A QoS Performance Measure Framework for Distributed Heterogeneous Networks

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

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

[2]Naval Postgraduate School
CS and ECE Departments
Monterey, CA 93943, USA
{irvine, nwporter}@cs.nps.navy.mil
{hensgen, kidd}@acm.org

[3]Anteon Corporation
2600 Garden Rd., Ste. 220A
Monterey, CA 93940, USA
{stjohn, levin}@cs.nps.navy.mil

[4]University of Southern California
Department of EE-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

In a distributed heterogeneous computing environment, users' tasks are allocated resources to simultaneously satisfy, to varying degrees, the tasks' different, and possibly conflicting, quality of service (QoS) requirements. When 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 or no service at all. 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) ratio introduced here is a measure for quantifying this collective value. The FISC ratio is a multi-dimensional measure, and may include priorities, versions of a task or data, deadlines, situational mode, security, application- and domain-specific QoS, and dependencies. In addition to being used for evaluating and comparing RMSs, the FISC ratio can be incorporated as part of the objective function in a system's scheduling heuristics.

This research is 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 number DABT63-99-C-0010.

1. Introduction

In many distributed 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 provide a distributed suite of different types of machines, connected by diverse types of networks, to meet the varied computational and input requirements of such task mixtures (e.g., [4, 8, 18]). 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. Due to limitations on resource availability in certain situations, some tasks will receive degraded service or none at all. A performance measure is needed to quantify the collective value of the tasks completed during an interval of time, as perceived by the user, application, or policy maker. This measure can be a part of a metric to assess the success of an RMS (other parts may include execution time, ability to work with different operating systems, and user interface). This research describes attributes that can be combined to derive such a performance measure, and provides a way to combine them.

The Flexible Integrated System Capability (FISC) ratio is introduced in this research. It is a multi-dimensional performance measure, and may include priorities, versions of a task or data, deadlines, situational mode, security, application- and domain-specific QoS, and task dependencies. This FISC performance measure combines these attributes to determine the collective value of the tasks completed during an interval of time on a distributed computing system.

The FISC ratio can be used for the postmortem analysis of a system to determine the best scheduling heuristic for a given environment. It can also compare, via experimentation or simulation, the effectiveness of changing the resources available in a given distributed system. Furthermore, the FISC ratio can be incorporated as part of the objective function in a system's scheduling heuristics.

This research is part of three related DARPA activities: the DARPA Quorum-sponsored Management System for Heterogeneous Networks (MSHN) project [10], the DARPA Battlefield Awareness and Data Dissemination (BADD) program [22, 7], and the Agile Information Control Environment (AICE) program [1]. The goal of the Quorum MSHN project is to design, prototype, and refine a distributed RMS that leverages the heterogeneity of resources and tasks to deliver the requested QoS. In the Quorum environment, it is sometimes the case that not all tasks requested can achieve their most preferred QoS. 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 a set of requests in a way that has the greatest collective perceived value.

The performance measure described in this research is used to evaluate, for a given interval of time, the total value of tasks completed in the MSHN project and the total value of data received in the BADD and AICE programs. In this sense, the set of completed tasks for the MSHN project is equivalent to the set of satisfied data item requests for the BADD and AICE programs. A major difference between MSHN and BADD/AICE is that in MSHN tasks are assigned to resources by the RMS. In the BADD/AICE project, 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, task will be used to represent a user's process execution in the Quorum MSHN context, and a user's request for a data item in the BADD/AICE context. While this research is motivated by military applications, the FISC ratio can be adapted for other environments, such as industrial intra-nets, government agency intra-nets (e.g., NASA), and computational grids [9].

The next section provides a brief overview of some of the literature related to this work. In Section 3, several examples of individual QoS requirements are presented. These requirements 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. This section also presents a baseline for that measure and discusses a generalized form of the performance measure. An example of how this measure would be instantiated in a military C4I (command, control, communications, computers and intelligence) environment is provided in Section 5. The last section presents 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 scheduling and mapping of tasks, messages, and data items; in this section, some of the literature is described.

An optimistic priority-based concurrency control protocol that schedules active transactions with firm-deadline in real-time database systems is described in [12]. 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 research presented here includes other attributes that are important in evaluating the overall value of the tasks completed.

In [17], 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 [17], priority and deadline is used as measures of performance but the research effort does not consider other QoS attributes used in heterogeneous distributed networks.

An algorithm that allows transmission of messages belonging to several classes of situational mode is presented in [20]. The algorithm takes into account the actual priority of a message in a given class. This algorithm rejects packets with deadlines shorter than a minimum acceptance deadline defined for a particular class. There can be more than one simple deadline. This and other important QoS attributes are discussed in detail.

Data staging, an important data management problem for a distributed heterogeneous networking environment, is discussed in [28]. This was done under the assumption that each requested data item is associated with a specific deadline and priority. The research presented here generalizes part of the objective function used in [28] to include more types of deadlines and other QoS attributes.

From the works mentioned, parameters such as task priority and deadline appear to be important attributes for making scheduling decisions regarding tasks, messages, or data items. A measure of the overall value is needed that can be used in an objective function to compare and analyze the algorithms, protocols, and heuristics while incorporating all the QoS parameters used. The works mentioned above consider only a few of the QoS parameters that are being used 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 this research.

The research on a 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 requested QoS is taken into account when scheduling computational resources in a network is presented in [19]. The model proposed a benefit function that uses application deadlines and application priorities as metrics in maximizing the total benefit for the applications. Multiple versions of a task in addition to priorities and deadlines in the objective function are described in [14]. The FISC performance measure presented in this paper serves a purpose similar to the ones in [5, 19, 14]. 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 an effective schedule in the Quorum MSHN and BADD/AICE environments. Furthermore, the QoS input specification for ERDoS [24] 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 FISC ratio is similar to the utility function described in [31] in that both measure the system value. The utility function in [31] calculates the system value by summing the value of resources allocated and resources reserved depending on the duration of a job, the deadline of a job, and the price of allocated time slots. The FISC ratio uses a conceptually similar calculation, but formulates it differently and includes priorities and other QoS measures to calculate the collective value of task completed.

A security policy that allows a percentage of packets authenticated to vary with network load is described in [25]. This type of policy can be accommodated with the variant components included in the FISC security vector (see Subsection 3.5). While the FISC security vector contains a set of Boolean security policy statements, it does not specify a general-purpose language for these statements. Related work on network security policy specification languages can be found in [2, 3], and works in progress [6, 23]. A framework for quantifying the strength of a set of security mechanisms is described in [30], where high-level static security properties can be decomposed hierarchically. However, in [30] the approach cannot accommodate the measurement of how well an executed task meets the security requirements of its environment. Nor does [30] account for variant security policies or mechanisms.

The FISC ratio measures the effectiveness of a schedule in terms of tasks completed and percentage of their requirements satisfied. In [26], other types of attributes for evaluating 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 outside the scope of the FISC performance measure, which focuses on the collective worth of the completed tasks.

3. Example QoS Attributes

3.1. Overview

Examples of attributes that need to be considered in the FISC performance measure will be described in this section. The attributes discussed include user-based and application-based priority levels, user preferences for different versions of a task, deadlines, security, other application- and domain-specific QoS attributes, and task dependencies. The attributes used in any given situation must have interpretations that are meaningful to the users, policy makers, and the RMS. A method of determining the weightings, worths, functions, and factors are described in this section.

3.2. Priorities

Policy makers determine the number of priority levels and assign some semantic meaning to each priority level, such that each level qualitatively reflects the relative importance (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. Such a scheme, therefore, cannot be used alone in the measure. A more meaningful weight must instead be assigned to each priority level so that the 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 the 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 the war mode, one high priority level task might be more important than 1000 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 ratio is being used to compute the value of a subset of tasks successfully completed, during some time interval, from a set of $\underline{t}$ tasks that have been requested. Let the priority level (e.g., high, medium, low) of task j ($0 \leq j < t$) be $\underline{p_j}$, and let $\underline{m}$ be the situational mode. The priority weight function $\underline{\pi(p_j, m)}$ calculates the weight of p_j given m. The weight assigned to a priority level may be considered to be the maximum value of completing the corresponding task, if all of the task's specified QoS requirements are completely satisfied.

3.3. 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. 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 [24], in the sense that each version of a task may have different accuracy and precision.

For each version of a given task, in a given situational mode, a worth (preference) relative to the other versions will be indicated by the application developer, the user, and/or the policy makers. 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.

	worth			normalized worth		
Task \ version	0	1	2	0	1	2
0	1	1	8	.125	.125	1
1	25	35	40	.625	.875	1
2	.2	.3	.5	.4	.6	1
3	.1	.2	.7	.143	.286	1

Table 1: Worths and normalized worths that indicate preference for each version of a task.

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. Therefore without a normalization procedure, a version with the smallest worth of a certain task can always be chosen for processing ahead of other tasks for no logical reason. For example, the version 0 of task 1 that is the version with the lowest worth of task 1 would be chosen over the version 2 of task 0 that is the version with the highest worth for task 0. In extreme cases, worths that are not normalized could make the priority irrelevant depending on how priorities and worths of version interact.

To avoid this type of anomalous behavior, worths are normalized as follows. Assume there are $\underline{I_j}$ number of versions for each task *j*. Let $\underline{v_{ij}}$ be the $\underline{i}$-th ($0 \leq i < I_j$) version of task *j*. Let $\underline{w_{ij}(m)}$ 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. 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. The normalized worth ($\underline{\eta_{ij}}$) of $w_{ij}(m)$ is then given by

$$\eta_{ij} = w_{ij}(m) \Big/ \max_{0 \leq a < I_j} w_{aj}(m). \tag{1}$$

As shown in Table 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 relative to the version with the largest worth. This is the reason why the indicated worths for each task need not sum to one.

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 would allow a greedy person to obtain a higher value for his/her preference for the version with the largest worth. For example, consider task 0 and task 1 in Table 1. If this alternative approach is used, the normalized worth for task 0 would be .1, .1, and .8, while for task 1 it would be .25, .35, and .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. In extreme cases, priorities may be irrelevant depending on how priorities and worth of versions interact.

3.4. 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 ($\underline{f_{ij}^d}$). These three deadlines are illustrated by example in Figure 1. The deadline attribute discussed here is related to the timeliness parameter in [24].

The earliest deadline (e_{ij}^d) is the time when the system is ready to complete a task, based upon, for example, the availability of the required input data. Therefore, if a task completes before the earliest deadline the task has 0 value.

The soft deadline (s_{ij}^d) is the time by which a task must complete to be of full value to the user [27]. 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 [15, 27]. 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.

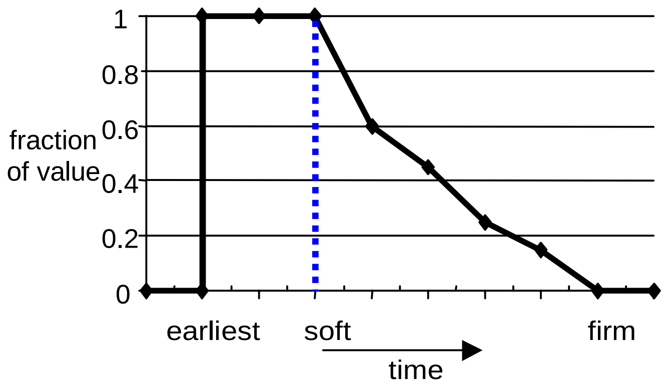

Figure 1: The deadline coefficient graph (non-increasing) shows the variation in the value of a task with various deadlines.

Let $\underline{\tau_{ij}}$ be the time that the *i*-th version of task *j* actually completes. The deadline function $\underline{\delta_{ij}(\tau_{ij}, m)}$ 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 $0 \leq \delta_{ij}(\tau_{ij}, m) \leq 1$. If no version of a task is completed, $\delta_{ij}(\tau_{ij}, m) = 0$ for all versions of the task. The deadlines may be the same for all versions of a certain task.

3.5. 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 ($\underline{S}$) 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 [11].

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 an ordered set of resources utilized by v_{ij} and let $\underline{S}_{ij}$ be a sub-vector of vector *S*. A component $\underline{c}$ in *S* is in S_{ij} if and only if *c* depends on v_{ij} or on an element of R_{ij}, and is denoted $\underline{S_{ij}.c}$. Let $\underline{Z}_{ij}$ be 1 if the instantiated Boolean value of all *c* in S_{ij} is true and 0 otherwise. Z_{ij} corresponds to the required security attributes. This means that if minimum security requirements are not met, then no benefit is accrued from executing v_{ij}.

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. Thus, additional benefit is accrued if increased security is selected within the range.

The desire to provide *adaptable* security motivates the inclusion of variant security components in the system [16]. 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 $\underline{[S_{ij}.c]}$ be the number of quanta in $S_{ij}.c$ (in the above case this is five) and $\underline{g_{ij}.c}$ be the fraction of *c* in S_{ij} satisfied. If a task achieves the third quantum (70%), then $g_{ij}.c$ is $3/[S_{ij}.c] = 3/5 = 0.6$. Suppose $\underline{n}$ is the number of security components in S_{ij}. To quantify the effectiveness of the RMS in providing variant security, let $\underline{A}_{ij}$ be the sum of all $g_{ij}.c$ divided by *n* as shown in Equation 2.

$$A_{ij} = \Big(\sum_{c \in S_{ij}} g_{ij}.c\Big)/n \qquad (2)$$

The above is just one possible way to combine the values of these security components. For example, the $g_{ij}.c$ values in A_{ij} equation can have relative weightings for given *m*. Thus, if the military situation changes from peace to war, encryption rate would be considered relatively more important and might be given a high relative weighting.

The overall <u>security</u> <u>factor</u> is defined as: $\underline{\sigma}_{ij} = A_{ij} \times Z_{ij}$, where $0 \le \sigma_{ij} \le 1$. It indicates how the value of a task may be degraded due to lack of its most desirable security services.

3.6. Other Application- & Domain-Specific QoS

There is a multi-dimensional space of application- and domain-specific QoS attributes (e.g., jitter level, frame rate, 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 ratio will use $\underline{\alpha}_{ij}$ to denote such QoS components, analogous to the security factor σ_{ij} [13].

3.7. Dependencies

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. If there is a descendant along a path of tasks that does more than just generate data for a subsequent task 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. If there is no descendant task that produces a result important to a user, all predecessors are worthless to the user.

The first task in a path that does more than generate data for a subsequent task will be known as a <u>required associate</u> of all of its predecessors. This is one way of determining the value of ascendants that only generate data for subsequent tasks. The variable $\underline{\rho}_{ij}$ 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 $\rho_{ij} = 0$. For the BADD/AICE environment, a similar definition of required associates can be established based on the set of data requested by a given application.

4. Performance Measure

4.1. FISC Ratio

The question of what it means to "provide good QoS" to a mixture of applications in a distributed system is considered in this section. In general, it is a difficult problem to determine whether a distributed system has delivered good service to a mixture of applications with different priorities, receiving different degraded level of QoS etc. A meaningful way to combine the QoS attributes previously discussed is motivated and proposed in this section.

First, consider a set of tasks, possibly with different priorities and different deadlines, where the firm and soft deadlines are identical. Assume also that each task only has a single version. Therefore, the initiators of a task need not specify a preference for a version. It is obvious that a system with sufficient resources will complete all tasks by their deadlines. However, if resources are insufficient and tasks cannot complete by their deadlines, then the RMS will complete the tasks with the higher priorities by their deadline. Such an RMS maximizes

$$\sum_{j=0}^{t-1} \pi(p_j, m) \times \delta_j(\tau_j, m), \tag{3}$$

where $\delta_j(\tau_j, m)$ is 1 if the task is completed by its deadline and 0 otherwise. Even when the deadline is more general, as described in Subsection 3.4, it is easy to see that the same criterion should be maximized.

Now consider the same situation, except with tasks having multiple versions. Again, if there are sufficient resources to complete the most preferred versions of each task, then an RMS will allocate resources and initiate those most preferred versions. However, if resources are insufficient for completing the most preferred versions of all tasks, the RMS must consider using less preferred versions for some tasks. Typically, this will result in a reduction of both the value of completing the task and the amount of the resource consumed. Thus, for the resources available the RMS should maximize

$$\sum_{j=0}^{t-1} \pi(p_j, m) \times [\max_{0 \le i < I_i} \delta_{ij}(\tau_{ij}, m) \times \eta_{ij})]. \tag{4}$$

If only one version of a task is completed, δ_{ij} for all other versions would be 0. Therefore, the max function would be used to indicate whether a version of a task is completed within its deadline.

Similarly, the RMS must consider any reductions in value of a task's completion that is a result of not receiving all required and desired security, and other application- and domain-specific QoS requirements. Furthermore, any dependencies must be obeyed for a task's completion to have any value. One way to accomplish this is to multiply Equation 4 by ρ_{ij}, σ_{ij}, and α_{ij}:

$$\sum_{j=0}^{t-1} \pi(p_j, m) \times [\max_{0 \le i < I_i} [\eta_{ij} \times \rho_{ij} \times \delta_{ij}(\tau_{ij}, m) \times \sigma_{ij} \times \alpha_{ij}]]. \tag{5}$$

However, the measure shown above makes it difficult to compare one RMS, operating within one distributed system, to another RMS operating in a different distributed system. To allow this kind of comparison, the equation

$$\frac{\sum_{j=0}^{t-1} \pi(p_j, m) \times [\max_{0 \le i < I_i} [\eta_{ij} \times \rho_{ij} \times \delta_{ij}(\tau_{ij}, m) \times \sigma_{ij} \times \alpha_{ij}]]}{\text{baseline}} \tag{6}$$

normalizes the collective value of the completed tasks by the results from some baseline, which depends on the tasks and underlying distributed system. This will be discussed further in Subsection 4.2.

Recall the goal of the FISC measure is to determine the performance of a schedule (mapping) for tasks in an oversubscribed distributed system by calculating the collective value of the tasks completed. This measure can also be used as a critical part of an objective function of a scheduling heuristic. Components in addition to the FISC measure may be useful in the determining of the objective function. For example, in [29] the objective function that is used includes expected time between when the data request will be satisfied and its deadline (i.e., urgency).

4.2. Baseline

This section provides a baseline for the FISC ratio. The purpose of the baseline in the FISC ratio is to determine how an RMS performs compared to another RMS in a given 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 [29]. The algorithm used to compute the baseline uses the concept of perfect completion. A task is said to achieve perfect completion if there exists available resources, to which it can be assigned, that would allow it to complete with $\eta_{ij} = \delta_j = \sigma_{ij} = \alpha_{ij} = 100\%$ and $\rho = 1$.

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 is shown in Figure 2 and it proceeds as follows. First, it assigns an ordering to the tasks according to their priorities, deadlines, and expected execution times 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.

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

if task can get $\eta_{ij} = \delta_{ij} = \alpha_{ij} = \sigma_{ij} = 100\%$
and $\rho_{ij} = 1$
    schedule
    add $\pi(p_j, m)$
    update status of resources
else
    no value added
    no resources consumed
```

Figure 2: Baseline algorithm.

4.3. Generalization

The previous subsection describes one instantiation of the FISC ratio. It can be generalized such that the numerator is any function of $\pi(p_j, m)$, η_{ij}, ρ_{ij}, $\delta_j(\tau_{ij}, m)$, 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 3.5). Let P_r be a primary factor where there can be u number of primary factors ($0 \le r \le u-1$) and s_e be a secondary factor where there can be v_r number of secondary factors ($0 \le e \le v_r-1$). The generalization of FISC ratio can be represented as

$$FISC = \mathrm{f}(P_0, P_1, \dots, P_{u-1}) / \text{baseline and} \qquad (7)$$

$$P_r = \mathrm{f}_r(s_0, s_1, \dots, s_{v_r-1}),$$

where each s_e is a secondary factor for P_r. Linear or nonlinear weightings 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 described is one method of normalizing the numerator of the FISC ratio. Other methods for normalizing could be incorporated to compare the performance of different RMSs in a given environment.

5. Examples of FISC Ratio Use

As an example of how the FISC ratio 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 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) [21].

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). The deadline is firm, with the simulation results required within an hour. If received within an hour the results will achieve some fraction of their maximum worth. After that time, they receive 0 value. The stochastic version is preferred because it will produce higher confidence results, but the deterministic simulation may also be useful. The stochastic version is assigned a preference of eight, on a scale of ten, while the deterministic version is assigned a preference of five. Security level in this case is binary. The information must be sent over a secure link. If it is, a version is assigned a security value of 1, if not, it is assigned a security value of 0. If only one of the two versions can be completed, and these are the only ones to choose from, then the stochastic version will be completed because it will give a higher value than the deterministic version. This is a simple case; usually other factors have to be considered (e.g., expected execution time) when scheduling tasks.

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 wise decision by maximizing an objective function where the FISC ratio would be a major component. While this example is from a military environment, the FISC ratio can be adapted for other environments as well.

6. Summary and Future Work

The FISC ratio provides a way to quantify the value of the performance received by a set of applications in a distributed system. Thus, it can be used to evaluate the effectiveness of the mapping of a collection of requests to resources done by a scheduler. In addition, it may be used in a simulation mode to analyze the impact of proposed changes to the distributed system. Therefore, 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 ratio may be used as a critical part of a scheduling heuristic's objective function. A generalization of the ratio is also discussed in this research. Additional issues that may be considered in future research include weighting the relative importance of the π, η, ρ, δ, σ, and α factors, using a non-linear combination of task values to compute the overall measure, and using of negative fractions in the deadline function in case of catastrophic results from a missed deadline could be incorporated.

Acknowledgments: The authors thank B. Beaton, G. Koob, J. Rockmore, E. Chong, M. Jurczyk, I-J. Wang, S. Jones, T. D. Braun, N. Beck, S. Ali, S. Chatterjea, A. Naik, P. Dharwadkar, J. Kresho, and N. Rowe for their valuable comments.

References

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

[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 Symp. Security and Privacy*, May 1995, pp. 66-77.

[3] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," *1996 IEEE Symp. 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, and B. Yao, "A taxonomy for describing matching and scheduling heuristics for mixed-machine heterogeneous computing systems," *17th IEEE Symp. on Reliable Distributed Systems*, Oct. 1998, pp. 330-335.

[5] S. Chatterjee, B. Sabata, and J. Sydir, *ERDoS QoS Architecture*, ITAD-1667-TR-98-075, SRI, Menlo Park, CA, 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/internetdrafts/ draft-ietf-ipsec-spsl-00.txt.

[7] DARPA, Battlefield Awareness and Data Disse-mination, Apr. 1999, www.darpa.mil/iso/ badd/.

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

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

[10] 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.

[11] C. E. Irvine and T. Levin, "Toward a taxonomy and costing method for security services," *15th Annual Computer Security Applications Conf.* (*ACSAC '99*), Dec. 1999, to appear.

[12] 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.

[13] J. K. Kim, D. A. Hensgen, T. Kidd, H. J. Siegel, D. St. John, C. Irvine, T. Levin, N. W. Porter, V. K. Prasanna, and R. F. Freund, *A Multi-Dimensional QoS Performance Measure for Distributed Heterogeneous Networks*, ECE School, Purdue, technical report in preparation.

[14] J. P. Kresho, *Quality Network Load Information Improves Performance of Adaptive Applications*, Master’s Thesis, Dept. of Computer Science, Naval Postgraduate School, Monterey, CA, Sep. 1997 (D. A. Hensgen, advisor).

[15] 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 Int’l Workshop on Real-Time Computing Systems and Applications*, Oct./Nov. 1996, pp. 74-80.

[16] T. Levin and C. Irvine, *Quality of Security Service in a Resource Management System Benefit Function*, Technical Report, Computer Science Dept., Naval Postgraduate School, to appear.

[17] J. P. Li and M. W. Mutka, “Priority based real-time communication for large scale wormhole networks,” *8th Int’l Parallel Processing Symp***.**, Apr. 1994, pp. 433-438.

[18] M. Maheswaran, T. D. Braun, and H. J. Siegel, “High-performance mixed-machine heterogeneous computing,” *6th Euromicro Workshop on Parallel and Distributed Processing*, Jan. 1998, pp. 3-9.

[19] M. Maheswaran, “Quality of service driven resource management algorithms for network computing,*” Int’l Conf. on Parallel and Distributed Processing Techniques and Applications* (*PDPTA’99*), June/July 1999, pp. 1090-1096.

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

[21] 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, to appear (D. A. Hensgen, advisor).

[22] A. J. Rockmore, *BADD Functional Description*, Internal DARPA Memo, Feb. 1996.

[23] T. Ryutov and C. Neuman, “Access control framework for distributed applications,” *INTERNET-DRAFT*, CAT Working Group, Nov. 1998, ftp://ftp.ietf.org/internet-drafts/draft-ietf-cat-acc-cntrl-frmw-01.txt.

[24] B. Sabata, S. Chatterjee, M. Davis, J. Sydir, and T. Lawrence, “Taxonomy for QoS specifications,” *3rd Int’l Workshop on Object-Oriented Real-Time Dependable Systems* (*WORDS ‘97*), Feb. 1997, pp. 100-107.

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

[26] B. Shirazi, L. Welch, B. Ravindran, C. Cavanaugh, B. Yanamula, R. Brucks, E. Huh, “DynBench: a dynamic benchmark suite for distributed real-time systems,” in *Parallel and Distributed Processing: IPPS/SPDP Workshops*, J. Rolim et al., eds., Springer, Berlin, Apr. 1999, pp. 1335-1349.

[27] 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.

[28] M. Tan, M. D. Theys, H. J. Siegel, N. B. Beck, and M. Jurczyk, “A mathematical model, heuristic, and simulation study for a basic data staging problem in a heterogeneous networking environment,” *7th Heterogeneous Computing Workshop* (*HCW ’98*), Mar. 1998, pp. 115-129.

[29] M. D. Theys, M. Tan, N. B. Beck, H. J. Siegel, and M. Jurczyk, *Heuristics and a Mathematical Framework for Scheduling Data Requests in a Distributed Communication Network*, TR-ECE 99-2, ECE School, Purdue, Jan. 1999, 58 pp.

[30] C. Wang and W. A. Wulf, “A framework for security measurement,” *The Nat’l Information Systems Security Conf***.**, Oct. 1997, pp. 522-533.

[31] W.E. Walsh, M. P. Wellman, P. R. Wurman, and J. K. Mackie-Mason, “Some economics of market-based distributed scheduling,” *18th Int’l Conf. on Distributed Computer Systems*, May 1998, PP. 612-621.