Collective Value of QoS: A Performance Measure Framework for Distributed Heterogeneous Networks

Jong-Kook Kim[1], Taylor Kidd[2],
Howard Jay Siegel[1*], Cynthia Irvine[2], Tim Levin[3], Debra A. Hensgen[2],
David St. John[3], Viktor K. Prasanna[4], Richard F. Freund[5], and N. Wayne Porter[2]

[1]Purdue University
Electrical and Computer Engineering School
West Lafayette, IN 47907-1285, USA
{kim42, hj}@purdue.edu

[2]Naval Postgraduate School
Department of Computer Science
Monterey, CA 93943, USA
dhensgen@opentv.com
kidd@acm.org
irvine@cs.nps.navy.mil
Norman.Porter@jac.af.mil

[3]Anteon Corporation
2600 Garden Rd., Ste. 220A
Monterey, CA 93940, USA
levin@cs.nps.navy.mil
david@marakservices.com

[4]University of Southern California
Department of Electrical
Engineering-Systems
Los Angeles, CA 90089, USA
prasanna@ganges.usc.edu

[5]NOEMIX Inc.
1425 Russ Blvd., Ste. T-110
San Diego, CA 92101 USA
rffreund@noemix.com

Abstract

When users' tasks in a distributed heterogeneous computing environment are allocated resources, and the total demand placed on system resources by the tasks, for a given interval of time, exceeds the resources available, some tasks will receive degraded service, receive no service at all, or may be dropped from the system. One part of a measure to quantify the success of a resource management system (RMS) in such an environment is the collective value of the tasks completed during an interval of time, as perceived by the user, the application, or the policy maker. For the case where a task may be a data communication request, the collective value of data communication requests that are satisfied during an interval of time is measured. The Flexible Integrated System Capability (FISC) measure defined here is one way of obtaining a multi-dimensional measure for quantifying this collective value. While the FISC measure itself is not sufficient for scheduling purposes, it can be a critical part of a scheduler or a scheduling heuristic. The primary contribution of this work is providing a way to measure the collective value accrued by an RMS using a broad range of attributes and to construct a flexible framework that can be extended for particular problem domains.

1. Introduction

Mixed-machine heterogeneous computing (HC) environments provide a distributed suite of different types of machines, connected by diverse types of networks, to meet the varied computational and input requirements of widely varying task mixtures (e.g., [4, 9, 26]). The goals of a resource management system (RMS) in an HC environment are to assign communication, computation, and other resources in an attempt to satisfy users' requests, which may require different types and levels of quality of service (QoS). When users' tasks in a distributed heterogeneous computing environment are allocated resources, and the total demand placed on system resources by the tasks, for a given interval of time, exceeds the resources available, some tasks will receive degraded service, receive no service at all, or may be dropped from the system.

This research was supported by the DARPA/ITO Quorum Program, by the DARPA/ISO BADD Program and the Office of Naval Research under ONR grant number N00014-97-1-0804, and by the DARPA/ITO AICE program under contract numbers DABT63-99-C-0010 and DABT63-99-C-0012.

[*]Beginning Aug. 2001, H. J. Siegel will be a professor of ECE at Colorado State University.

As described in [10], evaluation of performance is an essential activity, and for any system, the most important condition to satisfy is that the system performs its functions correctly. Therefore, in the evaluation of the performance of an RMS, it is essential to measure: (1) how well it performed its goals or functions, (2) if it performed correctly, and (3) how well it performed compared to other RMSs.

The goal of this research is to quantify the collective value of the tasks completed during an interval of time, as perceived by the user, application, or policy maker. Intuitively, if one RMS performs better than another in a given environment, the better RMS would have a higher collective value. This measure can be a part of a metric to assess the success of an RMS in a certain environment (other parts may include execution time, ability to work with different operating systems, and user interfaces). This research describes attributes that can be combined to derive such a performance measure, provides a way to combine them, and discusses other issues concerning the performance measure.

The proposed approach is called the Flexible Integrated System Capability (FISC) measure. This research extends our earlier work ([15]) by exploring a different way to combine the attributes to derive the FISC measure and introducing some other issues concerning various aspects of the measure.

The FISC measure is a multi-dimensional performance measure, and may include factors such as priorities, versions of tasks or data, deadlines, situational mode, security, application- and domain-specific QoS, and task dependencies. The FISC measure is a flexible framework for quantifying the collective value of a set of tasks completed during a given interval of time in an overloaded distributed system. It provides one way of combining the factors described. For the case where a task may be a data communication request, the collective value of data communication requests that are satisfied during an interval of time is measured.

The FISC measure by itself is not a scheduler evaluator; other factors, such as scheduler execution time, need to be included for the FISC measure to be a part of a scheduler evaluator. The FISC measure by itself is not a scheduling heuristic, where parameters such as urgency (time to deadline) or matching of task requirements to machine capabilities are usually included (e.g., [4, 25, 38]).

The FISC measure can be used to determine the scheduling heuristic that results in the highest value of the tasks completed, for a given environment. It can also be used to compare, via experimentation or simulation, the effectiveness of changing the resources available in a given distributed system. Furthermore, the FISC measure can be incorporated as part of the objective function in a system's scheduling heuristics.

There are varieties of performance measures that can be considered when analyzing systems (e.g., [36]). In some situations a combinations of QoS (or performance) attributes must be considered (e.g., [23]). The FISC measure will be focused on calculating the value of the tasks completed using various QoS attributes. The measure presented here is one instantiation of the FISC measure where it is a linear measure. In the generalization subsection (Subsection 4.8) of this work, a non-linear measure is discussed.

This research is part of three related DARPA activities: the DARPA Quorum-sponsored Management System for Heterogeneous Networks (MSHN) project [12], the DARPA Battlefield Awareness and Data Dissemination (BADD) program [8, 32], and the Agile Information Control Environment (AICE) program [1]. While this research is motivated by military applications, the FISC measure can be adapted for other environments, such as industrial and government agency intra-nets (e.g., NASA), clusters of PCs and workstations, and computational grids [11].

The test of the goodness of a performance measure for an HC RMS is if it allows a system administrator the flexibility to quantify how it is desired for the system to behave. Furthermore, the performance measure should provide a vehicle for comparing the results achieved by different RMSs given the same operational scenario, so that the best RMS for a given environment can be selected. The FISC measure has these qualities. Thus, the primary contribution of this work is providing a way to measure the collective value accrued by an RMS using a broad range of attributes and to construct a flexible framework that can be extended for particular problem domains.

The next section mentions some of the literature related to this work. In Section 3, several examples of individual QoS requirements are presented. Section 4 shows how all the example QoS attributes can be combined into a single measure. In addition, this section presents other issues such as averaged FISC versus multiplied FISC, multiple copies of versions, coefficients for the FISC measure, priority levels within classes, and a generalized form of the performance measure. A comparison of game theoretical solution using the FISC measure is provided in Section 5. The last section gives a brief summary of this research.

2. Related Work

The FISC performance measure discussed here embodies parameters that are considered important in scheduling tasks and communications on a distributed computing system. There is much literature on the

scheduling and mapping of tasks, messages, and data items. In this section, some examples of this literature are mentioned. This is followed by a discussion of examples of prior performance measure studies that the FISC measure extends.

An optimistic priority-based concurrency control protocol that schedules active transactions with a deadline in real-time database systems is described in [16]. This protocol combines forward and backward validation processes to control more effectively concurrent transactions with different priorities. The protocol is designed such that deadlines of higher priority transactions have a greater probability of being met than those of lower priority transactions. While this is also the case for MSHN and BADD/AICE, the FISC research presented here includes other attributes that are important in evaluating the overall value of the tasks completed.

In [24], priority is used for the mapping, adjustment, and dropping of messages. The priority of a task is adjusted by the length of time it was blocked by a higher priority message, and tardy messages that already missed their deadlines are dropped. This Least-Laxity-First priority mapping gives an improved missed deadline ratio, which is the rate of messages missing their deadlines. In [24], priorities and deadlines are used as measures of performance but the research effort does not consider other QoS attributes used in heterogeneous distributed networks that the FISC measure includes.

The work presented in [28] describes an algorithm that enables each node of a system to schedule the transmission of messages generated locally while observing their constraints. This algorithm uses the actual priority and the deadline of a message for the scheduling of the messages. The FISC measure allows more than one simple deadline and includes other important QoS attributes that can be considered in the calculation of the collective value of tasks completed, which can be used as part of a scheduling process.

Data staging, an important data management problem for a distributed heterogeneous networking environment, is discussed in [38]. The research in [38] assumed that each requested data item is associated with a specific deadline and priority. The FISC research presented here generalizes the performance measure used in [38] to include more types of deadlines and other QoS attributes.

From the works mentioned, parameters such as task priorities and deadlines appear to be important attributes for making scheduling decisions regarding tasks, messages, or data items. A measure of the overall value accrued of completed tasks is needed that can be used to compare and analyze the algorithms, protocols, and heuristics while incorporating all the QoS parameters that are relevant. The works mentioned above consider only a few of the QoS parameters that might be present in a distributed system. Other parameters (e.g., accuracy, precision, and security) that are QoS requirements and part of the users' requests must be included in the performance analysis. These and other QoS parameters that affect the overall value of requests satisfied are discussed in the FISC research.

The FISC research on the performance measure presented here builds on and extends a body of earlier work in this field. Some examples are mentioned here.

The ERDoS project [5] describes an objective function for optimizing the effectiveness of its QoS scheduling mechanisms in meeting clients' needs. This function reflects the benefit received by the user and a weight assigned to each user application. An approach where the requested QoS is taken into account when scheduling computational resources in a network is presented in [27]. The model proposed a benefit function that uses application deadlines and application priorities as metrics in maximizing the total benefit for the applications. The incorporation of multiple versions of a task, in addition to priorities and deadlines, in the objective function is described in [17]. The FISC performance measure presented in this paper serves a purpose similar to the ones in [5, 17, 27]. However, the research presented here provides a more detailed description of a measure using more parameters, so that it can be used to find the performance of schedules in the Quorum MSHN and BADD/AICE environments. Furthermore, the QoS input specification for ERDoS [34] accounts for only two specific security parameters (confidentiality and integrity), whereas the security component of the performance measure in this research can describe an arbitrarily complex set of security features.

The resource allocation model for QoS management proposed in [31] indicates multiple dimensions of QoS and multiple resources. In [31], some of the QoS attributes studied in this research are mentioned but not elaborated. The utility that is described in [31], is the same as the value accrued in a given time interval using a set of resources. The FISC research discusses QoS attributes in more detail and gives more QoS factors to consider. Work done in [21, 22] presents specific usage of the model presented in [31]. These use only a subset of QoS attributes that the FISC research describes, indicating that the FISC measure would be a generalized version of what they have used as the utility function.

The FISC measure is similar to the utility function described in [40] in that both measure system value. The utility function in [40] calculates the value of the resource management done in a system by summing the value of resources allocated and resources reserved. The resources reserved depend on the duration of a job, the deadline of a job, and the price of allocated time slots. In contrast, the

FISC measure uses priorities and other QoS measures to calculate the collective value of tasks that are completed.

The research in [29] describes a utility function that considers the throughput and the link congestion of the network and extends their analysis to QoS sensitive requests. The utility function described in [29] and the FISC measure both seek to achieve the optimum value of the requests satisfied. In this FISC measure paper, QoS attributes (e.g., deadlines, security) are considered in detail while in [29], the QoS factor is represented by the link congestion.

In the model proposed by [7], a function that indicates the utility due to QoS attained when a certain bandwidth is allocated to the user and the "willingness-to-pay of the user" factor is employed to calculate the net benefit. The FISC measure and the utility function in [7] are similar in that they calculate the overall value of resources allocated. While [7] considers only bandwidth, the FISC work presented in this paper discusses one way to combine different QoS attributes to result in a framework for determining the total value accrued from completing tasks in a given interval of time.

A security policy that allows a percentage of packets authenticated to vary with network load is described in [35]. This type of policy can be accommodated with the variant components included in the FISC security vector (see Subsection 3.4). Related work on network security policy specification languages can be found in [2, 3], and works in progress [6, 33]. While the FISC security vector contains a set of Boolean security policy statements, it does not specify a general-purpose language for these statements. A framework for quantifying the strength of a set of security mechanisms is described in [39], where high-level static security properties can be decomposed hierarchically. However, in [39] the approach cannot accommodate the measurement of how well an executed task meets the security requirements of its environment. Nor does [39] account for variant security policies or mechanisms.

The FISC measure assesses the effectiveness of a schedule in terms of the value accrued by tasks that are completed and the fraction of their QoS requirements satisfied. In [35], other types of attributes for the evaluation of different QoS and RMS services in distributed real-time systems are investigated. Some examples include survivability (fault-tolerance), openness (open architecture), and testability (easily testable and verifiable). These are beyond the scope of the FISC performance measure.

3. Example QoS Attributes

3.1. Priorities

Policy makers determine the number of priority levels available within a system and assign some semantic meaning to each priority level, such that relative importance of each level is qualitatively reflected (e.g., high, medium, and low). Each priority level will then be given a weight that can be calculated by a priority weight function and this weight may be a function of the situational mode (e.g. war or peace). It is assumed that the FISC measure is being used to compute the value of a subset of tasks successfully completed, during some time interval, from a set of t tasks that have been requested. Let the priority level (e.g., high, medium, or low) of task j ($0 \leq j < t$) be p_j, and let m be the situational mode. The priority weight function $\pi(p_j)$ calculates the weight of p_j given the situational mode m. The weight assigned to a priority level may be considered to be the maximum value possible for completing the corresponding task, if all of the task's specified QoS requirements are completely satisfied.

3.2. Versions

A task may exist in different versions, each with its own resource requirements. Because available resources will vary dynamically, it may not be possible to complete the most desired version of a task. When a user's first choice of a task version cannot be completed, a method for choosing an alternative version is needed. Having multiple versions of a task is related to the precision and accuracy parameters discussed in [34], in the sense that each version of a task may have different accuracy and precision.

For each version of a given task, a worth (preference) relative to the other versions will be indicated by the application developer, the user, and/or the policy makers. The worth may be a function of the situational mode. To allow worth to be quantified in an arbitrary format, the worths assigned to different versions of a task must be normalized so that there is a consistent scheme for evaluating worths across tasks and versions.

The worths are normalized as follows. Assume there are I_j versions for a given task j. Let v_{ij} be the i-th ($0 \leq i < I_j$) version of task j. Let $w_{ij}(m)$ be the worth the user assigns to i-th version of task j given m, the situational mode. Thus, the normalized worth (η_{ij}) of $w_{ij}(m)$ is given by

$$\eta_{ij} = \frac{w_{ij}(m)}{\left(\max_{0 \leq i < I_j} w_{ij}(m) \right)}. \qquad (1)$$

All worths for all versions of each task are normalized by the version with the largest worth. Therefore, the version with the largest worth of each task will have a normalized worth of 1 and the rest of the versions will have normalized worths that are less than 1.

3.3. Deadlines

Many tasks in typical heterogeneous computing environments have deadlines associated with them. Frequently, due to limited resources and the multiplicity of tasks sharing these resources, not every task can be completed by its deadline. Three types of deadlines will be considered for the i-th version of task j: earliest ($\underline{e}_{ij}^{d}$), soft ($\underline{s}_{ij}^{d}$), and firm ($\underline{f}_{ij}^{d}$). The deadline attributes discussed here are related to the timeliness parameter given in [34].

The earliest deadline (e_{ij}^{d}) is the time when the system is ready to complete the i-th version of task j. The soft deadline (s_{ij}^{d}) ([37]) is the time by which the i-th version of task j must complete to be of full value to the user. If a task is completed between the earliest deadline and the soft deadline, then the task will have its maximum value. A task that is completed after its firm deadline (f_{ij}^{d}) ([18, 37]) will have 0 value, because the task will be of no use after that deadline. These three deadlines are illustrated by example in Figure 1.

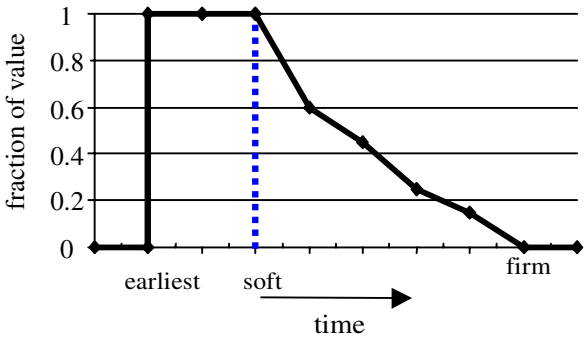

Figure 1: The deadline coefficient graph shows the variation in the value of a task with various deadlines.

Let $\underline{\tau}_{ij}$ be the time that the i-th version of task j actually completes. The deadline function $\underline{\delta}_{ij}$ assigns a fraction of the maximum value of the i-th version of task j based on m, τ_{ij}, e_{ij}^{d}, s_{ij}^{d}, and f_{ij}^{d}, where $0 \le \delta_{ij} \le 1$. The deadlines e_{ij}^{d}, s_{ij}^{d}, and f_{ij}^{d} may be the same for all versions of a certain task. A characteristic function δ_{ij}' is used to represent whether a version completes with a $\delta_{ij} > 0$: $\delta_{ij}' = 1$ if $\delta_{ij} > 0$, and $\delta_{ij}' = 0$ if $\delta_{ij} = 0$. If no version of task j is completed, $\delta_{ij} = 0$ and $\delta_{ij}' = 0$ for all versions of the task.

3.4. Security

User and task security requirements are met by "security services." Overall network security can be viewed as a multi-dimensional space, represented with a vector ($\underline{S}$) of security services. A Boolean statement, for a given situational mode, specifies the functional requirement for each component. Both resources and tasks may have multiple security components [13, 20].

The instantiation of a task either meets, or does not meet, each component's requirement. Let the characteristic function $\underline{\sigma_{ij}'}$ be used to represent required security attributes. If the minimum security requirements are not all met, there is no value accrued for executing v_{ij} and $\sigma_{ij}' = 0$. The characteristic function $\sigma_{ij}' = 1$ if the instantiated Boolean value of all components is true.

The desire to provide *adaptable* security motivates the inclusion of variant security components in the system [14]. Thus, security affects the performance measure when components are variable. Let $\underline{S}_{ij}$ be a subset of S for version i of task j, and $g_{ij,\kappa}$ be the fraction of κ satisfied in S_{ij}. Let $\underline{n}$ be the number of security components in S_{ij}. To quantify the effectiveness of the RMS in providing variant security, let security factor $\underline{\sigma_{ij}}$ be the sum of all $g_{ij,\kappa}$ divided by n as shown in

$$\sigma_{ij} = \frac{\left(\sum_{\kappa \in S_{ij}} g_{ij,\kappa} \right)}{n}. \qquad (2)$$

The overall security measure is defined as: $\sigma_{ij} \times \sigma_{ij}'$, where $0 \le \sigma_{ij} \times \sigma_{ij}' \le 1$.

3.5. Application Specific QoS

There is a multi-dimensional space of application- and domain-specific QoS attributes (e.g., jitter level, frame rate, and bit error rate) that can also be represented by a vector. There will be zero or more such components per task or data set. These components can be specified in the same way that security was represented. The FISC measure will use $\underline{\alpha_{ij}}$ to denote such QoS components and the characteristic function $\underline{\alpha_{ij}'}$ is used to represent the required application specific QoS attributes. If all minimum application specific QoS requirements are met $\alpha_{ij}' = 1$; if

they are not all met $\alpha_{ij}' = 0$. This QoS attribute is analogous to the security factor σ_{ij}.

3.6. Associates

There are many possible types of inter-task dependencies. For example, for the MSHN environment (Figure 2), consider a task whose only function is to generate data for other tasks (descendants). There may be an inter-related set of such tasks. If there is a descendant along a dependency path of tasks that generates an external output (as opposed to only generating an input for a subsequent task) and if this descendant completes its execution, then all of its predecessors will have a value.

The first task in a dependency path that generates an external output is called required associate of all of its predecessors. The variable ρ_{ij} will be used to represent whether a required associate of a given task completes. That is, if at least one required associate of a given task completes, then $\rho_{ij} = 1$, otherwise $\rho_{ij} = 0$.

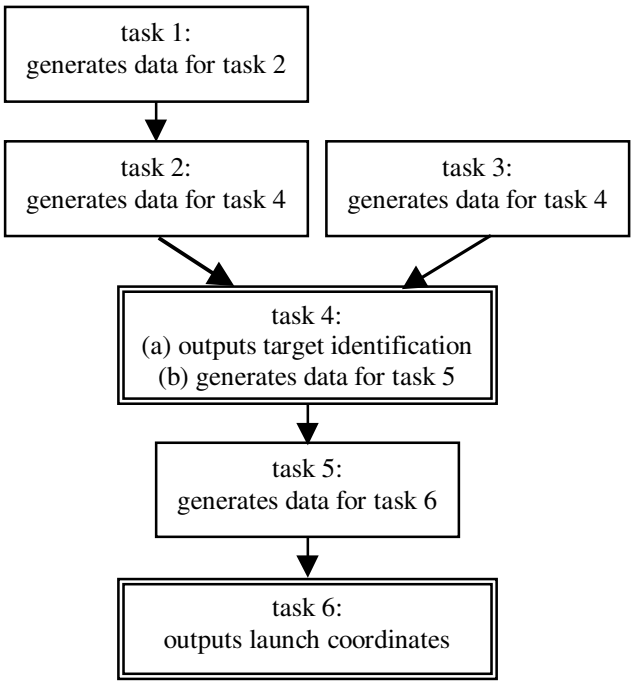

Figure 2: An example set of tasks that has dependency. Tasks 1, 2, 3, and 5 are tasks that only generate input data for other tasks. Tasks 4 and 6 give user output.

In Figure 2, task 1 is needed for task 2 to generate data for task 4. Task 4 needs data from tasks 2 and 3 to generate data for task 5 and output target identification information (an external output). Only if task 4 completes will tasks 1, 2, and 3 have value. Only if task 6 outputs the launch coordinates (an external output) will tasks 4 and 5 have value.

For the BADD/AICE environment, the objective is to schedule inter-machine data transfers to satisfy data requests. There may be multiple data requests for a given application to provide the needed input. Unless all such data requests are satisfied, the application cannot execute. If even one of the required data requests is not available, the value of all data requests that were satisfied is zero (i.e., $\rho_{ij} = 0$ for all data requests).

In Figure 3, data item 1, 2, and 3 are required for application 1 to execute. Therefore, if any one of these data requests is not satisfied (if only a subset of the data requests arrive by the firm deadline), there is no value for any of the satisfied data requests.

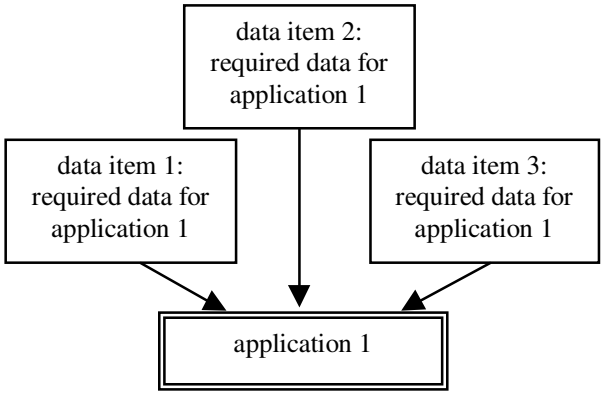

Figure 3: An example set of data items that have dependency. Data items 1, 2, and 3 are required input data for application 1.

4. Performance Measure

4.1. FISC Measure

In general, it is a difficult problem to determine whether a distributed system has delivered "good" service to a mixture of applications. For example, the applications may be compute-intensive and others interactive, perhaps having stringent security requirements. In this research, the collective value of QoS achieved is used for determining how "good" a system is.

A meaningful way to combine the QoS attributes previously discussed to capture the above is proposed in this section. This version of FISC will be called the averaged FISC and this method is similar in idea to the benefit function described in [13, 20]. To allow the comparison of one RMS, operating within one distributed system, to another RMS, operating in a different distributed system, the FISC measure can be normalized by a baseline, which depends on the tasks and underlying distributed system (Subsection 4.2). When the FISC measure is normalized by a baseline, the resulting function is called the FISC ratio.

The averaged FISC ratio is:

$$\frac{1}{\text{baseline}} \times \sum_{j=0}^{t-1} \left(\pi(p_j) \times \max_{0 \le i < I_j} \left(\rho_{ij} \times \delta'_{ij} \times \sigma'_{ij} \times \alpha'_{ij} \times \frac{[\eta_{ij} + \delta_{ij} + \sigma_{ij} + \alpha_{ij}]}{4} \right) \right) \quad (3)$$

Note that the value of the "max" never exceeds 1.

Equation 3 is only one way of representing the collective value accrued. Another version of FISC called the multiplied FISC (Equation 4) was introduced in earlier work [15].

$$\frac{1}{\text{baseline}} \times \sum_{j=0}^{t-1} \left(\pi(p_j) \times \max_{0 \le i < I_j} \left(\eta_{ij} \times \rho_{ij} \times \delta_{ij} \times \delta'_{ij} \times \sigma_{ij} \times \sigma'_{ij} \times \alpha_{ij} \times \alpha'_{ij} \right) \right) \quad (4)$$

4.2. Baseline

The purpose of the baseline in the FISC ratio is to determine how an RMS performs compared to another RMS in a different environment. For example, if one RMS performs well in an environment and another RMS gets good results in another environment, then the comparison only using the FISC measure would not make sense. Because the environments are different, how well a RMS performed should be how much better it performed than the baseline for its environment. If the RMS cannot perform much better than this baseline, then a naive algorithm for resource assignment would perform almost as well as the RMS. The baseline for a given environment is calculated by using the same set of tasks with same attribute requirements. The baseline builds upon and extends the example given by [38]. The algorithm used to compute the baseline uses the concept of perfect completion. A task is said to achieve perfect completion if there exist available resources, to which it can be assigned, that would allow it to complete with $\eta_{ij} = \delta_{ij} = \sigma_{ij} = \alpha_{ij} = 100\%$ and $\rho = 1$. This means that in given situations (i.e., resource availability, situational mode) tasks with the most preferred version, all security services and application specific QoS are satisfied 100%, a required associate that completes, and completion time before the soft deadline are considered.

A simple algorithm, which assumes knowledge of the expected resources needed by a task to complete, can be used to obtain a baseline. For the results of the obtained baseline to be reproducible within a certain tolerance, an ordering of the tasks is needed.

```
all given tasks are ordered by priority, deadline,
and expected execution time;
if all are equal, order is random;

for each task{
        if a task can get η_ij = δ_ij = α_ij = σ_ij =
                100% and ρ_ij = 1
        schedule at soonest possible time
                add π(p_j, m)
                update status of resources
        else
                no value added
                no resources consumed
}
```

Figure 4: Baseline algorithm.

The algorithm to obtain a baseline is shown in Figure 4 and proceeds as follows. First, it assigns an ordering to the tasks according to their priorities (highest first), deadlines (soonest first), and expected execution times (shortest first) where the above criteria are considered in the aforementioned order. For the tasks with the same priority level, the deadline would be used as a tiebreaker. If tasks have same priority level and deadline, the expected execution time would serve as a tiebreaker. Only if tasks have the same priority, deadline, and expected execution time would the ordering be random. Alternatively, additional characteristics of the task could be used for finer ordering. In other problem domains, other parameters could be more appropriate for ordering the tasks.

After the ordering, the algorithm determines whether the first task (according to the ordering) can be expected to

achieve perfect completion using the available resources. If so, it computes the expected availability of resources after that task has completed, under the assumption that the task uses the first such available resources. It also adds the weighted priority of this task to the baseline, which was initialized to 0. If a task cannot achieve perfect completion, nothing is added to the baseline and the task is not considered again. The same process is repeated for each task, considering them according to the ordering.

4.3. Averaged Versus Multiplied FISC

Both versions of the FISC measure mentioned in Subsection 4.1 can be used to calculate the collective value of the tasks completed. The averaged FISC (Equation 3) captures the intuition that when a service attribute is added to the measure, the worth of the service added should not lower the value of the task completed below the percentage satisfied of the least satisfied attribute. For example, assume there is a task j completed with a version that is worth 50%, was completed before the soft deadline (100%), and $\pi(p_j)$ is 1. Initially the completed task's value would be 0.75 by the averaged FISC and 0.5 by multiplied FISC (Equation 4). Then assume security service was added to the task and the overall security was 50% satisfied. If the FISC value was calculated by the multiplied FISC measure, the task's value would be $0.5 \times 1 \times 0.5 = 0.25$. The value of the task has decreased below the security services satisfied (50%) and the worth (50%) of the version used. However, if the FISC value was calculated by the averaged FISC, the task's value would be $(0.5 + 1 + 0.5)/3 = 0.67$, which is not lower than the security service satisfied (50%) and the worth (50%) of version used.

In addition, if a service is included and the service is satisfied to the fullest (100%), multiplied FISC would give the same value as previously calculated, but the averaged FISC would give a higher value. The intuition is that if a task is given more services and if those are 100% satisfied, then the overall value for that task should increase.

4.4. Resource Constraint

In addition to an optimization criterion such as the FISC measure, constraints are required to define any optimization problem. There is a limited amount of resources so there is a constraint on resource availability. Therefore, in any time instant, the total amount used of a particular resource cannot exceed the total available resource at that time instant. Assume that there are Ξ number of resources in the system. Let $R_{rj}(\Delta)$ be the amount of resource r ($0 \le r < \Xi$) used for task j ($0 \le j < t$) during the time interval Δ and let $U_r(\Delta)$ be the total resource r that is available during time interval Δ. The sum of all $R_{rj}(\Delta)$ cannot exceed $U_r(\Delta)$, as indicated in Equation 5.

$$\sum_{j=0}^{t-1} R_{rj}(\Delta) \le U_r(\Delta) \text{ for resources } r = 0 \ldots \Xi - 1 \quad (5)$$

4.5. FISC Ratio with Multiple Copies of Versions

It is possible that multiple versions of the same task or multiple copies of the same version can be attempted for fault tolerance or for maximizing the speed of the process. Assume that Γ_{ij} copies of version i of task j are considered. Then the averaged FISC measure (Equation 3) can be extended to include $v_{ij\gamma}$, where i is the version, j is the task, and γ is the copy number ($0 \le \gamma < \Gamma_{ij}$).

$$\frac{1}{\text{baseline}} \times \sum_{j=0}^{t-1} \left(\pi(p_j) \times \max_{\substack{0 \le i < I_j \\ 0 \le \gamma < \Gamma_{ij}}} \left(\rho_{ij\gamma} \times \delta'_{ij\gamma} \times \sigma'_{ij\gamma} \times \alpha'_{ij\gamma} \times \frac{\left[\eta_{ij\gamma} + \delta_{ij\gamma} + \sigma_{ij\gamma} + \alpha_{ij\gamma} \right]}{4} \right) \right) \quad (6)$$

4.6. Coefficients in FISC Measure

There can be coefficients for each attribute mentioned in this research. The coefficients can be incorporated into the FISC measure to indicate the relative importance of one attribute to another.

Let c_η, c_δ, c_σ, and c_α be the coefficients of η (worth), δ (deadline met), σ (security services satisfied), and α (application specific QoS satisfied) factors, respectively.

If the multiplied FISC measure is used, the coefficients used would not be able to indicate the relative importance. Because,

$$\sum_{j=0}^{t-1} \left(\pi(p_j) \times \max_{0 \le i < I_j} \left(c_\eta \times \eta_{ij} \times \rho_{ij} \times c_\delta \times \delta_{ij} \times \delta'_{ij} \times c_\sigma \times \sigma_{ij} \times \sigma'_{ij} \times c_\alpha \times \alpha_{ij} \times \alpha'_{ij} \right) \right) \quad (7)$$

would be equal to

$$c_\eta \times c_\delta \times c_\sigma \times c_\alpha \times \sum_{j=0}^{t-1} \left(\pi(p_j) \times \max_{0 \le i < I_j} \left(\eta_{ij} \times \rho_{ij} \times \delta_{ij} \times \delta'_{ij} \times \sigma_{ij} \times \sigma'_{ij} \times \alpha_{ij} \times \alpha'_{ij} \right) \right) \quad (8)$$

This would mean that the coefficients (c_η, c_δ, c_σ, and c_α) are no longer the coefficients for each component but a multiplier of the resulting overall value.

The averaged FISC measure can incorporate these coefficients in a meaningful way. First consider:

$$\sum_{j=0}^{t-1} \left(\pi(p_j) \times \max_{0 \le i < I_j} \left(\rho_{ij} \times \delta'_{ij\gamma} \times \sigma'_{ij\gamma} \times \alpha'_{ij\gamma} \times \left[c_\eta \times \eta_{ij} + c_\delta \times \delta_{ij\gamma} + c_\sigma \times \sigma_{ij\gamma} + c_\alpha \times \alpha_{ij\gamma} \right] \right) \right) \quad (9)$$

For the value of the "max" term to never exceed 1, the measure will be divided by the sum of the coefficients (Equation 10). This will be the averaged FISC with coefficients. By dividing the measure by the sum of the coefficients, the "max" term of the FISC measure will be 1 when all attributes are 100 percent satisfied and less than 1 when a certain task gets degraded QoS.

$$\sum_{j=0}^{t-1} \left(\pi(p_j) \times \max_{0 \le i < I_j} \left(\rho_{ij\gamma} \times \delta'_{ij\gamma} \times \sigma'_{ij\gamma} \times \alpha'_{ij\gamma} \times \frac{\left[c_\eta \times \eta_{ij\gamma} + c_\delta \times \delta_{ij\gamma} + c_\sigma \times \sigma_{ij\gamma} + c_\alpha \times \alpha_{ij\gamma} \right]}{c_\eta + c_\delta + c_\sigma + c_\alpha} \right) \right) \quad (10)$$

4.7. Priority Levels within Classes

In some environments, it may be the case that all higher priority level tasks must be attempted for execution and completion first, before any of the lower priority level tasks can be considered. In this scheme, a higher priority level task is worth more than any number of lower priority level tasks (i.e., highest priority level task is worth an infinite number of lower priority level tasks). To represent this, classes of priority levels will be needed.

When classes are used, each task will have a priority level, and priority levels will have relative weightings as before. Tasks will not have classes but the priority level that the task was assigned will correspond to a class predetermined by the system administrator or the policy maker. Each class will consist of one or more priority levels. There can be several classes. The number of priority levels assigned to a class can vary from one class to another.

As shown in the example in Figure 5, there could be L priority levels and C classes. Priority 1 is more important than priority 2 and class 1 is more important than class 2. Any number of priority levels can be in one class. To compare two schedules, first consider only class 1 tasks. The scheduler with the higher class 1 FISC value is considered best. Only if there is a tie at class 1, is the FISC value accrued by class 2 tasks considered. In general, only when the FISC values of classes 1 to k are the same for more than one schedule are the FISC values of class $k + 1$ used in the comparison.

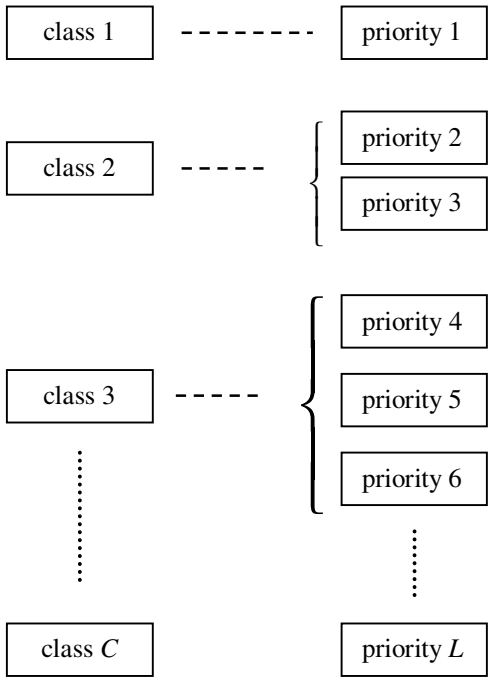

Figure 5: Example of priority levels within classes.

4.8. Generalization of FISC Ratio

The previous subsections describe a particular example of the FISC measure. It can be generalized such that the numerator is any function of $\pi(p_j)$, η_{ij}, ρ_{ij}, δ_{ij}, σ_{ij}, and α_{ij} (or other factors), and each of these primary factors can be any function of secondary factors (e.g., primary factor σ_{ij} includes an average of $g_{ij}.c$ secondary factors in the security context described in Subsection 3.4). Let P_y be a primary factor where there can be u number of primary factors ($0 \le y < u$) and s_e be a secondary factor where there can be v_y

number of secondary factors ($0 \leq e < v_y$). The generalization of FISC measure can be represented as

$$FISC \text{ measure} = f(P_0, P_1, \ldots, P_{u-1}) \text{ and} \quad (11)$$

$$P_y = f_y(s_0, s_1, \ldots, s_{v_y - 1}), \quad (12)$$

where each s_e is a secondary factor for P_y. Linear or nonlinear combinations of the factors, depending on the importance of the factor considered in a given environment, may be included in all the functions of primary and secondary factors.

The baseline algorithm described is one method of normalizing the FISC measure. Other methods for normalizing could be incorporated to compare the performance of different RMSs in a given environment.

5. Game Theory

5.1. Basis of Game Theory

Game theory is a tool that helps people understand the phenomena when decision-makers interact [30]. It is used in this research to determine what kind of characteristics the FISC measure will have.

A game is a description of strategic interaction that includes the constraints on the actions that the players can take and the players' interests, but does not specify the actions that the players do take. A player may be interpreted as an individual or a group of individuals making a decision. A solution is a systematic description of the outcomes that may emerge in the family of games. Game theory suggests reasonable solutions for classes of games and examines their properties [30].

In a heterogeneous environment, it may be the case that the total demand placed on the system may exceed the total resources available. Each request or task in the system will then compete for the resources available at the given interval of time to try to satisfy their QoS requirements and this becomes a game situation. The requests or tasks will be the players in the game theory context, where the scheduler will then try to choose a solution according to the preferences and other indicated requirements of the tasks.

5.2. Evaluating Various Game Theoretic Solutions using the FISC Measure

Assume there are two tasks in the system and one resource. Further, assume: (1) that these two tasks have the same priority level, and the weight of that priority level is one; (2) that the soft deadline and firm deadline for each task is the same (referred to as just the deadline); (3) the deadlines of these two tasks are the same; (4) each task has only one version; (5) the system has only one QoS attribute to satisfy (e.g., variable security); and (6) a resource unit is the total amount of given resource available during a time unit (given) [19]. Both tasks are assumed to finish by the deadline. Therefore, for both tasks $\delta_{ij}' = 1$, $\delta_{ij} = 1$, $\pi(p_j, m) = 1$, and $\eta_{ij} = 1$. Assume that the one QoS attribute to satisfy is one variable security attribute and that minimum security requirement is met ($\sigma_{ij}' = 1$). For the variable security, assume that if a task receives only 0.x of its required resources, its value will be only 0.x.

Assume task 1 needs 30 units of resource to complete with every requirement fully satisfied and task 2 needs 70 resource units to be satisfied in the same way. If the total resources available correspond to 60 units, then the total needs of task 1 and task 2 exceeds total available resource. Because resources are limited, tasks (individuals) have to compete or bargain for the resources. This is a game situation. Out of many schemes of game theory, this situation is similar to the concept of the Nash bargaining scheme [30]. The solutions that result from the Nash bargaining scheme will be used in the calculation of the corresponding FISC value. The Nash bargaining solutions are said to be feasible, optimal, and fair. In addition, the Nash solutions that the scheme produces are said to be consistent [30].

The bargaining scheme gives various solutions to problems like resource division. For the example in the previous paragraph, where there are only 60 units of resources and task 1 needs 30 units to complete satisfying all requirement 100% and task 2 needs 70 units to complete in the same way. Some of the solutions are: (1) equal division; (2) division proportional to resource requirements; (3) division proportional to what each task would get if it was the only task in the system; and (4) a split where the undisputed resource is first given to the task that needs more resource and the rest is equally divided. All these division schemes are examples of Nash bargaining solutions. Using the example mentioned, for the equal division solution, each task gets 30 units of resources. For the division proportional to resource requirements solution, the resource division will result in a 30 to 70 split, i.e., task 1 gets 18 units and task 2 gets 42 units of resources. For the division proportional to what each task would get if it were the only task in the system solution, the resource division will result in a 30 to 60 split, i.e., task 1 gets 20 units and task 2 gets 40 units. For the undisputed division solution, the undisputed amount of resources which are 30 units (task 1 only needs 30 units of resources) is first given to the task that needs more resource units than the other task, the rest of the resources are divided in half. The undisputed resource division gives a 15 units and 45 units split.

This paragraph gives the results of FISC value calculation using the averaged equation. For this example, the resource requirements (70 units for task 1 and 30 units for task 2), the resource constraint (60 units), and the various splits are considered. When calculating the total value accrued, the averaged FISC is used. If the resource is equally divided, task 1 will receive a FISC value of 30/70 and task 2 will receive 30/30. The total FISC value accrued would be (1+1+3/7)/3 + (1+1+1)/3 = 1.81. If the resource is divided proportionally into a 30:70 split, task 1 will receive a FISC value of (1+1+42/70)/3 and task 2 would receive (1+1+18/30)/3. The total FISC value accrued for this type of allocation would be 1.74. In another split (a 30:60 split of 60 units), task 1 would get a FISC value of (1+1+40/70)/3 and task 2 would get (1+1+20/30)/3. The total FISC value accrued in this scenario is 1.75. When the resource is divided using the last split method, then task 1 would get a FISC value of (1+1+45/70)/3 and task 2 would get (1+1+15/30)/3. The total FISC value accrued would be 1.71. In this context, equally dividing the resource gives the best total FISC value achieved. All of these results are shown in Table 1.

Table 1: Example value calculation using different possible resource division solutions and the averaged FISC measure.

	even split	30:70 split	30:60 split	undisputed split
Task 1 (70)	0.81	0.87	0.86	0.88
Task 2 (30)	1	0.87	0.89	0.83
Total	1.81	1.74	1.75	1.71

Calculating the actual collective value of tasks completed during an interval of time is more complicated than shown here. There are more attributes to consider and attributes may have relative importance weightings. In this example, the same priority, arrival time, and deadline are assumed, which will not always be valid. In addition, resources may be reused depending on how tasks are scheduled and completed. It may be the case that if a task gets more resource than it needs, the task may end early making it possible for other tasks to reuse the resource that the task was using. Therefore, actual resource allocation is more complicated than the example shown in this subsection.

6. Summary

In some environments, distributed heterogeneous computing system may be over-subscribed, where the total demand placed on system resources by the tasks, for a given interval of time, exceeds the resources available. In such environments, users' tasks are allocated resources to simultaneously satisfy, to varying degrees, the tasks' different, and possibly conflicting, QoS requirements. When tasks are allocated resources, some tasks will receive degraded QoS or no service at all. The FISC measure provides a way to quantify the value of the performance received by a set of applications in a distributed system. With the FISC measure, the effectiveness of the mapping of a collection of requests to resources done by a scheduler can be evaluated in terms of value as perceived by the user, policy maker, administrator, or system. In addition, the FISC measure may be used in a simulation mode to analyze the impact of proposed changes to the distributed system. The kind of result that the FISC measure would give was explored using some game theoretic solutions.

The FISC performance measure presented here will help the distributed computing community in the implementation of resource management systems and the analysis and comparison of such systems. Furthermore, the FISC measure may be used as a critical part of a scheduling heuristic's objective function.

Additional issues that may be considered in future research include: using a non-linear combination of task values to compute the overall measure; the use of negative fractions in the deadline function in case where catastrophic results from a missed deadline; how to incorporate FISC measure in a scheduling heuristic; investigating other factors that are important in calculating the value of task mapped to the user; and investigating variations in the factors already considered.

Acknowledgments: The authors thank Bob Beaton, Gary Koob, Joe Rockmore, Michael Jurczyk, I-Jeng Wang, Steve Jones, John Kresho, Edwin K. P. Chong, Rudolf Eigenmann, Neil Rowe, Carl Kesselman, Noah Beck, Tracy Braun, Shoukat Ali, Surjamukhi Chatterjea, Amit Naik, and Pranav Dharwasdkar for their valuable comments and suggestions.

References

[1] Agile Information Control Environment Proposers Information Package, BAA 98-26, Sep. 1998, http://web-ext2.darpa.mil/iso/aice/aicepip. htm.

[2] L. Badger, D. F. Stern, D. L. Sherman, K. M. Walker, and S. A. Haghighat, "Practical domain and type enforcement for Unix," *1995 IEEE Symposium on Security and Privacy*, May 1995, pp. 66-77.

[3] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," *1996 IEEE Symposium on Security and Privacy*, May 1996, pp. 164-173.

[4] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, accepted and scheduled to appear 2001.

[5] S. Chatterjee, B. Sabata, and J. Sydir, *ERDoS QoS Architecture*, Technical Report, SRI, Menlo Park, CA, ITAD-1667-TR-98-075, May 1998.

[6] M. Condell, C. Lynn, and J. Zao, "Security policy specification language," *INTERNET-DRAFT*, Network Working Group, Oct. 1998, ftp://ftp.ietf.org/ internet-drafts/draft-ietf-ipsec-spsl-00.txt.

[7] C. Coutcoubetis, G. D. Stamoulis, C. Manolakis, and F. P. Kelly, "An intelligent agent for optimizing QoS-for-money in priced ABR connections," *IEEE Transactions on Telecommunications Systems*, Special Issue on Network Economics, to appear, (preprint at http://www.ics.forth.gr /ICS/acti/ netgroup/publications/abr_agent.html).

[8] DARPA, Battlefield Awareness and Data Dis-semination, April 1999, www.darpa.mil/iso/ badd/.

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

[10] D. Ferrari, ed., *Computer Systems Performance Evaluation*, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[11] I. Foster and C. Kesselman, eds., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco, CA, 1999.

[12] D. A. Hensgen, T. Kidd, D. St. John, M. C. Schnaidt, H. J. Siegel, T. D. Braun, M. Maheswaran, S. Ali, J. Kim, C. Irvine, T. Levin, R. F. Freund, M. Kussow, M. Godfrey A. Duman, P. Carff, S. Kidd, V. Prasanna, P. Bhat, and A. Alhusaini, "An overview of MSHN: The Management System for Heterogeneous Networks," *8th Heterogeneous Computing Workshop* (*HCW '99*), Apr. 1999, pp. 184-198.

[13] C. Irvine and T. Levin, "Toward quality of security service in a resource management system benefit function," *9th IEEE Heterogeneous Computing Workshop* (*HCW 2000*), May 2000, pp. 133-139.

[14] C. E. Irvine and T. Levin, "Toward a taxonomy and costing method for security services," *15th Annual Computer Security Applications Conference*, Dec. 2000, pp 183-188.

[15] J.–K. Kim, D. Hensgen, T. Kidd, H. J. Siegel, D. St. John, C. Irvine, T. Levin, N. W. Porter, V. K. Prasanna, and R. F. Freund, "A QoS performance measure framework for distributed heterogeneous networks," *8th Euromicro Workshop on Parallel and Distributed Processing*, Jan. 2000, pp. 18-27.

[16] J. H. Kim and H. S. Shin, "Optimistic priority-based concurrency control protocol for firm real-time database systems," *Information & Software Technology*, Vol. 36, No. 12, Dec. 1994, pp. 707-715.

[17] J. P. Kresho, *Quality Network Load Information Improves Performance of Adaptive Applications*, Thesis, Department of Computer Science, Naval Postgraduate School, Monterey, CA, Sep. 1997 (D. A. Hensgen, advisor), 164 pp.

[18] C. G. Lee, Y. K. Kim, S. H. Son, S. L. Min, and C. S. Kim, "Efficiently supporting hard/soft deadline transactions in real-time database systems," *3rd International Workshop on Real-Time Computing Systems and Applications*, Oct./ Nov. 1996, pp. 74-80.

[19] T. Levin and C. Irvine, *Approach to Characterizing Resource Usage in a Resource Management Benefit Function*, Technical Report, NPS-CS-99-005, Naval Postgraduate School, Monterey, CA, June 1999, 11 pp.

[20] T. Levin and C. Irvine, *Quality of Security Service in a Resource Management System Benefit Function*, Technical Report, NPS-CS-00-02, Naval Postgraduate School, Monterey, CA, Nov. 1999, 8 pp.

[21] C. Lee, J. Lehoczky, R. Rajkumar, and D. P. Siewiorek, "On quality of service optimization with discrete QoS options," *5th IEEE Real-Time Technology and Applications Symposium*, June 1999, pp. 276-286.

[22] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen, "A scalable solution to the multi-resource QoS problem," *20th IEEE Real-Time Systems Symposium*, Dec. 1999, pp. 315-326.

[23] K. J. Liszka, J. K. Antonio, and H. J. Siegel, "Problems with comparing interconnection networks: Is an alligator better than an armadillo?," *IEEE Concurrency*, Vol. 5, No. 4, Oct.-Dec. 1997, pp.18-28.

[24] J. P. Li and M. W. Mutka, "Priority based real-time communication for large scale wormhole networks," *8th International Parallel Processing Symposium*, Apr. 1994, pp. 433-438.

[25] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *Journal of Parallel and Distributed Computing*, Vol. 59, No. 2, Nov. 1999, pp.107-121.

[26] M. Maheswaran, T. D. Braun, and H. J. Siegel, "Heterogeneous distributed computing," in *Encyclopedia of Electrical and Electronics Engineering*, J. G. Webster, ed., John Wiley, New York, NY, 1999, Vol. 8, pp. 679-690.

[27] M. Maheswaran, "Quality of service driven resource management algorithms for network computing," *1999 International Conference on Parallel and Distributed Processing Technologies and Applications* (*PDPTA'99*), June/July 1999, pp. 1090-1096.

[28] D. C. Marinescu, "A protocol for multiple access communication with real-time delivery constraints," *IEEE INFOCOM '90*, June 1990, pp. 1119-1126.

[29] P. Marbach, "Pricing priority classes in a differentiated services network," *37th Annual Allerton Conference on Communication, Control, and Computing*, Sep. 1999, pp. 1075-1084.

[30] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*, MIT Press, Cambridge, MA, 1998.

[31] R. Rajkumar, C. Lee, J. P. Lehoczky, and D. P. Siewiorek, "A resource allocation model for QoS management," *IEEE Symposium on Real-Time Systems*, Dec. 1997, pp. 289-307.
[32] A. J. Rockmore, *BADD Functional Description*, Internal DARPA Memo, Feb. 1996.
[33] T. Ryutov and C. Neuman, "Access control framework for distributed applications," *INTERNET-DRAFT*, CAT Working Group, November 1998, ftp://ftp.ietf.org/internet-drafts/draft-ietf-cat-acc-cntrl-frmw-01.txt.
[34] B. Sabata, S. Chatterjee, M. Davis, J. Sydir, and T. Lawrence, "Taxonomy for QoS specifications," *3rd International Workshop on Object-Oriented Real-Time Dependable Systems* (*WORDS '97*), Feb. 1997, pp. 100-107.
[35] P. A. Schneck and K. Schwan, "Dynamic authentication for high-performance networked applications," *6th International Workshop on Quality of Service* (*IWQoS '98*), May 1998, pp. 127-136.
[35] B. Shirazi, L. Welch, B. Ravindran, C. Cavanaugh, B. Yanamula, R. Brucks, and E. Huh, "DynBench: A dynamic benchmark suite for distributed real-time systems," *Parallel and Distributed Processing: 11th IPPS/SPDP '99*, J. Rolim et al., eds., Springer, Berlin, Apr. 1999, pp. 1335-1349.
[36] L. J. Siegel, H. J. Siegel, and P. H. Swain, "Performance measures for evaluating algorithms for SIMD machines," *IEEE Transactions on Software Engineering*, Vol. SE-8, No. 4, July 1982, pp. 319-331.
[37] J. A. Stankovic, M. Supri, K. Ramamritham, and G. C. Buttazzo, *Deadline Scheduling for Real-Time Systems*, Kluwer Academic Publishers, Norwell MA, 1998, pp. 13-22.
[38] M. D. Theys, M. Tan, N. B. Beck, H. J. Siegel, and M. Jurczyk, "A mathematical model and scheduling heuristics for satisfying prioritized data requests in an oversubscribed communication network," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 11, No. 9, Sep. 2000, pp. 969-988.
[39] C. Wang and W. A. Wulf, "A framework for security measurement," *The National Information Systems Security Conference*, Oct. 1997, pp. 522-533.
[40] W. E. Walsh, M. P. Wellman, P. R. Wurman, and J. K. Mackie-Mason, "Some economics of market-based distributed scheduling," *18th International Conference on Distributed Computer Systems*, May 1998, pp. 612-621.

Biographies

Jong-Kook Kim is currently pursuing a Ph.D. degree from the School of Electrical and Computer Engineering at Purdue University, where he has been working as a research assistant since August 1998. Mr. Kim received his B.S.E.E. from Korea University and his M.S.E.E. degree from Purdue University. He is a student member of the IEEE. He served in the ROK Army working with the US military on the Theater Automated Command and Control Information Management System and received the US Army Commendation medal. His research interests include heterogeneous computing, evolutionary heuristics, computer architecture, and parallel computing.

Taylor Kidd is currently working as a Senior Software Engineer for OpenTV, the leading middleware provider worldwide for interactive services on digital TV receivers. Dr. Kidd's present responsibilities include representing and coordinating OpenTV's technical activities in US and European standards bodies. Before joining OpenTV, Dr. Kidd was an Associate Professor of Computer Science at the Naval Postgraduate School in Monterey, CA. Dr. Kidd obtained his Ph.D. in Electrical Engineering from UCSD in 1991.

Howard Jay Siegel is a Professor in the School of Electrical and Computer Engineering at Purdue University, where he has been on the faculty since 1976. Beginning August 2001, he will hold the endowed chair position of Abell Distinguished Professor of Electrical and Computer Engineering at Colorado State University. Prof. Siegel received a B.S. degree in electrical engineering and a B.S. degree in management from the Massachusetts Institute of Technology (MIT), and the M.A, M.S.E, and Ph.D. degrees from the Department of Electrical Engineering and Computer Science at Princeton University. He is a Fellow of the IEEE and a Fellow of the ACM. Prof. Siegel has co-authored over 280 technical papers, has co-edited seven volumes, and wrote the book *Interconnection Networks for Large-Scale Parallel Processing*. He was a Coeditor-in-Chief of the *Journal of Parallel and Distributed Computing*, and was on the Editorial Boards of the *IEEE Transactions on Parallel and Distributed Systems* and the *IEEE Transactions on Computers*. Prof. Siegel's research interests include heterogeneous parallel and distributed computing, communication networks, parallel algorithms, interconnection networks, and reconfigurable parallel computer systems. Agencies that have supported his research include the Air Force Office of Scientific Research, Army Research Office, Ballistic Missile Defense Agency, DARPA, Defense Mapping Agency, IBM, NASA, Naval Ocean Systems Center, Naval Research Laboratory, National Science Foundation, NRaD, Office of Naval Research, and Rome Laboratory. He was Program Chair/Co-Chair of three international conferences, General Chair/Co-Chair of four international conferences, and Chair/Co-Chair of four workshops. He is a member of the Eta Kappa Nu electrical engineering honorary society and the Sigma Xi science honorary society. He is an international keynote speaker and tutorial lecturer, as well as a consultant for government and industry.

Cynthia E. Irvine is an Assistant Professor at the Naval Postgraduate School, Monterey, California, USA. She is also the Director of the Naval Postgraduate School Center for INFOSEC Studies and Research. She holds a B.A. from Rice University and a Ph.D. from Case Western Reserve University. Her current research involves the characterization of security within quality of service frameworks, dynamic costing and parameterization of security services in heterogeneous environments, architectural issues for modern high assurance networked systems including the effective use of hardware security features, enhancement of protection mechanisms in operating systems, and tools for teaching system security concepts. She has received several research awards from the Naval Postgraduate School. She has served on the organizing committee of several international meetings on computer security and computer security education. Dr. Irvine is

a Senior Member of the IEEE and a member of the Executive Committee of the IEEE Technical Committee on Security and Privacy.

Timothy Levin is a Senior Research Associate at the Naval Postgraduate School. Mr. Levin has spent over 15 years working in the design, development, evaluation, and verification of secure computer systems. His current research interests include management and quantification of security in heterogeneous networks, development of costing frameworks and scheduling algorithms for the dynamic selection of QoS security mechanisms, and the application of formal methods to secure computer systems.

Debra Hensgen is a member of the Research and Evaluation Team at OpenTV in Mountain View, California. OpenTV produces middleware for set-top boxes in support of interactive television. She received her Ph.D. in the area of Distributed Operating Systems from the University of Kentucky. Prior to moving to private industry, as an Associate Professor in the systems area, she worked with students and colleagues to design and develop tools and systems for resource management, network re-routing algorithms and systems that preserve quality of service guarantees, and visualization tools for performance debugging of parallel and distributed systems. She has published numerous papers concerning her contributions to the Concurra toolkit for automatically generating safe, efficient concurrent code, the Graze parallel processing performance debugger, the SAAM path information base, and the SmartNet and MSHN Resource Management Systems.

David St. John is technical director and head software architect for Marak Services, an Internet-based weather service. Previously, Mr. St. John was head of staff at the Heterogeneous Network & Computing Laboratory at the Naval Postgraduate School, Monterey. He has over eight years experience in object-oriented software development primarily for process control, sensor data collection, and Internet transaction processing systems. He is a member of IEEE and IEEE Computer Society. He was a recipient of the Chancellor's Fellowship and an M.S. degree in Engineering from the University of California, Irvine in 1994.

Viktor K. Prasanna (V. K. Prasanna Kumar) received his BS in Electronics Engineering from the Bangalore University and his M.S. from the School of Automation, Indian Institute of Science. He obtained his Ph.D. in Computer Science from the Pennsylvania State University in 1983. Currently, he is a Professor in the Department of Electrical Engineering as well as in the Department of Computer Science at the University of Southern California, Los Angeles. He is also an associate member of the Center for Applied Mathematical Sciences (CAMS) at USC. He served as the Division Director for the Computer Engineering Division during 1994-98. His research interests include parallel computation, computer architecture, VLSI computations, and high performance computing for signal and image processing and vision. Dr. Prasanna has published extensively and consulted for industries in the above areas. He has served on the organizing committees of several international meetings in VLSI computations, parallel computation, and high performance computing. He is the Steering Co-chair of the International Parallel and Distributed Processing Symposium [merged IEEE International Parallel Processing Symposium (IPPS) and the Symposium on Parallel and Distributed Processing (SPDP)] and is the Steering Chair of the International Conference on High Performance Computing (HiPC). He serves on the editorial boards of the Journal of Parallel and Distributed Computing, IEEE Transactions on Computers, and the IEEE Transactions on Parallel and Distributed Systems. He was the founding Chair of the IEEE Computer Society Technical Committee on Parallel Processing. He is a Fellow of the IEEE.

Richard F. Freund is a founder and CEO of NOEMIX, a San Diego based startup to commercialize distributed computing technology. Dr. Freund is also one of the early pioneers in the field of distributed computing, in which he has written or co-authored a number of papers. In addition, he is a founder of the Heterogeneous Computing Workshop, held each year in conjunction with IPPS/SPDP. Dr. Freund won a Meritorious Civilian Service Award during his former career as a government scientist.