A Flexible Multi-dimensional QoS Performance Measure Framework for Distributed Heterogeneous Systems

Jong-Kook Kim[1], *Debra A. Hensgen*[2],
Taylor Kidd[2], *Howard Jay Siegel*[3], *David St. John*[2], *Cynthia Irvine*[2],
Tim Levin[4], *N. Wayne Porter*[2], *Viktor K. Prasanna*[5], and *Richard F. Freund*[6]

[1]Purdue University
Electrical and Computer Engineering School
W. Lafayette, IN 47907-1285 USA
jongkook@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]Colorado State University
Department of Electrical and Computer Engineering
Department of Computer Science
Ft. Collins, CO 80523-1373 USA
HJ@ColoState.edu

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

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

[6]GridIQ
13833 Adrian St
Poway, CA 92064-3453, USA
rffreund@gridiq.com

July 2003

Submitted to Cluster Computing
Corresponding author: Jong-Kook Kim

Abstract

When users' tasks in a distributed heterogeneous computing environment (e.g., cluster of heterogeneous computers) are allocated resources, the total demand placed on some system resources by the tasks, for a given interval of time, may exceed the availability of those resources. In such a case, some tasks may receive degraded service or be dropped from the system. One part of a measure to quantify the success of a resource management system (RMS) in such a distributed environment is the collective value of the tasks completed during an interval of time, as perceived by the user, application, or policy maker. The Flexible Integrated System Capability (FISC) measure presented here is a measure for quantifying this collective value. The FISC measure is a flexible multi-dimensional measure, and may include priorities, versions of a task or data, deadlines, situational mode, security, application- and domain-specific QoS, and task dependencies. For an environment where it is important to investigate how well data communication requests are satisfied, the data communication request satisfied can be the basis of the FISC measure instead of tasks completed.

Keywords: cluster computing; distributed computing; heterogeneous computing; performance metrics; resource management 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, by the DARPA/ITO AICE program under contract numbers DABT63-99-C-0010 and DABT63-99-C-0012, and by the Colorado State University George T. Abell Endowment. Intel and Microsoft donated some of the equipment used in this research.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUL 2003	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Flexible Multi-Dimensional QoS Performance Measure Framework for Distributed Heterogeneous Systems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School 833 Dyer Rd., Code CS/Cp Monterey, CA 93943-5118**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**27**	

1. Introduction

In many distributed heterogeneous environments, the tasks that are executing have different quality of service (QoS) requirements. These different QoS requirements impose different machine and resource requirements. Furthermore, these tasks may require input data from a variety of distributed sources. Mixed-machine heterogeneous computing (HC) environments (e.g., a cluster of heterogeneous computers) 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., [BrS98, Esh96, MaB99]). 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 QoS. When users' tasks in a distributed heterogeneous computing environment are allocated resources, the total demand placed on some system resources by the tasks, for a given interval of time, may exceed the availability of those resources. In this case, some tasks may receive degraded service, or be dropped from the system. In the evaluation of the performance of an RMS, it is essential to include: (1) how well it performed its goals or functions and (2) 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 included in such a performance measure, provides a way to combine them, and discusses other issues such as multiple versions of tasks, relative importance of the different attributes, a generalization of the measure, and priority levels with classes.

The proposed approach is called the Flexible Integrated System Capability (FISC) measure. It is a multi-dimensional performance measure, and may include factors such as priorities, versions of a task or data, deadlines, situational mode, security, application 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. It provides one way of combining the factors listed above. For an environment where it is important to investigate how well data communication requests are satisfied, the collective value of requests satisfied can be the basis of the overall performance measure instead of tasks completed.

The FISC measure by itself is not an RMS evaluator; other factors, such as RMS execution time, need to be included. 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., [BrS01, BrS02, KiS03, MaA99, ThB00, ThT00]). 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., [SiS82]). In some situations, a combination of QoS (or performance) attributes must be considered (e.g., [LiA97]). The FISC measure will be focused on calculating the value of the tasks completed using various QoS attributes. The measure presented here is one linear instantiation of the FISC measure. As will be discussed, a non-linear measure is also possible.

This research is part of three related DARPA programs: Quorum [HeK99], Battlefield Awareness and Data Dissemination (BADD) [DAR99, Roc96], and the Agile Information Control Environment (AICE) [AIC98]. In the Quorum environment, it is sometimes the case that not all tasks requested can achieve their most preferred QoS by their deadline. Thus, there must be a performance measure that can determine a collective value of the set of tasks that were completed in a given time interval by a particular resource management strategy.

One aspect of the BADD and AICE programs involves designing a scheduling system for forwarding (staging) data items prior to their use as inputs to a local application in a wide area network (WAN) distributed computing environment. The BADD and AICE systems are similar to the Quorum environment in that, in some situations, not all data requests will be satisfied with their most preferred QoS by their deadline. Thus, the goal of the scheduler is to satisfy as many requests as possible in a given interval of time, in a way that has the greatest collective perceived value.

The performance measure described in this research can be used to evaluate, for a given interval of time, the total value of tasks completed in the Quorum program or the total value of data received in the BADD and AICE programs. In this sense, the set of completed tasks for the Quorum program is equivalent to the set of satisfied data item requests for the BADD and AICE programs. A major difference between Quorum and BADD/AICE is that in Quorum tasks are assigned to resources by the RMS. In the BADD/AICE program, task assignments are given and

fixed *a priori*, but the movement of data to the tasks must be scheduled. Throughout the rest of this paper, a task will be used to represent (1) a user's process execution in the Quorum context and (2) a user's request for a data item in the BADD/AICE context. While this research is motivated by military applications, the FISC framework can be adapted for other environments, such as clusters, intra-nets, and certain computational grids [FoK99].

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 provides a brief overview of some of the literature related to this work. In Section 3, several examples of individual QoS attributes are presented. These attributes may be considered when formulating the performance measure to be used in building and assessing RMSs. Section 4 shows how all the example QoS attributes can be combined into a single measure. In addition, this section presents a baseline for the FISC measure, discusses a generalized form of the performance measure, and possible variations of the FISC measure. Examples of where the FISC measure can be used are 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 in a distributed computing system. There is much literature on parameters considered important when scheduling and mapping; 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 [KiS94]. 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 are. While this is also the case for Quorum and BADD/AICE, the FISC research presented here includes other attributes that are important in evaluating the overall value of the tasks completed.

In [LiM94], laxity (deadline minus latency) of a task is used for the scheduling, adjustment, and dropping of messages. The "Least-Laxity-First (LLF)" scheme gives an improved missed deadline ratio, which is the rate of messages missing their deadlines. In [LiM94], only the timing constraints are used in the scheduling of messages and evaluation of the LLF scheme, but the research effort does not consider other QoS attributes used in heterogeneous distributed networks that the FISC measure includes.

The work presented in [Mar90] describes an algorithm that enables each node of a system to schedule the transmission of messages generated locally while obeying their deadline constraint (messages get dropped if they cannot meet their deadline). 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 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 [ThT00]. The research in [ThT00] 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 [ThT00] 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. A measure of the overall value accrued of completed tasks is needed that can be used in an objective function to compare and analyze RMSs while incorporating all the QoS parameters used. The works mentioned above consider only a subset 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. The QoS parameters that affect the overall value of requests satisfied are discussed in our 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 [ChS98] 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 is assigned to each user application. An approach where requested QoS is taken into account when scheduling computational resources in a network is presented in [Mah99]. 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 [Kre97]. The FISC measure serves a purpose similar to the performance measures in [ChS98, Mah99, Kre97]. However, the research presented here provides a more detailed description of a measure using more parameters, so that it can be used to compare the performance of schedules in the Quorum and BADD/AICE environments. Furthermore, the QoS input specification for ERDoS [SaC97] accounts for only two specific security parameters (confidentiality and integrity), whereas the security component of the FISC measure can describe an arbitrarily complex set of security features.

The resource allocation model for QoS management proposed in [RaL97] indicates multiple dimensions of QoS and multiple resources. In [RaL97], some of the QoS attributes studied in this research are mentioned, however there is no detailed description or discussion of those attributes. The utility that is described in [RaL97] is 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 [LeL99a, LeL99b] presents specific usage of the model presented in [RaL97]. 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 work in [WaW98] proposes a market mechanism that uses the length of a job, the deadline of a job, the price of the job done, and the price of allocated time slots to find the optimal allocation of jobs onto resources. The FISC measure uses priorities and other QoS measures to calculate the collective value of tasks that are completed.

The research in [Mar99] 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 [Mar99] and the FISC measure both seek to achieve the optimum value of the requests satisfied. In our paper, multiple QoS attributes (e.g., deadlines, security) are considered in detail while in [Mar99], the QoS factor is represented by the link congestion.

In the model proposed by [CoS99], 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 used to calculate the net benefit. The FISC measure and the utility function are similar in that they calculate the overall value achieved from the resources allocated. While [CoS99] considers only bandwidth, the FISC research 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 [ScS98]. 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 [BaS95, BlF96, CoL98, RyN98]. 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 [WaW97], where high-level static security properties can be decomposed hierarchically. However, in [WaW97] the approach cannot accommodate the measurement of how well an executed task meets the security requirements of its environment. Nor does [WaW97] account for variant security policies or mechanisms.

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 the relative importance of each level is qualitatively reflected (e.g., high, medium, and low). The policy makers may be the commanders in a military environment or executives in a corporation. Policy makers may assign different users, or classes of users, restricted ranges of priority levels that can be assigned to their tasks. Alternatively, a task itself could have an immutable priority level assigned to it by the policy makers. Each priority level will then be given a weight that can be calculated by a priority weight function, which is pre-determined by policy makers, described later in this section.

Priority levels with relative, quantitative weightings should be incorporated in scheduling systems so that a task with a higher importance will have a higher probability of meeting its QoS requirements. Application users and system builders often assign an arbitrary numbering scheme to priority levels that does not meaningfully quantify the relative importance of one priority level to another. A more meaningful weight must be assigned to each priority level so that t he relative importance can be reflected in the performance measure.

The relative importance (weighting) of priority levels may vary depending upon the situational mode. For example, there may be military modes of peace and war. In peace mode, it might be just as important to complete

ten low priority level tasks as to complete one high priority level task. However, in war mode, one high priority level task might be more important than 1,000 medium priority level tasks. To indicate this relative importance, for example, it may be necessary to give weightings of 10,000 to high priority level tasks and 10 to medium priority level tasks. This dependency can be indicated in the performance measure by expressing the weight of all priority levels as a function of the situational mode.

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, 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 of completing the corresponding task.

3.2. Versions

A task may exist in different versions, each with its own resource requirements. Because of system load, it may not be possible to complete the most desired version of a task. For example, a user requesting a map application may most desire a 24-bit color, three-dimensional topographical map. However, if this cannot be given to the user due to limited resources, the user would rather have a black and white, two-dimensional map than nothing at all.

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 [SaC97], in the sense that each version of a task may have different accuracy and precision, or to having different video image sizes considered in [XuN01]. For each version of a given task, a worth (preference) relative to the other versions will be indicated by the application developer, the user, or the policy makers. These worths may be a function of the situational mode. In the above example, the black and white version may only be worth 75% of the color version to the user. When selecting a version of a task to execute, an RMS's scheduling algorithms must consider this worth and the task's resource requirements as well as the availability of these resources. For example, one version may not be viable because its bandwidth requirement is too high. Typically, a higher worth version has greater resource requirements.

To allow worths 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. For example,

assume that all factors except version worths are equal across a set of tasks. The user can specify any number for the worth of a version as shown in Table 1 (a). Therefore, without a normalization procedure, a task with the largest worth version may always be chosen for processing ahead of other tasks for no logical reason. For example, if there is enough resources to complete one version of one of the tasks in Table 1 (a), task 1 will be chosen over all other tasks because version 2 of task 1 has the highest worth among all the versions of all the tasks. In extreme cases, worths that are not normalized could supersede the impact of priority depending on how priorities and worths of version interact.

To avoid this type of anomalous behavior, worths are normalized as follows. Assume there are I_j versions for a given task j. Let v_{ij} be the i-th ($0 \le i < I_j$) 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. Example $w_{ij}(m)$ values are provided in Table 1(a). One approach to the normalization problem is to divide each indicated worth of a task version by the largest worth for that task, resulting in the normalized worth as shown in Table 1(b). Thus, the normalized worth (η_{ij}) of $w_{ij}(m)$ is given by

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

Figure 1 is the graph representation of the normalized worth shown in Table 1(b). 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 relative to the version with the largest worth.

Another approach to the normalization would be to divide each version's worth by the total sum of the version worths of the task. This would not guarantee equal value for the most preferred version of each task. Furthermore, this approach may allow a greedy user to obtain a higher value for a given task's preferred version over another task's most preferred version. For example, consider task 0 and task 1 in Table 1(a). If this alternative approach is used, the normalized worth for task 0 would be 0.1, 0.1, and 0.8, while for task 1 it would be 0.25, 0.35, and 0.4. This means that, even if task 0 and task 1 have the same priority, the largest worth version of task 0 is worth more than the largest worth version of task 1, which should not be the case.

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 , soft , and firm. These deadlines are illustrated by example in Figure 2. The deadline attributes are related to the timeliness parameter in [SaC97].

The earliest deadline (e_{ij}^{d}) is the time when the task can start. For example, assume that data is being sent to a system is to process it. The task of sending data cannot start if the receiving system is not ready and it will have no value.

The soft deadline (s_{ij}^{d}) is the time by which a task must complete to be of full value to the user [StS98]. 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}) will have 0 value, because the task will be of no use after that deadline [LeK96, StS98]. For example, if a task that shows a map of an area completes after a mission is finished, then it will have no value. If a task completes between its soft and firm deadline, then it will have some fraction of its maximum possible value. For each task, the fraction of total value for each point between the soft and firm deadlines, and the time between the soft and the firm deadlines, may be a function of the situational mode. For example, during war mode, the soft and firm deadlines may be identical.

Let τ_{ij} be the time that the *i*-th version of task *j* actually completes. The deadline function δ_{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 of security services. This multi-dimensional space can be represented with a vector ($\boldsymbol{S}$) of security components, where the functional requirement for each component is specified by a Boolean statement for each given situational mode. Both resources and tasks may have multiple security components [IrL00a, LeI99b].

The instantiation of a network task either meets, or does not meet, each component's requirement. For example, consider the *i*-th version of task *j*. Let $\underline{R_{ij}}$ be a set of resources utilized by v_{ij} and let $\underline{\boldsymbol{S_{ij}}}$ be a sub-vector of vector $\boldsymbol{S}$. A component $\underline{\kappa}$ in $\boldsymbol{S}$ is in $\boldsymbol{S_{ij}}$ if and only if κ depends on v_{ij} or on an element of R_{ij}, and is denoted $\underline{\boldsymbol{S_{ij}}.\kappa}$. The characteristic function $\underline{\sigma_{ij}'}$ is used to represent required security attributes. If minimum security requirements are not all met, there is no value accrued for executing v_{ij} and $\sigma_{ij}' = 0$. If the instantiated Boolean value of all κ in $\boldsymbol{S_{ij}}$ is true, then $\sigma_{ij}' = 1$.

Additionally, some security components of a task can be <u>variant</u> in that they allow a range of behavior with respect to a requirement (e.g., length of cryptography key may vary between 40 and 256). For variant components, the user may request a particular value or permit a choice from a component's defined range. The RMS must select a specific value within the user's range, while considering resource availability, for the completed task to have a non-zero value. The measure will give only partial credit for a task completed with less than the most secure value in the defined range.

The desire to provide *adaptable* security motivates the inclusion of variant security components in the system [IrL00b]. Thus, security affects the performance measure when components are variant. For example, assume the percentage of authenticated packets can range between 50% and 90% in increments of 10%. The increment quantizes the range. Let $[\underline{\boldsymbol{S_{ij}}.\kappa}]$ be the number of quanta in $\boldsymbol{S_{ij}}.\kappa$ (in the above case, this is five) and $\underline{g_{ij}.\kappa}$ be the fraction of κ in $\boldsymbol{S_{ij}}$ satisfied. If a task achieves the third quantum (70%), then $g_{ij}.\kappa$ is $3/[\boldsymbol{S_{ij}}.\kappa] = 3/5 = 0.6$. This example can be represented as a graph as shown in Figure 3.

Suppose $\underline{n}$ is the number of security components in $\boldsymbol{S_{ij}}$. 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 Equation 2.

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

The above is just one possible way to combine the values of these security components. For example, the $g_{ij}.\kappa$ values in Equation 2 can have relative weightings for a given *m*. Thus, if the military situation changes from peace to war, authentication may be considered relatively more important and might be given a higher relative weighting.

The overall <u>security measure</u> is $\sigma_{ij} \times \sigma_{ij}'$, where $0 \le \sigma_{ij} \times \sigma_{ij}' \le 1$. It indicates how the value of a task may be degraded due to lack of its most desirable security services or a lack of meeting its minimum requirements.

3.5. Application Specific QoS

The model for application specific QoS is analogous to the security model in Subsection 3.4. There is a multi-dimensional space of application QoS attributes (e.g., jitter level, frame rate, bit error rate). For example, in [XuN01], the frame rate, image size, number of trackable objects, and buffer sizes are the application specific QoS attributes of the video streaming and tracking service (application). The overall application specific QoS measure is $\alpha_{ij} \times \alpha_{ij}'$. where α_{ij} and α_{ij}' are analogous to σ_{ij} and σ_{ij}' respectively. It indicates how the value of a task may be degraded due to lack of its most desirable application specific services or a lack of meeting its minimum requirements.

3.6. Associates

There are many possible types of inter-task dependencies, e.g., for the MSHN environment, consider a task whose only function is to generate data for other tasks (descendants). There may be an inter-related set of such tasks (Figure 4). If there is a descendant along a dependency path of tasks that generates an output for a user (e.g., tasks 4 and 6 in Figure 4) and if this descendant completes its execution, then the tasks that did nothing more than generate data for this particular task will have value, otherwise they will not. This is because the end user that submitted a task for completion will acknowledge the task to be finished only when the actual results can be determined. Thus, for a task to have value, either it or one of its descendants must generate an output for a user.

The first task in a dependency path that does more than generate data for a subsequent task will be known as a required associate of its predecessors. The variable ρ_{ij} will represent whether a required associate of a given task completes; i.e., if at least one required associate of a given task completes, then $\rho_{ij} = 1$, otherwise $\rho_{ij} = 0$.

For the BADD/AICE environment, consider a data request whose only function is to be used by an application to generate data for other applications. There may be multiple data requests for a given application. Unless all such data requests are satisfied, the application cannot execute and the value of any data request and the application that are satisfied would be zero (i.e., $\rho_{ij} = 0$ for all data requests). In Figure 5, data requests 1, 2, and 3 are required for application 1 to execute. Therefore, if any one of these data requests is not available (if only a subset of the data requests arrive by the firm deadline), there is no value for any of the satisfied data requests.

4. Performance Measure

4.1. FISC Measure

A meaningful way to combine the attributes discussed in the previous section is proposed here. The measure presented here is only one instantiation of the 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, some applications may be compute-intensive and others interactive, perhaps having stringent security requirements. In this research, the collective value of services achieved is used for determining how "good" a resource allocation is. A meaningful way to combine the performance attributes previously discussed is proposed in this subsection.

Consider a set of tasks with different priorities, different deadlines, multiple versions, different security and application specific QoS requirements, and dependencies. The value of a task must be calculated based on all of the attributes.

The maximum value of task j is $\pi(p_j)$; therefore, other attributes must be combined in a way that the maximum value of a task never exceed its priority weighting. In addition, all performance attributes must be included in the calculation of the value of a task. The factors that describe whether a minimum requirement of an attribute is met must be included. These are ρ_{ij}, δ_{ij}', σ_{ij}', and α_{ij}', and each function is equal to 0 if minimum requirement is not met or 1 if the minimum requirement is met. The intuition is that if a task does not meet its minimum requirement, the task is of no value to the user. Also, the RMS must consider any variations in the value of a task's completion that is a result of when a task completes (δ_{ij}), which version was used for the task (η_{ij}), and receiving different degrees of required and desired security (σ_{ij}), and other application- and domain-specific QoS services (α_{ij}). Equation 3 is one way to combine all performance attributes. The "max" function is used to indicate whether a version of a task has completed. Note that if a version of a task completes then all other versions are not completed and their calculated values are zero. To make the value of "max" less than or equal to one, the value of δ_{ij}, η_{ij}, σ_{ij}, and α_{ij} is averaged as shown in Equation 3. This also gives the variable component of each attribute equal weighting.

$$\sum_{j=0}^{t-1} \pi(p_j) \times \left(\max_{0 \le i < I_j} \rho_{ij} \times \delta_{ij}' \times \sigma_{ij}' \times \alpha_{ij}' \times \frac{\eta_{ij} + \delta_{ij} + \sigma_{ij} + \alpha_{ij}}{4} \right) \tag{3}$$

This version of FISC will be called the averaged FISC. This method is similar in concept to the benefit function described in [IrL00a, IrL00b].

There can be coefficients for each attribute mentioned in this research 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 (version used), δ (deadline met), σ (security services satisfied), and α (application specific QoS satisfied) factors respectively. To incorporate coefficients into the FISC measure, another version as shown in Equation 4 of the FISC is needed. For the coefficients to not affect the overall priority weighting and to indicate the relative importance among the attributes, the measure will be divided by the sum of the coefficients. By dividing the measure by the sum of the coefficients, the part of the FISC measure without the priority function will be one when all attributes are 100 percent satisfied and less than one when a certain task gets degraded QoS.

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

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. Therefore, in any time instant, the total amount of resource used of a particular resource cannot exceed the total available resource at that time instant. Assume that there is Ξ number of resources in the system. Let $\underline{R_{rj}}(\Delta)$ be the amount of resource $\underline{r}$ ($0 \le r < \Xi$) used for task j ($0 \le j < t$) during the time interval $\underline{\Delta}$ and let $\underline{U_r}(\Delta)$ 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 explained in Equation 5.

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

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. Then the FISC equation can be extended to include $v_{ij\gamma}$, where i is the version, j is the task, and γ is the copy number ($0 \le \gamma < \Gamma_j$). The FISC measure can be extended from Equation 4 to Equation 6 (averaged FISC with coefficient and multiple copies of versions). The goal would be to maximize the FISC measure over all relevant i and γ for j. For each copy of a version there could be different deadlines, security services, or application specific QoS. Therefore, these factors also need to consider the copy γ of the version. Because dependency is only among tasks not versions of a particular task, the dependency factor does not need to consider the copy number.

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

The FISC measure presented can be used to compare the performance of different RMSs operating in the same environment. A direct comparison using the FISC measure of two RMSs that operate in two different environments would not make sense. One method to compare different RMSs operating on different distributed systems, is to normalize the FISC measure by a baseline that depends on the tasks and underlying distributed system. The baseline can be calculated by using the same number of tasks with same attribute requirements for each environment and these baselines may be different. Because the environments are different, how well a RMS performed would equal to how much better it performed than the baseline of 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 builds upon and extends the example given by [ThT00]. 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 $\rho_{ij} = 1$. 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. The algorithm to obtain a baseline is shown in Figure 6 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.

When the FISC measure is normalized by a baseline, the resulting function is called the FISC ratio. The averaged FISC ratio is:

$$\frac{\sum_{j=0}^{t-1} \pi(p_j) \times \left(\max_{\substack{0 \le i < I_j \\ 0 \le \gamma < \Gamma_j}} \rho_{ij} \times \delta_{ij\gamma}' \times \sigma_{ij\gamma}' \times \alpha_{ij\gamma}' \times \frac{\left[c_\eta \times \eta_{ij\gamma} + c_\delta \times \delta_{ij\gamma} + c_\sigma \times \sigma_{ij\gamma} + c_\alpha \times \alpha_{ij\gamma} \right]}{c_\eta + c_\delta + c_\sigma + c_\alpha} \right)}{\text{baseline}} \tag{7}$$

Another version of FISC that will be called the multiplied FISC is presented. Similar to the averaged FISC, this version must consider all attributes and make sure that the "max" never exceeds 1. A way to accomplish this is to multiply all components of the measure (η_{ij}, ρ_{ij}, δ_{ij}, δ_{ij}', σ_{ij}, σ_{ij}', α_{ij}, and α_{ij}') as shown in Equation 8.

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

Both versions of the FISC measure (averaged and multiplied) mentioned in this subsection may be used to calculate the collective value of the tasks completed. The averaged FISC (Equation 3) captures the intuition that when calculating the value of a task, the value should be larger than or equal to the percentage satisfied of the least satisfied service attribute multiplied by the priority weighting of the task. For example, there is a task completed with a version that is worth 50%, the task was completed before the soft deadline (100%), the task received variable security services (40%), and assume $\pi(p_j)$ is one. Assume all minimum requirements are met and all other variable services do not exist. The completed task's value would be 0.63 by the averaged FISC (Equation 3) and 0.2 by the multiplied FISC (Equation 8). When using the multiplied FISC, the value of the task has decreased below the security services satisfied (40%). When all of a task's services are at least 40% satisfied, the value of the task should be larger than or equal to 40% of the maximum value of a task.

4.2. Generalization of FISC Measure

The previous subsection describes a particular example of the FISC measure. It can be generalized such that the numerator is any function of $\pi(p_j, m)$, η_{ij}, ρ_{ij}, δ_{ij}, δ_{ij}', σ_{ij}, σ_{ij}', α_{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 ($0 \leq y < u$) 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

$$FISC = \mathrm{f}(P_0, P_1, \ldots, P_{u-1})/ \text{ baseline and} \tag{9}$$

$$P_y = \mathrm{f}_y(s_0, s_1, \ldots, s_{v_y-1}), \tag{10}$$

where each s_e is a secondary factor for P_y. Linear or nonlinear weightings (coefficients) of each factor, 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 in Subsection 4.1 is one method of normalizing the numerator of the FISC measure. Other methods for normalizing could be incorporated to compare the performance of different RMSs in a given environment.

4.3. FISC Measure with 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. For example, if there are high, medium, and low priority level tasks, the high priority tasks will be considered first and if there are no more high priority tasks, then medium and low priority level tasks will be considered for execution. 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. If all the tasks of a certain class have been considered for the calculation of the FISC number, then the tasks of the next class will be considered for calculation.

Each task will have a priority level, and priority levels will have relative weightings. Tasks will not have classes but the priority level that the task was assigned with may correspond to a class predetermined by the system administrator or the policy maker. Each class will consist of one or more priority levels and there can be several

classes where the number of priority levels assigned to a class can be different. For any number of classes, the FISC number, which is calculated using the FISC measure, of class *k* is more important than the FISC number of class *k*+1 where *k* is an arbitrary number. Therefore, when comparing the accrued value of one scheduler to another, the FISC number of the highest class will be compared first and if they have equal FISC numbers, the FISC number of the next highest class will be compared.

As shown in Figure 7, there could be $\underline{L}$ number of priority levels and $\underline{C}$ number of classes. Priority level 1 is more important than priority level 2 and class 1 is more important than class 2. Any number of priority levels can be in one class. After calculating the FISC number for class 1 (priority 1 tasks) of schedulers, compare the number and if the number is same, calculate the FISC number for the next class of tasks.

5. Examples of Where FISC can be Used

5.1. QuO Middleware

The Quality Objects (QuO) middleware is a set of extensions to standard distributed object computing middleware that is used to control and adapt quality of service in a number of distributed application environments, from wide-area to embedded distributed applications [LoS01]. Several examples of QoS attributes that are used in real applications are described.

The first example is data dissemination in a wide-area network. In cases where the network is the source of a slowdown in the system and bandwidth reservation is not successful, the QuO middleware triggers application adaptation. The application trades off its data quantity or data quality requirements for its timing requirement, by requesting smaller images or lower resolution images to reduce the amount of network congestion. If the CPU is the source of slowdown, the application requests unprocessed images reducing the load on the CPU and enabling the images to be received faster but with reduced quality or analysis of the images. To evaluate the overall performance of the network, the FISC measure can calculate the value of the requests completed using the normalized worth η_{ij} (assuming each request has some kind of preference for different image sizes, resolutions, quality, or analysis) and the deadline graph (example shown in Figure 2). If the request received a lesser quality image than it had preferred, the request is receiving degraded QoS and the FISC measure will indicate this by giving the request a lower value than the maximum it can get.

The second example is dynamic mission planning in an avionics platform. In this example, QuO is used to manage the trade offs of timeliness versus image quality by image compression, image tiling, processor resource management, and network resource management. Using the FISC measure as described in the first example, the overall performance of the trade offs can be evaluated.

In the third example, Unmanned Air Vehicle (UAV) data is disseminated throughout a ship. While the data is sent out, system condition objects monitor the frame rate and the host load on the video display hosts. As the frame rate declines and/or the host load exceeds a threshold, they cause region transitions, which trigger the video distribution process to drop frames going to the display on the overloaded host and the video display on the overloaded host is told to reduce its display frame rate to the rate at which frames are being sent it. The application specific QoS attribute described in Section 3.5 can be used. If the frame rate is reduced from the most preferred frame rate (therefore receiving degraded service), the value of the task of sending frames over the network will be determined using α_{ij} (a graph similar to the one shown in Figure 3 may be used), and α_{ij}'.

While the QuO system monitors system loads, makes trade-offs of timeliness versus image quality, and degrades QoS attributes while dropping frames and reducing display frame rate, it needs a performance measure that can estimate how well these activities were done in terms of the overall performance. The FISC measure can provide a framework for estimating the overall performance of such a system.

5.2. EADSIM

As an example of how the FISC measure might be applied in practice, consider the following scenario. The Joint Force Air Component Commander (JFACC) staff are preparing an Air Tasking Order (ATO). As the ATO develops, one tool available to the JFACC staff for its evaluation is the Extended Air Defense Simulation (EADSIM) system from the US Army Space and Missile Defense Command. EADSIM is a warfare modeling application offering great flexibility in the areas modeled, the capabilities of the platforms simulated, and the method of simulation (deterministic or stochastic) [Por99].

EADSIM utilizes a wide range of computing resources, depending on the features enabled. For example, the stochastic mode may use approximately 20 times the computing resources as the deterministic mode (based on the number of runs required to obtain a statistically significant number of samples). Of course, results obtained in stochastic mode are likely to be more reliable.

The JFACC planners select two versions of EADSIM, the stochastic mode and the deterministic mode, and submit them, with different preferences, to their RMS for execution. Because this information is urgently needed for combat mission planning, the priority of this request is seven on a scale of ten (ten being highest). Using the priority level, p_j of the request, the priority weighting can be calculated using a predetermined function $\pi(p_j)$. The request has a firm deadline and the results are required in an hour after the request is submitted. The deadline graph shown in Figure 2 with the soft deadline being the same with the firm deadline can be used. Therefore, if the results are received within an hour then $\delta_{ij} = 1$ and the overall value of the request is still $\pi(p_j)$. However, if the results are not presented after an hour, they have no value ($\delta_{ij} = 0$). The stochastic version is preferred because it will produce higher confidence results, but the deterministic simulation may also be useful because of faster execution time. Assume that the stochastic version is assigned a preference of eight, on a scale of ten and the deterministic version is assigned a preference of five. Using the FISC measure, the preferences can be normalized and the normalized worth (η_{ij}) of the stochastic version is 1 and the η_{ij} of the deterministic version is 0.625. If there is enough resources to complete only one of the two versions and these are the only ones to choose from, then the stochastic version will be completed because it has a higher worth than the deterministic version. The security level scheme is binary. The information must be sent over a secure link. If it is, the request is assigned a security value of 1 ($\sigma_{ij}' = 1$), if not, it is assigned a security value of 0 ($\sigma_{ij}' = 0$). The FISC measure determines how well a request is satisfied in terms of value accrued.

An RMS such as MSHN would evaluate the expected resource requirements of each version as well as the ability to complete each version based on the current resource availability. Using this information, the RMS could make a decision by maximizing an objective function where the FISC measure would be a major component.

6. Summary

In some environments, the 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, quality of service (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. By using 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, it may be used in a simulation mode to analyze the impact of proposed changes to the distributed system. For the FISC measure to be more flexible, a generalization of the measure is discussed as well as the use of classes in addition to priority levels. Examples of how the FISC measure can be used are presented for two different environments.

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: how to determine the relative weighting of the π (priority level weighting), η (version used), ρ (required associate present or not), δ (deadline met), σ (security services satisfied), and α (application specific QoS satisfied) factors; using a non-linear combination of task values to compute the overall measure; whether to use negative fractions in the deadline function in case of 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 a task to the user; and investigating variations in the factors already considered.

Acknowledgments: The authors thank, B. Beaton, G. Koob, J. Rockmore, M. Jurczyk, I. Wang, S. Jones, J. Kresho, E. K. P. Chong, R. Eigenmann, N. Rowe, C. Kesselman, N. Beck, T. Braun, S. Ali, S. Chatterjea, A. Naik, and P. Dharwasdkar for their valuable comments and suggestions. A preliminary version of portions of the material was presented at the 10th Heterogeneous Computing Workshop.

Table 1: (a) Worths that users indicate for each version of a task. (b) The worth for each version of a task is divided by the largest worth of that task to get the normalized worth.

task \ version	0	1	2
0	1	1	8
1	25	35	40
2	.2	.3	.5
3	.1	.2	.7

(a)

task \ version	0	1	2
0	.125	.125	1
1	.625	.875	1
2	.4	.6	1
3	.143	.286	1

(b)

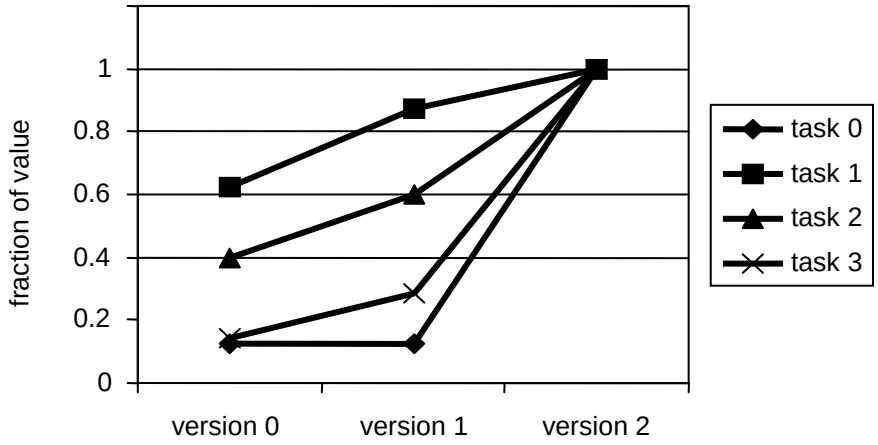

Figure 1: Graph representation of the normalized worth of each version of a task.

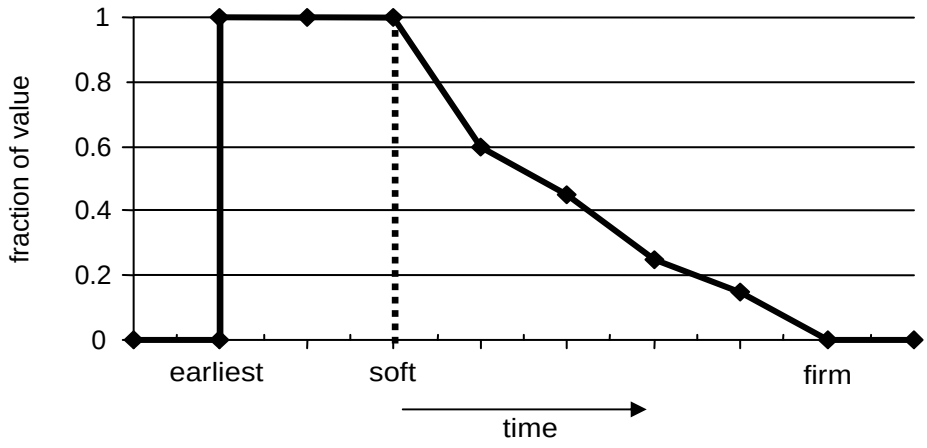

Figure 2: The deadline graph shows the variation in the value of a task with various deadlines.

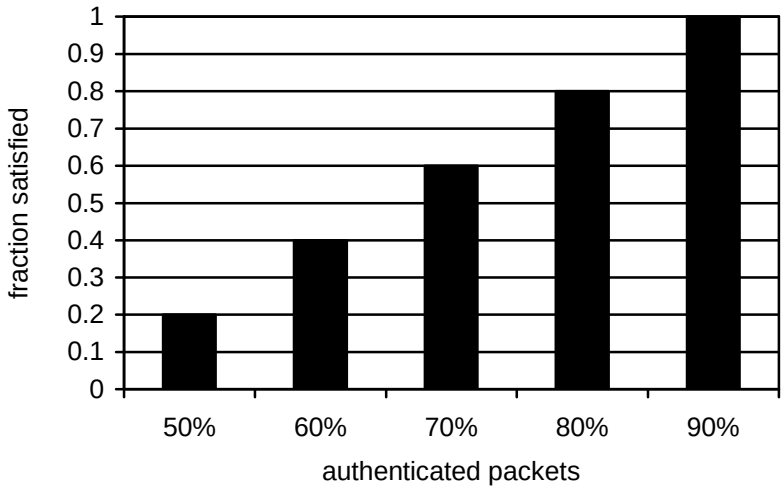

Figure 3: Graph representation of fraction satisfied of authenticated packets that can range between 50% authenticated and 90% authenticated, incremented by 10%.

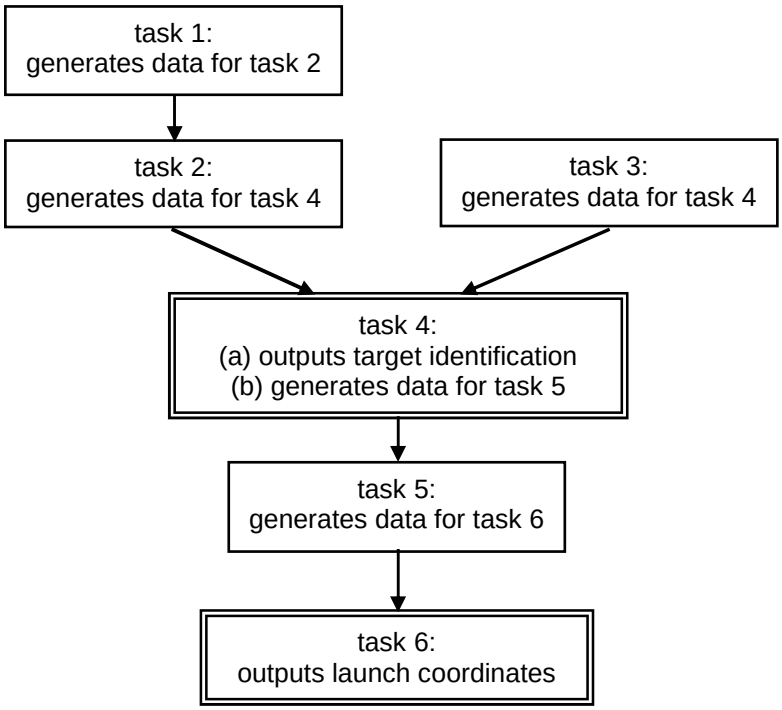

Figure 4: An example set of tasks that have dependency. Tasks 1, 2, 3, and 5 are tasks that only generate input data for other tasks. Tasks 4 and 6 generates output for the user.

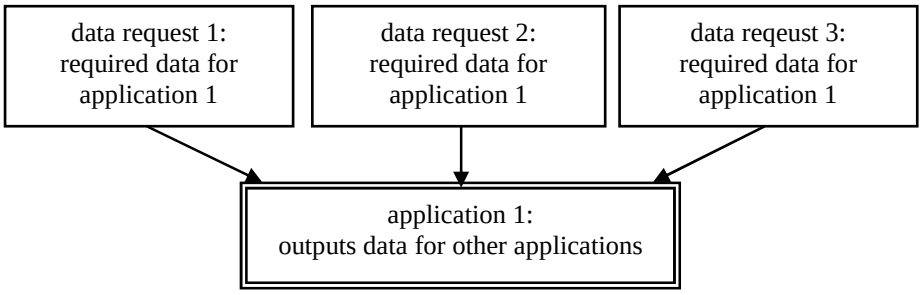

Figure 5: An example set of data requests that have dependency. Data requests 1, 2, and 3 are required input data for application 1.

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 $\eta_{ij} = \delta_{ij} = \alpha_{ij} = \sigma_{ij} = 100\%$ and $\rho_{ij} = 1$
    schedule at soonest possible time
        add $\pi(p_j)$
        update status of resources
    else
        no value added
        no resources consumed
}

Figure 6: Baseline algorithm

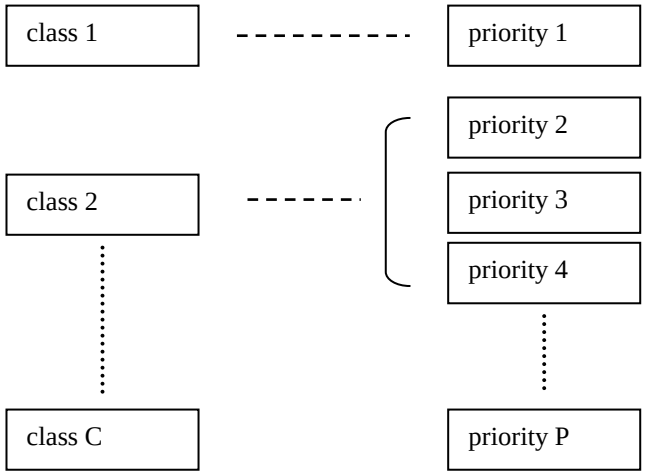

Figure 7: Example of priority levels within classes.

References

[AIC98] AICE, Agile Information Control Environment Proposers Information Package, BAA 98-26, September 1998, http://web-ext2.darpa.mil/iso/aice/aicepip.htm.

[AlK03] Shoukat Ali, Jong-Kook Kim, Yang Yu, Shriram B. Gundala, Sethavidh Gertphol, Howard Jay Siegel, Anthony A. Maciejewski, and Viktor Prasanna, "Utilization-based techniques for statically mapping heterogeneous applications onto the hiper-d heterogeneous computing system," *Parallel and Distributed Computing Practices*, accepted to appear in 2003.

[BaS95] 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.

[BlF96] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," *1996 IEEE Symposium on Security and Privacy*, May 1996, pp. 164-173.

[BrS98] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao, "A taxonomy for describing matching and scheduling heuristics for mixed-machine heterogeneous computing systems," *IEEE Workshop on Advances in Parallel and Distributed Systems*, October 1998 (in the proceedings of the *17th IEEE Symposium on Reliable Distributed Systems*, October 1998, pp. 330-335).

[BrS01] T. D. Braun, H. J. Siegel, N. Beck, L. Boloni, R. F. Freund, D. Hensgen, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and Bin Yao, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, Vol. 61, No. 6, June 2001, pp. 810-837.

[BrS02] T. D. Braun, H. J. Siegel, and A. A. Maciejewski, "Static mapping heuristics for task with dependencies, priorities, deadlines, and multiple versions in heterogeneous environments," in the CD-ROM proceedings of the *16th International Parallel and Distributed Processing Symposium*, (*IPDPS 2002*), April 2002.

[ChS98] S. Chatterjee, B. Sabata, and J. Sydir, *ERDoS QoS Architecture*, Technical Report, SRI, Menlo Park, CA, ITAD-1667-TR-98-075, May 1998.

[CoL98] M. Condell, C. Lynn, and J. Zao, "Security policy specification language," *INTERNET-DRAFT*, Network Working Group, October 1998, ftp://ftp.ietf.org/internet-drafts/draft-ietf-ipsec-spsl-00.txt.

[CoS99] C. Coutcoubetis, G. D. Stamoulis, C. Manolakis, and F. P. Kelly, "An intelligent agent for optimizing QoS-for-money in priced ABR connections," *Telecommunications Systems*, Special Issue on Network Economics, (at http://www.statslab.cam.ac.uk/~frank/PAPERS/iaabr.html).

[DAR99] DARPA, Battlefield Awareness and Data Dissemination, April 1999, www.darpa.mil/iso/ badd/.

[Esh96] M. M. Eshaghian, ed., *Heterogeneous Computing*, ArTech House, Norwood, MA, 1996.

[Fer78] D. Ferrari, *Computer Systems Performance Evaluation*, Prentice-Hall, Englewood Cliffs, NJ, 1978

[FoK99] I. Foster and C. Kesselman, eds., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco, CA, 1999.

[HeK99] 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*), April 1999, pp. 184-198.

[IrL00a] 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.

[IrL00b] C. E. Irvine and T. Levin, "Toward a taxonomy and costing method for security services," *15th Annual Computer Security Applications Conference*, December 2000, pp 183-188.

[KiS94] 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, December 1994, pp. 707-715.

[KiS03] J.-K. Kim, S. Shivle, H. J. Siegel, A. A. Maciejewski, T. D. Braun, M. Schneider, S. Tideman, R. Chitta, R. B. Dilmaghani, R. Joshi, A. Kaul, A. Sharma, S. Sripada, P. Vangari, and S. S. Yellampalli, "Dynamic mapping in a heterogeneous environment with tasks having priorities and multiple deadlines," *12th Heterogeneous Computing Workshop* (in the proceedings of the *17th International Parallel and Distributed Processing Symposium* (*IPDPS 2003*)), April 2003.

[Kre97] J. P. Kresho, *Quality Network Load Information Improves Performance of Adaptive Applications*, Master's Thesis, Department of Computer Science, Naval Postgraduate School, Monterey, CA, September 1997 (D. A. Hensgen, advisor), 164 pp.

[LeK96] 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*, October/ November 1996, pp. 74-80.

[LeL99a] 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.

[LeL99b] 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*, December 1999, pp. 315-326.

[LiA97] 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, October/December 1997, pp.18-28.

[LiM94] J. P. Li and M. W. Mutka, "Priority based real-time communication for large scale wormhole networks," *8th International Parallel Processing Symposium*, April 1994, pp. 433-438.

[LoS01] J. Loyall, R. Schantz, J. Sinky, P. Pal, R. Shapiro, C. Rodrigues, M. Atighetchi, D. Karr, J. M. Gossett, and C. D. Gill, "Comparing and contrasting adaptive middleware support in wide-area and embedded distributed object applications," *21st International Conference on Distributed Computing Systems* (*ICDCS 2001*), April 2001, pp. 625-634.

[MaA99] 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, November 1999, pp. 107-121.

[MaB99] 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.

[Mah99] 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.

[Mar90] D. C. Marinescu, "A protocol for multiple access communication with real-time delivery constraints," *IEEE INFOCOM '90*, June 1990, pp. 1119-1126.

[Mar99] P. Marbach, "Pricing priority classes in a differentiated services network," *37th Annual Allerton Conference on Communication, Control, and Computing*, September 1999.

[Por99] N. W. Porter, *Resources Required for Adaptive C4I Models in a Heterogeneous Computing Environment*, Master's Thesis, Dept. of CS, Naval Postgraduate School, Monterey, CA, June 1999 (D. A. Hensgen, advisor), pp. 181.

[RaL97] R. Rajkumar, C. Lee, J. P. Lehoczky, and D. P. Siewiorek, "A resource allocation model for QoS management," *IEEE Symposium on Real-Time Systems*, December 1997, pp. 289-307.

[Roc96] A. J. Rockmore, *BADD Functional Description*, Internal DARPA Memo, February 1996.

[RyN98] 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

[SaC97] 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*), February 1997, pp. 100-107.

[ScS98] P. A. Schneck and K. Schwan, "Dynamic authentication for high-performance networked applications," 6*th International Workshop on Quality of Service* (*IWQoS '98*), May 1998, pp. 127-136.

[ShW99] 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," in *Parallel and Distributed Processing: 11th IPPS/SPDP '99*, J. Rolim et al., eds., Springer, Berlin, April 1999, pp. 1335-1349.

[SiS82] 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.

[StS98] J. A. Stankovic, M. Supri, K. Ramamritham, and G. C. Buttazzo, "Terminology and assumptions," in *Deadline Scheduling for Real-Time Systems*, Kluwer Academic Publishers, Norwell MA, 1998, pp. 13-22.

[ThS01] Mitchell D. Theys, Howard Jay Siegel, and Edwin K. P. Chong, "Heuristics for scheduling data requests using collective communications in a distributed communication network," *Journal of Parallel and Distributed Computing*, Vol. 61, No. 9, pp. 1337-1366, Sep. 2001.

[ThT00] 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.

[WaW97] C. Wang and W. A. Wulf, "A framework for security measurement," *The National Information Systems Security Conference*, October 1997, pp. 522-533.

[WaW98] 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.

[XuN01] D. Xu, K. Nahrstedt, and D. Wichadakul, "QoS and contention-aware multi-resource reservation," *Cluster Computing*, Vol. 4, No. 2, April 2001, pp.95-107.