adapt to changes in system load. That is, to transparently utilize replicas within an ESB infrastructure, the replication infrastructure must provide a mechanism for routing requests to their physical destination. In this work, we focus on the allocation of ESB components to the shipment of service requests; however, our presented approach can easily be extended to enable multiple service providers for a given service definition where the ESB components apply an analogous approach to allocating service requests to service providers.

This work investigates the application of a decentralized market-based approach to resource allocation within a heterogeneous deployment of a replicated ESB. In this system, service requesters send tasks to service providers using an overlay network provided by the ESB infrastructure. The decision of how to allocate individual ESB component capacity to service requests is made in a decentralized manner based on a quantification of current resource demand relative to current system capacity. Individual service requesters select a transmission rate through ESB components that maximizes their individual benefit, while the ESB components adjust "prices" for network links and their own computing capacity to reflect current demand. Thus, price setting enables service requester resource allocation decision making by communicating a simple quantification of current system congestion to the decision makers.

Figure 1 presents a graphical model of a simple overlay network provided by a replicated ESB. Service requesters are depicted in the figure as triangles, service providers are shown as squares, and ESB components are depicted as circles. Each service requester is connected to a collection of ESB components that "service" requests by dispatching them to a service provider capable of completing the request. The number of requests produced by each service requester may vary with time according to some unknown process. In this model, the capacity of the ESB components to service incoming requests may differ from one to another, i.e., the collection of ESB components are assumed heterogeneous in their performance [9, 11, 13]. Finally, each ESB component is connected to a collection of service providers by a finite-capacity link.

In a real-world computing environment, network traffic is often triggered by world events outside the control of the computing system. For example, a financial market sell off could reasonably be expected to result in an increase in network traffic communicating market sell orders to the financial market. In a similar manner, changes in the volume of service requests that a replicated ESB must process is a source of system uncertainty. That is, a resource manager responsible for allocating ESB components to service requests cannot accurately predict the upcoming volume of requests that will need to be serviced.

A system can be considered robust to perturbations in system parameters, if the change in system performance features due to this uncertainty is limited [1]. In this system, we utilize an overall performance measure that accounts for requester priorities, the number of requests being transferred, and the quality of the network routes being used. Requests are produced by service requesters at some rate and transmitted to the ESB where they are buffered before being forwarded on to their final destination. Because the system is decentralized and the production rate of service requests is changing with time, it is possible for the system to experience contention for shared system resources. When contention for shared resources occurs, potentially due to changes in the production rate of requests, it is possible for low-priority requests to inhibit the transfer of high-priority requests causing the performance measure to degrade.

Intuitively, the robustness [1, 4, 17, 18, 20] of the decentralized allocation approach can be established by answering the following three questions. What behavior of the system makes it robust? What uncertainties is the system robust against? Quantitatively, exactly how robust is the system? Uncertainty in how shared resources will collectively be used by service requesters can directly impact the system performance measure, given fluctuations in service request production rates. In this system, we might consider a resource allocation strategy robust as long as it maintains a performance measure that is within $X\%$ of the optimal value, where X is a user defined constraint on the acceptable performance of the system. We can quantify robustness in this environment as the proximity of the system performance measure to its optimal value.

Our mechanism for resource allocation can be thought of as a market-based approach where market demand for shared resources helps the system to set prices for those resources. It is common in market-based approaches to instead rely on an auction to create a market where prices are set by the highest bidder [10, 14]. In contrast, our approach utilizes price setting based on duality theory. Our approach is analogous to that used to implement congestion control on the internet [21] and in ad hoc sensor networks [5]. In our system model, price variables are introduced to model market demand for shared resources—prices are a simple quantification of current demand for shared resources. For example, as the number of requests through an ESB component increases, the ESB component raises its "price." Conversely, if the number of requests through an ESB component decreases, its quoted price also decreases. In addition to measuring the demand for the component itself, the ESB compo-

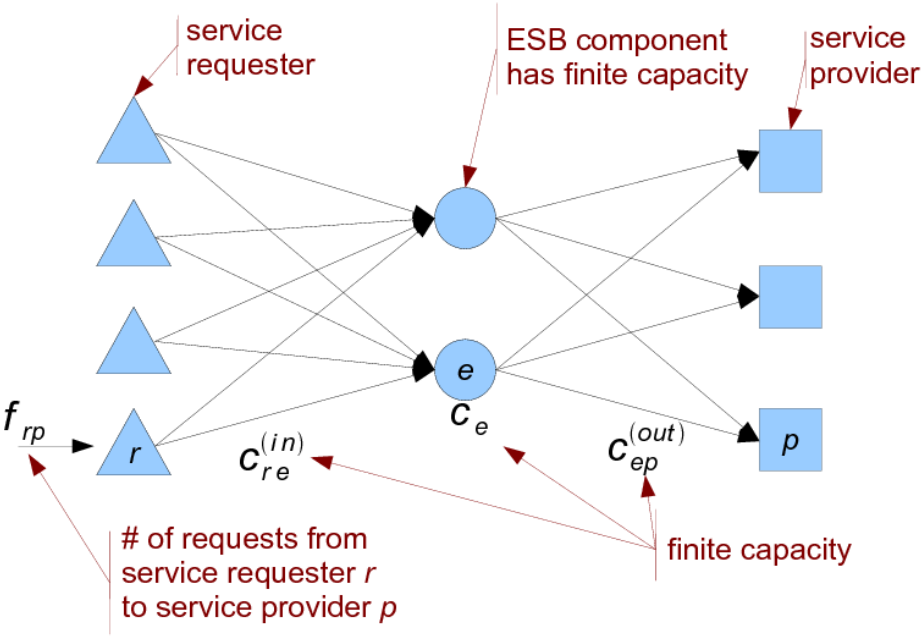

Figure 1. An example system where four service requesters are utilizing two ESB components to communicate with three service providers. The service requesters are shown as triangles, the ESB components as circles, and the service providers as squares. The input links to the ESB components from the service requesters each has a finite-capacity, denoted $c_{re}^{(\text{in})}$. The output links from the ESB components to the service providers each has a finite-capacity denoted $c_{ep}^{(\text{out})}$. Finally, each ESB component has a finite-capacity for servicing requests, denoted c_e.

nent is also responsible for stating the demand for the links from that ESB component to all of the service providers that the component can communicate. The procedure used to calculate an optimal pricing for resources will be presented. Individual service requesters directly utilize the current pricing information provided by the ESB components to make local resource allocation decisions about how best to assign their volume of requests within the overlay network given current network utilization.

Although an analogous methodology has been studied previously in the context of internet congestion control, it has never been applied within the context of an Enterprise Services Bus. A major contribution in this work is the design of a decentralized market-based approach to resource allocation. This approach is a novel use of market-based control over shared resources within an ESB environment. We also demonstrate through simulation that our decentralized market-based control mechanism is capable of providing near optimal resource management in an ESB setting.

In the next section, we present a more detailed view of the system model that is used in section 3 to present our mathematical model of the system. Section 4 presents a decentralized implementation of our solution approach. Sections 5 and 6 present our simulation setup and results for the evaluation of this approach in a real-world system setting, with conclusions presented in Section 8.

2 System Model

An example model of a replicated ESB overlay network is shown in Figure 1. In this computing system, there are three types of entities: service requesters, ESB components, and service providers. Service requesters send requests to ESB components that post the requests to the appropriate service provider. It is helpful to consider a service requester as an agent that is making service requests to the ESB on behalf of an application that is external to the replicated ESB system. That is, an external application passes requests to the service requester which makes routing decisions on behalf of the application. In this way, the service requester can be treated as a component of the ESB instead of as an outside entity. Let $\underline{\mathcal{R}}$ denote the set of all service requesters, $\underline{\mathcal{P}}$ the set of all service providers, and $\underline{\mathcal{E}}$ the set of all ESB components. In this system, rates of requests from individual service requesters are not required to be observable; instead, the system only requires that each ESB component be able to measure the number of requests that have arrived to it at each time step—a value readily available by inspection of the ESB component's input buffer. Each service requester r ($r \in \mathcal{R}$) produces requests for a service provider p ($p \in \mathcal{P}$) at a rate of $\underline{f_{rp}(t)}$ requests per unit of time. Service requesters are modeled as having an output buffer, and all requests produced by the requester are written to the output buffer prior to transmission. Requests are transmitted from the service requester's output buffer using the appropriate overlay network link to the input buffer of the selected ESB component. Each ESB component processes requests from its input buffer, forwarding requests to its chosen service provider.

In the example of Figure 1, service requesters are connected to all of the ESB components, i.e., there is a finite-capacity link connecting each service requester to each ESB component, where the finite capacity is modeled as a constraint on the transmission rate through that link. Thus, a link between a service requester $r \in \mathcal{R}$ and an ESB component $e \in \mathcal{E}$ is subject to a capacity constraint $\underline{c_{re}^{(\text{in})}}$ such that the rate of requests sent from service requester r to ESB component e cannot exceed $c_{re}^{(\text{in})}$.

Requests received by an ESB component are assumed to be buffered in a finite-capacity input buffer for that component. Each ESB component also maintains a finite-capacity output buffer for storing requests that have been processed and are ready to be transmitted to an appropriate service provider. Each ESB component $e \in \mathcal{E}$ is subject to its own capacity constraint on its computing capabilities such that the rate at which e can process requests is limited to $\underline{c_e}$. That is, an ESB component e with processing capacity c_e can move at most c_e requests from its input buffer to its output buffer in a single time unit.

The outgoing links from each ESB component to its attached service providers are subject to capacity constraints. The link connecting an ESB component e to a service provider $p \in \mathcal{P}$ has a finite capacity $\underline{c_{ep}^{(\text{out})}}$ to move data from the output buffer of ESB component e to the input buffer of service provider p. Lastly, there are no modeled constraints on the capacity of the service providers. That is, for this model we chose to focus on applying the market-based resource allocation strategy only to the ESB components. However, it should become clear that the presented approach could easily be extended to include allocation decisions regarding multiple service providers for the same class of service.

In our model, there are two types of shared resources, the ESB components and the links connecting the ESB components and the service providers. There is a difference between the advertised capacity (c_e and $c_{ep}^{(\text{out})}$) of shared resources and the physical capacity of a resource. The physical capacity of any given shared resource can be viewed as the "true" capacity afforded by the physical limitations of the device, i.e., the physical capacity is an upper bound on the performance of the device. It should be clear that if a system were sized such that it were required to operate at its physical limit for the duration of its execution, then if that system were ever asked to process more than that limit it would be incapable. That is, it is incumbent upon each ESB component e to construct its prices not from its true capacity but instead from its advertised capacity. Specifically, the advertised capacity is given as some reasonable fraction of the true system capacity, e.g., we can define the advertised capacity as $\underline{\rho}$ times the true capacity where $\rho \in (0, 1)$ is often called the "load factor." By communicating its availability in terms of the advertised capacity, the ESB component is insulated from instantaneous fluctuations that occur during the normal course of system operation that might otherwise overwhelm the component. The capacity for each shared resource used in our model (c_e and $c_{ep}^{(\text{out})}$) is assumed to be the advertised capacity.

The Information Technology (IT) industry is attempting to recover the cost of maintaining Wide Area Network (WAN) links by billing for network traffic that is transported across these links. That is, companies are beginning to monitor WAN traffic volume in an attempt to bill customers for the amount of data transferred across these expensive network links. In a globally distributed system such as a replicated ESB, network link costs need to be accounted for during resource allocation. By introducing a pricing scheme for WAN links, the network provider has inadvertently created a situation where some links are more valuable than others. To help illuminate the impact of such a decision, consider the following simple example. Using the model of Figure 1, assume that one ESB component is physically located in Fort Collins, CO, in the U.S.A., and another is located in Bangalore, India. If a service requester located in the U.S.A. intends to communicate with a service provider also in the U.S.A., then the lowest cost route may be through the ESB component located in Fort Collins. By sending traffic through this U.S.A. based ESB component a network charge can be avoided. Although somewhat exaggerated, this example demonstrates the heterogeneity of the available routes. The system model accounts for this heterogeneity by incorporating a quantification of route quality into the optimization problem. The value of each route from service requester r through ESB component e to a service provider destination p is given a single numeric value, denoted $\underline{s_{rep}}$, which quantifies the quality (e.g., speed) of the route to the service requester.

The goal of the model is to help ascertain the optimal allocation of ESB components and associated links to service providers for transmitting service requests. Recall, each service requester r produces $f_{rp}(t)$ service requests per unit time for a particular service provider p. This traffic is sent through some ESB component in $\mathcal{E}$ and all of the traffic must be eventually transmitted to its destination, e.g., some service provider p. Thus, we can identify the percentage of requests from a service requester r sent to a service provider p that are transmitted through ESB component e, denoted $\underline{g_{rep}}$ in the model. The goal of a resource management heuristic in this environment is to choose a combination of the g_{rep} values such that the system-wide "utility" is maximized.

In our system model, we assume that a service requester receives some utility from successfully transferring a request to a service provider. For example, the service provider may provide a printer repair service—the data being transferred by the service requester might be a notification of a printer outage. By successfully delivering the service requester's request to the service provider, the provider can dispatch a technician to repair the broken printer. Additionally, some service requesters may be more important than others, e.g., the system provider may wish to provide to some special customers a higher level of service than what is normally offered. That is, each service requester's traffic should be individually prioritized. Let θ_{rp} denote the priority of requests sent from service requester r ($r \in \mathcal{R}$) to service provider p ($p \in \mathcal{P}$). The utility, denoted $U(x)$, quantifies the worth of receiving service quality x.

3 Centralized Optimization

Given the discussion of the previous section, we pose the replicated ESB resource allocation problem as an optimization problem. The optimization problem can directly be solved in a centralized approach to resource allocation by a system-wide resource manager. The centralized approach could be a reasonable solution for the special case of a static rate of requests from each service requester. That is, if the rate of requests produced by all requesters in the system remains constant, then it may be reasonable to calculate a solution to the centralized problem off-line. However, if the rate of requests is changing with time, then the centralized solution becomes infeasible to maintain for all systems of reasonable size, i.e., the optimal solution to the problem changes faster than the centralized solution can be calculated. Similarly, if the centralized problem requires too many decision variables, i.e., there are too many service requesters, service providers, or ESB components, then it may be unreasonable to calculate the solution in advance off-line. That is, the centralized approach to resource allocation does not scale well, in this environment. We will use the centralized problem to motivate an alternative decentralized approach presented in the next section.

Using the definitions and system constraints from the previous section, the following centralized optimization problem can be defined, where each g_{rep} is to be chosen such that system resources are allocated optimally:

$$\text{maximize} \sum_{r,p} \theta_{rp} f_{rp}(t) U\left(\sum_{e} g_{rep} s_{rep}\right) \tag{1}$$

$$\text{subject to:} \quad \forall\ r,p, \sum_{e} g_{rep} = 1 \tag{2}$$

$$\forall\ r,e,p, \quad g_{rep} \geq 0 \tag{3}$$

$$\forall\ r,e, \sum_{p} g_{rep} f_{rp}(t) \leq c_{re}^{(\text{in})} \tag{4}$$

$$\forall\ p,e, \sum_{r} g_{rep} f_{rp}(t) \leq c_{ep}^{(\text{out})} \tag{5}$$

$$\forall\ e, \sum_{r,p} g_{rep} f_{rp}(t) \leq c_{e} \tag{6}$$

In the proposed centralized problem, overall achieved service quality is calculated as the weighted average of the service quality levels received where the s_{rep} values provide the weights. Equation (1) expresses the overall goal of the optimization to maximize the realized utility given a collection of service requesters each with their own priority. The constraint of Equation (2) ensures that all of the requests from a given service requester are routed through the overlay network. Finally, Equations (4), (5), and (6) enforce the capacity constraints for each of the constrained system resources and Equation (3) ensures that the g_{rep} values are positive.

Because the rate of requests sent from each service requester to each service provider, i.e., $f_{rp}(t)$, is a function of time, this centralized form of the problem must be solved whenever any of the request production rates change. However, for any problem of reasonable size the computation time required to solve this problem makes it difficult to complete the solution prior to the request production rates changing again. In the next section, we identify an equivalent solution that is much faster to calculate.

4 Decentralized Algorithm

4.1 Decentralized Approach

To transform the centralized optimization problem into a decentralized algorithm, we apply the Lagrangian multiplier method [2,3,6] to the centralized optimization problem to define an equivalent optimization problem that can be separated into $|\mathcal{R}|$ in-dependent sub-problems. The constraints of Equations (4) and (5) in the centralized problem reflect constraints on *shared* resources. The Lagrangian method provides a way to introduce lagrange multipliers (interpreted as "prices") for these shared resources that can be directly accounted for in the optimization criterion itself. An important feature of this approach is that solving for the g_{rep} values for each service requester no longer requires detailed knowledge of the g_{rep} values of the other service requesters. These detailed values are instead replaced by a collection of price vectors that are produced by the ESB components, where the price vectors reflect the current demand for the shared resources. Let $\underline{\pi_e}$ denote the price of ESB component e ($\forall e \in \mathcal{E}$) and let $\underline{q_{ep}}$ be the price for a link from component e to service provider p ($\forall e \in \mathcal{E}$, $\forall p \in \mathcal{P}$). Each service requester r must choose g_{rep} to solve the following problem:

$$\text{maximize} \sum_p \theta_{rp} f_{rp}(t) U \left(\sum_e g_{rep} s_{rep} \right) - \sum_{e,p} \left(\pi_e + q_{ep} \right) g_{rep} f_{rp}. \quad (7)$$

Service requester r also must enforce the following constraints corresponding to Equations (2), (3), and (4) from the centralized problem,

$$\forall\ p, \qquad \sum_e g_{rep} = 1 \quad (8)$$

$$\forall\ e,p, \qquad g_{rep} \geq 0 \quad (9)$$

$$\forall\ e, \sum_p g_{rep} f_{rp}(t) \leq c_{re}^{(\text{in})}. \quad (10)$$

The prices π_e and q_{ep} ($\forall e \in \mathcal{E}, \forall p \in \mathcal{P}$) are obtained from each ESB component e as input to the above model in the form of $\pi_e + q_{ep}$. An example of this communication is shown in Figure 2. In this example, service requester r_1 is sending requests to service provider p_1 using two ESB components e_1 and e_2. To facilitate the allocation decisions made by r_1 each of the ESB components communicates the price for using each of the shared resources that it is responsible for in the system, e.g., the ESB component and its link to p_1. In this simple example, there is only a single price to be communicated to the service requester. In more complicated systems, the ESB component is required to communicate prices for each complete route through the system that involves the component.

The above optimization problem only depends on the prices obtained from each ESB component and information that is locally available to the service requester. By reformulating the centralized optimization problem in this manner, we are able to obtain a collection of sub-problems whose combined solution is equivalent to the original centralized optimization problem. The final required ingredient in this model is an algorithm for computing the prices for each of the ESB components and the links connecting them to the service providers.

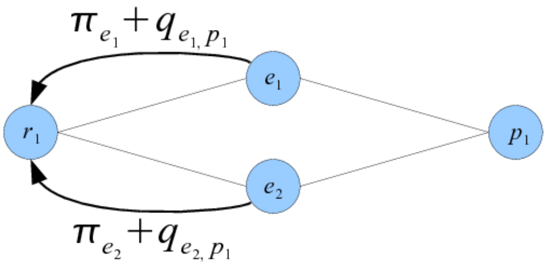

Figure 2. An example of the price update procedure. In the example, service requester r_1 receives updates for the price of using ESB component e_1 and ESB component e_2 to communicate with service provider p_1.

Each ESB component e is required to update prices in the model for its shared resources such that the prices reflect expected near-term future demand. The price of each shared resource reflects the amount of excess capacity that the resource has for processing requests. If the demand for a shared resource is greater (less) than the supply, then the price of the resource should increase (decrease). In our approach, we utilize a constant step-size for applying price updates. That is, the price of each shared resource is updated according to the difference between the capacity of the shared resource and current demand for the resource scaled by a constant factor. In our model, we differentiate between updates to ESB component prices and link prices by introducing two different constant step-size values, one for the ESB component prices, denoted $\underline{\alpha}$, and one for the link prices, denoted $\underline{\gamma}$—both step-sizes must be greater than 0. The ESB component e can simply update prices π_e and q_{ep} ($\forall p$) using the following update procedure where $[x]_+ = \max\{x, 0\}$:

$$\pi_e(t+1) = \left[\pi_e(t) - \alpha \left(c_e - \sum_{rp} g_{rep} f_{rp}(t) \right) \right]_+$$

$$q_{ep}(t+1) = \left[q_{ep}(t) - \gamma \left(c_{ep}^{(out)} - \sum_r g_{rep} f_{rp}(t) \right) \right]_+$$

The step-sizes α and γ determine how the system will react to fluctuations in the demand for its shared resources. Notice that the step-size determines the magnitude of price updates. That is, if α or γ is too small the prices will be slow to react to changes in demand leading to potentially large buffer sizes. For example, if price updates are too small and the demand for a resource is much greater than its capacity then, the price for the shared resource may not increase enough to deter requesters from using it. Consequently, the number of queued requests in the input buffer of the shared resource may increase because the resource is consistently receiving more requests than it can process. The values for α and γ need to be large enough to enable the system to react to substantial changes in demand, i.e., increase the price enough to deter excessive demand. Care must also be taken to prevent the opposite scenario, where prices are increased so much that future demand for the shared resource is reduced to 0. If α and γ are set too high, then the system may thrash, i.e., demand may oscillate between shared resources potentially overwhelming some resource in any given time step.

4.2 Resource Management Using this Approach

Each service requester r must have a procedure for utilizing the results obtained by solving the local optimization problem for the g_{rep} values. Because the service requester cannot send fractions of a record, we need to approximate the direct use of the g_{rep} decision variables. The simplest mechanism for converting the g_{rep} decision variables into a resource allocation is to interpret their values as probabilities. Recall that the g_{rep} values are constrained to the interval [0,1]. Thus, each g_{rep} value can represent the probability that any given record will utilize the route from service requester r through ESB component e to service provider p. To make use of the probability, service requester r generates a uniform random number indexing into the available routes according to the g_{rep} probabilities to select a route for each request to be processed. Using the g_{rep} values in this way will, over many sent records, approximate the direct use of the g_{rep} values.

5 Simulation Setup

Several simulations were conducted to evaluate both the accuracy of the implemented system as well as the efficacy of the overall approach. To evaluate the efficacy of the approach in a realistic setting requires establishing a variable production rate for each service requester in the environment. The variable production rate of records in the system is modeled using a simple scaled sinusoid that provides a periodic change in record production. The record production rate for each of the service requesters modeled in this simulation varies according to a uniquely scaled sinusoid. The superposition of the record production functions is presented in Figure 3(a). Each point in the plot corresponds to the number of records produced in the simulation at that time step and presents a view of the load placed on the overall system in that time step. The combination of sinusoids chosen is such that the summed traffic is periodic, repeating the exact same pattern of production approximately every 2200 time steps.

The plot of Figure 3(a) can be scaled according to the priority of each service requester (θ_{rp}) producing a plot (Figure 3(b)) of the optimal utility given homogeneous network routes, i.e., all routes are of equal value ($\forall r, e, p \quad s_{rep} = 1$). Ideally, the solution found by the described decentralized routing mechanism will track this optimal utility function.

The simulations conducted consisted of four service requesters, two ESB components, and three service providers where collective the production rate of the service requesters is varied according to Figure 3(a). For this simulation, we compared the results of the centralized solution to the results of our decentralized solution to demonstrate the effectiveness of our approach. We did this by periodically extracting the information required to produce the centralized optimization problem from the details of the simulation at a given time step. We solved these instances of the centralized problem offline and compared the centralized result to our decentralized approach.

6 Results

In a real-world environment, the rate of requests submitted to the system will vary with time unpredictably. To assess the viability of our approach in this kind of environment, we modeled the production rate of requests from service requesters as a function of time. Thus, at each instant in time the optimal allocation is given by a unique optimization problem with its own unique solution. Consequently, a centralized solution in this environment is impractical because it would require recalculating the entire solution at every time-step. In contrast, in the decentralized approach each service requester can effectively solve their own local optimization problem using the prices provided by the ESB components. The combined results of all of the service requesters should be the same as the centralized solution—the decentralized implementation should be capable of tracking the optimal solution as it varies over time.

Figure 4 presents the results for a sample implementation where service requester production rates are functions of time and all routes in the system are of equal quality, i.e., all $s_{rep} = 1 \ \forall r, e, p$. The three plots of Figure 4 present (a) the total number of requests at each time step during the simulation, (b) the realized utility over time, and (c) the average price for shared resources in the system over

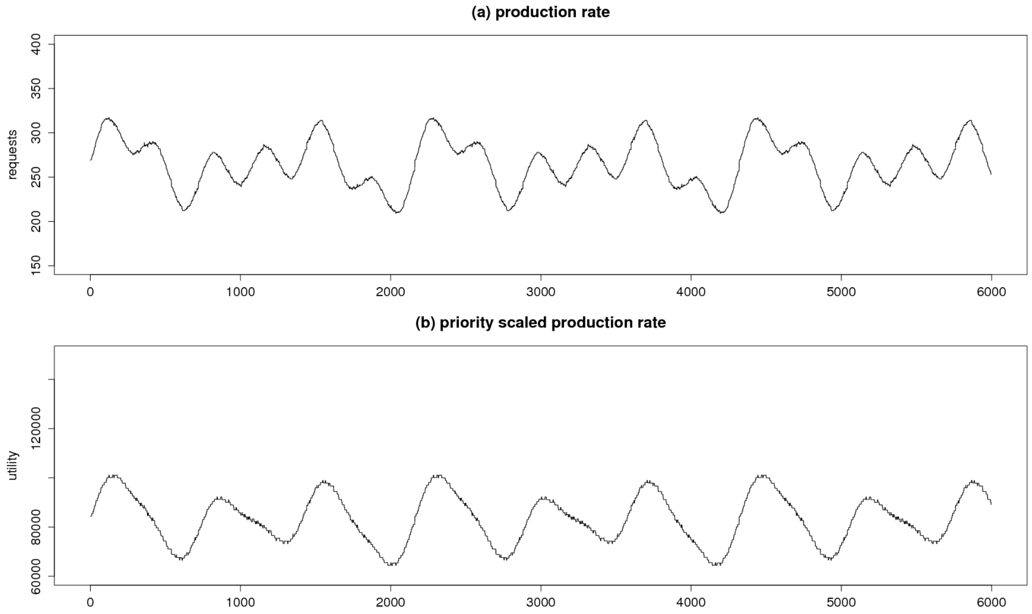

Figure 3. The figure includes two plots, (a) the summed production rates over all service requesters plotted versus simulation time step and (b) the summed production rates scaled by service requester priority plotted versus simulation time step.

time.

Embedded within the plot of realized utility (Figure 4 (b)), we have plotted the exact centralized solution overlaid as circles on top of the decentralized solution. From the plots of the figure, we can clearly see that the decentralized solution tracks the interpolated centralized solution. However, there is a slight (less than 1%) degradation in performance due to the buffering of data within the system. That is, each service requester service provider pair has a different priority and in any given instant there may be minor contention for a shared resource. When this contention occurs some high priority traffic may be delayed due to the system processing lower priority traffic first. Thus, the realized utility will be temporarily reduced as a result of the delay but will increase in some subsequent time step as the delayed records arrive at their destination.

An alternative approach to evaluating the results of the decentralized solution is possible when the quality of the given routes are homogeneous, i.e., $s_{rep} = 1 \; \forall r, e, p$. In this case, it is possible to compare the results of the decentralized solution to the sum of the scaled request production rate over all service requesters. In other words, we sum the decentralized realized utility over each time step for the entire simulation and do the same for the scaled request production rate. Ideally, these two values should be identical given homogeneous routes, and they are nearly in fact our results produce 99.5% of the scaled production rate result, i.e., they are nearly identical.

The complete simulation includes routes of differing quality, where the s_{rep} values differed from one route to another. Recall that the s_{rep} values appear in the utility function as a multiplier for the g_{rep} values. In this way, the g_{rep} values are scaled according to the s_{rep} values prior to calculating utility. Consequently, a centralized controller might attempt to try and maximize the likelihood that high priority traffic will be assigned to its best route, thus maximizing the realized utility. In Figure 5, we can see that given heterogeneous routes the results of the decentralized solution still effectively track the results of the centralized solution.

In both Figures 4(c) and 5(c), we plotted the average shared resource price for the entire simulation. Notice that the prices are periodic with a period identical to that of the production rates plotted in Figures 4(a) and 5(a) respectively. For these simulations the system appears stable. That is, because the request production rate is periodic, a stable system would imply that the average shared resource price would return to its starting point at the end of the period. In our simulations, the initial configuration of the system represents a slightly overprovisioned system. Thus, the initial prices of the shared resources are all 0. At the end of the period, we should expect the average shared resource price to return to 0 and it does. Recall that the periodicity of the request production rate is such that the pattern repeats approximately every 2200 time

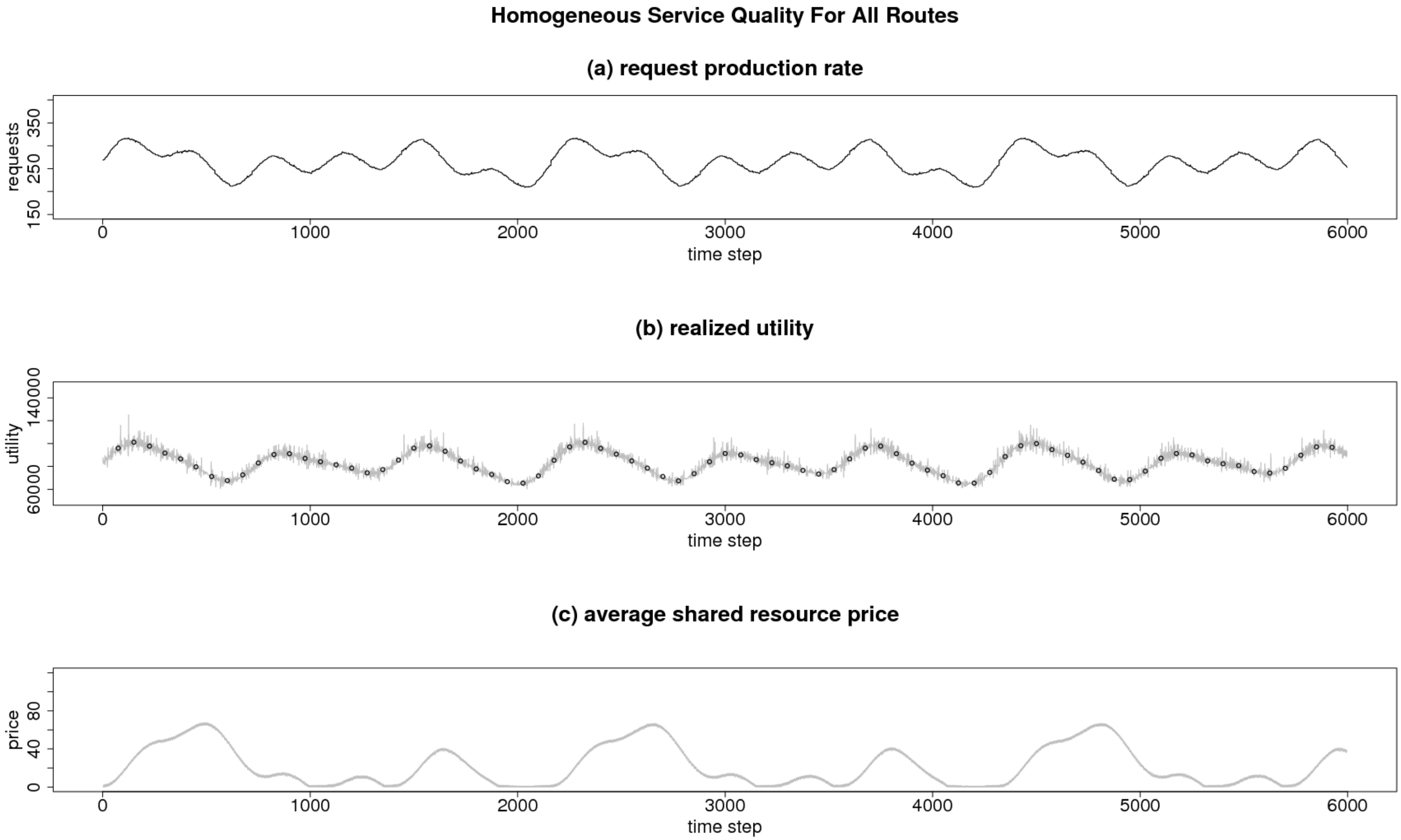

Figure 4. Sample results for a homogeneous network, i.e., $s_{rep} = 1 \ \forall r, e, p$, given service request production rates that vary as functions of time. Plot (a) of the figure presents the collective request production rate for the system as a function of time. Plot (b) presents the realized utility for our decentralized approach (plotted as a line). Periodically, throughout the simulation, an equivalent centralized optimization problem was extracted from the state of the simulation at a given time-step and solved. These results are plotted as circles in Plot (b) overlaid on top of the decentralized solution. Plot (c) presents the average shared resource price in the network as a function of time.

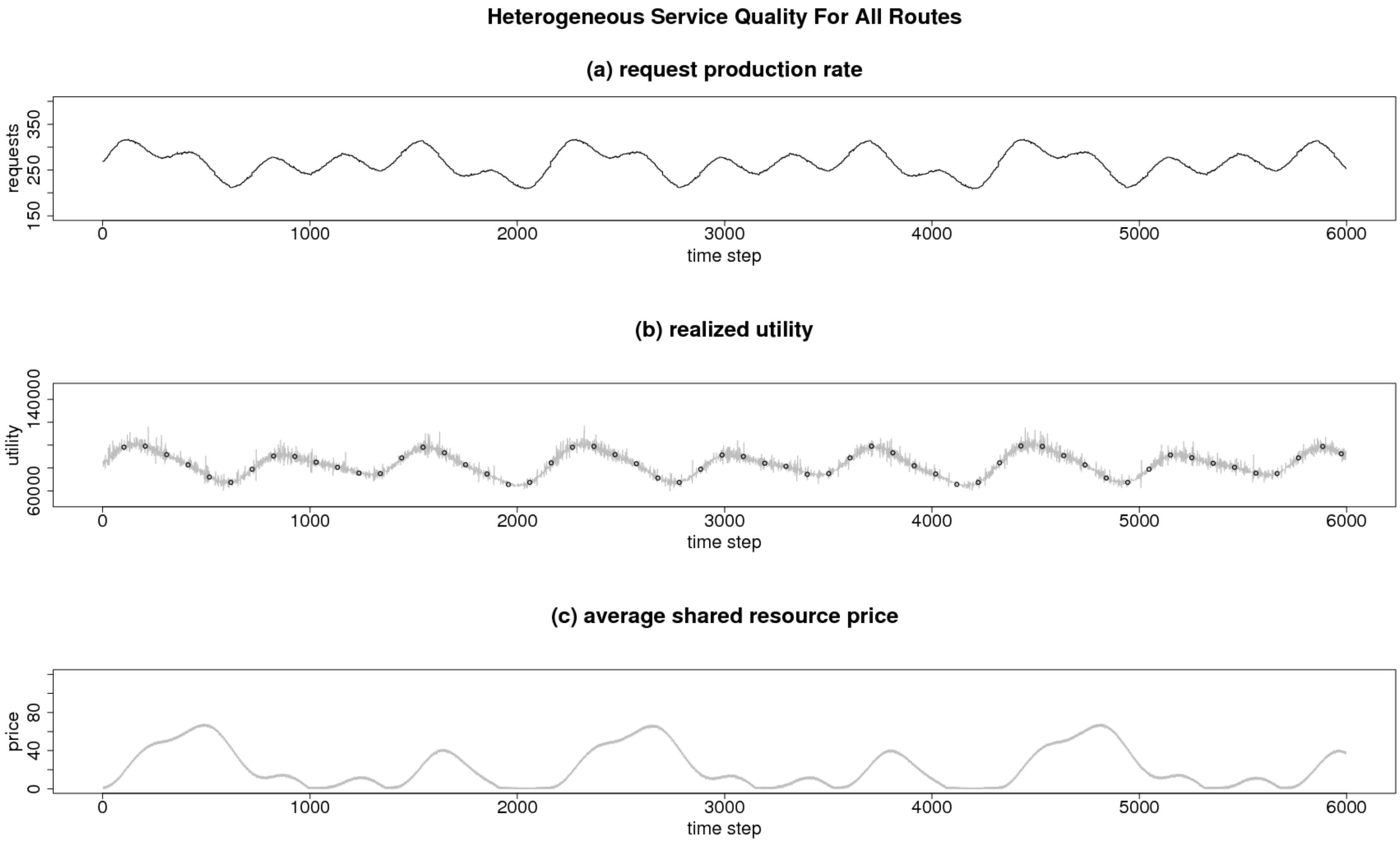

Figure 5. Sample results for a heterogeneous network, i.e., each through the network connecting a service provider to a service requester has a unique service quality associated with it. Plot (a) of the figure presents the collective request production rate for the system as a function of time. Plot (b) presents the realized utility for our decentralized approach (plotted as a line). Periodically, throughout the simulation, an equivalent centralized optimization problem was extracted from the state of the simulation at a given time-step and solved. These results are plotted as circles in Plot (b) overlaid on top of the decentralized solution. Plot (c) presents the average shared resource price in the network as a function of time.

steps; similarly the average shared resource price is periodic repeating the exact same pattern every 2200 time steps.

7 Related Work

A related field to the study of an Enterprise Services Bus is that of a Content Delivery Network (CDN) commonly used to improve the apparent performance and reliability of web sites by distributing their content throughout the world wide web. In a CDN, web-site content is cached at replica servers that are capable of replying to web requests on behalf of the owning web site. In [15], the authors introduce the concept of a collaborative CDN (CCDN). A CCDN is described as being an overlay network that utilizes end-user machines in a peer-to-peer fashion to provide a CDN across a wide-area network. In the Globule system, user requests for data available on the CCDN are delivered to replica servers using a redirection service capable of HTTP redirection. In [19], the authors present the AS-path length heuristic used in Globule to provide a redirection policy for user requests. The AS-path length heuristic greedily redirects user requests to the closest replica server available in the CDN where proximity is defined in terms of the number of network hops between the requester and the replica. This simple greedy approach does not account for contention among the shared resources of the CCDN, i.e., the replica servers. Because the Globule system is an instance of an overlay network it can be modeled as a transshipment network flow problem. By modeling an overlay network in this manner, our market based resource allocation technique can be applied to the routing of web-site requests to replica servers based on current network load where the proximity of the requester to the caches and the network bandwidth of the caches can be used to construct a quality value for each route in the CDN, i.e., s_{rep}. In this way, our approach can account for both the proximity of replicas to users as well as any contention for the shared resources of the CCDN.

Another approach to market based resource allocation involves the use of an auction as opposed to price setting. The Tycoon resource allocation system presented in [14] provides such an auction based market for resource allocation in a distributed system. In the Tycoon system, users bid for the right to use compute resources within the Tycoon network. Bids are accepted by a collection of auctioneers that manage access to Tycoon compute resources. In an auction system, the auctioneer must accept bids for a resource for some period of time before closing the auction. The waiting period for an auction to close is acceptable as long as the time required to complete the auction is less than that of the task to be executed. In our environment, tasks are extremely short lived, e.g., the time required to produce a static web page or deliver a message in an overlay network. Consequently, the time delays incurred by an auction for a resource are infeasible within our context. However, prices for shared resources are set within our network based on current demand. Thus, our price setting approach is more analogous to a bid-ask auction system where the resource seller sets an asking price and the buyer accepts that price by purchasing the right to that resource. In this way, as demand for a shared resource fluctuates so do the prices for that resource.

In [12], a system called WebSeAl is introduced that provides resource allocation in a CDN. One of the many claims of the WebSeAl system is its ability to balance the request load for a web site across multiple geographically dispersed replica web servers. Their approach to resource management of the server pool is to introduce prices for the use of servers in their network that force the clients to route their requests based on this price information. Clients in the WebSeAl environment make routing decisions based on a combination of performance data about the response time of each replica server and a weighting factor for each replica. The authors assume that clients in the WebSeAl environment will be “sensitive” to the weighting factor and account for current system weights while making resource allocation decisions. By their own admission the WebSeAl environment is therefore best suited to serving web sites where there is no incentive to circumvent the balancing aspects of the system, e.g., web sites delivered on a corporate intranet. By ignoring the weighting factor a client may instead request solely based on selfish performance data, i.e., always selecting the replica that provides the best possible performance to the client.

Like the WebSeAl environment our system utilizes a price setting scheme to enable clients in the system to make routing decisions. However, unlike WEbSeAl prices in our environment are set based on direct feedback from the system regarding current demand for shared resources. Further, the clients (service requesters) in our system solve a local optimization problem that leverages current prices for shared resources to account for network congestion. By solving the local optimization problem to maximize their individual utility, the system as a whole is able to maximize its realized utility.

8 Conclusion

In this paper, we have demonstrated a technique for resource allocation in an overlay network that is derived from Lagrangian optimization techniques similar to those of internet congestion control. Our approach has some clear advantages over some obvious solutions for routing data within an overlay network. Principally, our decentralized approach is capable of producing a near optimal assignment while still maintaining some of the more attractive attributes of a decentralized solution, e.g., scalability and reliability. Throughout this paper we have assumed that a feasible solution to the centralized allocation problem exists. In future work, we will remove this assumption and consider problems where the system may require additional capacity to convert an infeasible problem into a feasible one, i.e., by adding additional ESB components to increase system capacity.

References

[1] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim. Measuring the robustness of a resource allocation. *IEEE Transactions on Parallel and Distributed Systems*, 15(7):630–641, July 2004.

[2] D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, MA, second edition, 2003.

[3] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Prentice Hall, Englewood Cliffs, NJ, USA, first edition, 1989.

[4] L. Bölöni and D. Marinescu. Robust scheduling of metaprograms. *Journal of Scheduling*, 5(5):395–412, Sept. 2002.

[5] E. K. P. Chong and B. E. Brewington. Decentralized rate control for tracking and surveillance networks. *Ad Hoc Networks, special issue on Recent Advances in Wireless Sensor Networks*, (6):910–928, Aug. 2007.

[6] E. K. P. Chong and S. H. Zak. *An Introduction to Optimization.* John Wiley, New York, NY, second edition, 2001.

[7] G. Coullouris, J. Dollimore, and T. Kindberg. *Distributed Systems Concepts and Design.* Addison Wesley, Harlow, England, fourth edition, 2005.

[8] K. Emo. The ‘enterprise-class’ service bus: IT’s direct route to SOA. *Business Integration Journal*, Oct. 2005.

[9] M. M. Eshaghian, editor. *Heterogeneous Computing.* Artech House, Norwood, MA, 1996.

[10] M. Feldman, K. Lai, and L. Zhang. A price-anticipating resource allocation mechanism for distributed shared clusters. In *EC ’05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 127–136, New York, NY, USA, 2005. ACM Press.

[11] R. F. Freund and H. J. Siegel. Heterogeneous processing. *IEEE Computer*, 26(6):13–17, June 1993.

[12] M. Karaul, Y. A. Korilis, and A. Orda. A market-based architecture for management of geographically dispersed, replicated web servers. *Decision Support Systems*, 28(1-2):191–204, Mar. 2000.

[13] A. Khokhar, V. K. Prasanna, M. E. Shaaban, and C. Wang. Heterogeneous computing: Challenges and opportunities. *IEEE Computer*, 26(6):18–27, June 1993.

[14] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. A. Huberman. Tycoon: an implementation of a distributed, market-based resource allocation system. *Multiagent and Grid Systems*, 1(3):169–182, 2005.

[15] G. Pierre and M. van Steen. Globule: a collaborative content delivery network. *IEEE Communications Magazine*, 44(8):127–133, Aug. 2006.

[16] M. T. Schmidt, B. Hutchison, P. Lambros, and R. Phippen. The enterprise service bus: Making service-oriented architecture real. *IBM Systems Journal*, 44(4):781–797, 2005.

[17] V. Shestak, H. J. Siegel, A. A. Maciejewski, and S. Ali. The robustness of resource allocations in parallel and distributed computing systems. In *Architecture of Computing Systems – ARCS 2006, 19th International Conference Proceedings*, pages 17–30, Mar. 2006.

[18] V. Shestak, J. Smith, H. J. Siegel, and A. A. Maciejewski. A stochastic approach to measuring the robustness of resource allocations in distributed systems. In *Proceedings of the 2006 International Conference on Parallel Processing (ICPP 2006)*, pages 6–12, Aug. 2006.

[19] S. Sivisubramanian, B. van Halderen, and G. Pierre. Globule: A user-centric content delivery network. In *Proceedings of the 4th International Systems Administration and Network Engineering Conference (SANE 2004)*, 2004.

[20] J. Smith, L. D. Briceño, A. A. Maciejewski, and H. J. Siegel. Measuring the robustness of resource allocations in a stochastic dynamic environment. In *Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS 2007)*, Mar. 2007.

[21] R. Srikant. *The Mathematics of Internet Congestion Control.* Birkhauser, Boston, MA, 2003.