*In Proceedings of 2th IASTED European Conference on Parallel and Distributed Systems, pp 224--254*

# Determining the Accuracy Required in Resource Load Prediction to Successfully Support Application Agility *

John Kresho
Debra Hensgen
Taylor Kidd
Geoffrey Xie

Computer Science Department
Naval Postgraduate School
Monterey, CA 93940

Corresponding Author: Debra Hensgen
hensgen@cs.nps.navy.mil
(408)656-4074 (voice)
(408)656-2814 (fax)

April 22, 1998

## Abstract

*Largely due to the proliferation of the World Wide Web, and interfaces such as Netscape, users expect to have many different types of information immediately available. When they encounter a lengthy delay caused by heavy loads on shared resources, such as networks or servers, users often (manually) adapt by requesting different forms of the same information. As both mobile and agent computing becomes more popular, users will expect their applications to automatically adapt to heavy resource loads by fetching the information in a different form, e.g., text instead of graphics. This paper studies the accuracy with which resource loading information, particularly network loading information, must be known in order for applications to successfully, and with agility, adapt. We determine that under many normal conditions, fairly inaccurate estimates of currently available bandwidth suffice. However, when the system is heavily loaded, some strategies can perform much better with very accurate load estimates. That is, assuming that the adaptive applications have hard deadlines for obtaining the data they request, up to 20% more of them will receive some form of that data on time, if the adaptation strategy has a good estimate of available bandwidth. Additionally, in these situations, applications that have a better estimate of bandwidth can deliver, on average, larger sized messages corresponding to, in many cases, higher quality data. Finally, the accuracy with which the bandwidth must be known varies not only with inter-arrival rate, but also with the adaptation strategy used and the percentage of adaptive applications in the system.*

## 1 Introduction

Most of today's applications do not automatically adapt their resource needs to changing computational environments. For example, an Internet browsing application may attempt to load a 5 MByte web page whether the expected time to obtain that web page is 10 minutes (lightly loaded network) or 1 hour (heavily loaded network). In such an application, the burden is placed on the user to manually stop the loading of the current web page and request, for example, similar information in a different format, perhaps without

*This research was supported by DARPA under contract number E583. Additional support was provided by the Naval Postgraduate School and the Institute for Joint Warfare Analysis.

graphics, in order to obtain the desired information in a reasonable amount of time. In this paper, we investigate support for applications that automatically adapt to resource availability by reacting to estimates of loads on those resources. In particular, we investigate, through simulation, how accurate the predictions of resource loads, specifically loads on the network, must be in order for adaptive clients to obtain good performance.

Using simulation, we examine the performance of three different client adaptation strategies. One is a control strategy wherein the client does not adapt, and the other two adapt by making use of resource loading information of varying quality. Our clients set deadlines, after which the requested information is not needed. We use a mix of adaptive and non-adaptive clients and vary the load they place on the network. In one of our adaptive strategies, all clients, not just the adaptive clients, stop sending information if the information does not arrive before its deadline.

In the body of this paper, we first describe the general problem on which we are working, then identify the specific part of this problem that we concentrate on in this paper. In our related work section, we compare this work to other ongoing work in the field of support for adaptive applications and we describe a particular existing service implementation, known as a Communications Server, that supports adaptation by estimating the current network load. We describe in that section how far the Communications Server's estimates were from the actual network load in our experiments. We use the results from these experiments as input for some of our simulations. We then describe our simulation model. Our model includes our adaptation strategies, our simulation parameters, and our methodology. We then present our results. Finally, after briefly describing our current work on an architecture to support adaptive applications, we summarize our conclusions.

## 2 The Problem

This paper is the first in a series of papers about the software architecture for our Management System for Heterogeneous Networks(MSHN). MSHN leverages our experience in designing, building, and experimenting with SmartNet, which is a scheduling framework for heterogeneous computing [7]. MSHN is an extensible software architecture that is layered on top of native operating systems and network protocols. MSHN is responsible for determining which large scale resources, such as compute servers and file servers, should be used to execute particular requests, and shepherding the solutions to those requests. It is also responsible for determining which version of an *adaptable application* should be used to respond to a request.

MSHN uses a model of the underlying resources, and the systems that manage those resources, to determine both which resources should be used and which version of an application to execute. It uses estimates of resource requirements, which it also maintains, as well as estimates of resource availability to make its determinations. Unlike SmartNet, it does not assume that it has control over all applications using the resources.

One example of an adaptive application is one that is capable of receiving and displaying data in many different formats. Another example is an application that has many different synchronization formats, different ones of which would be more appropriate under various resource and loading environments. Still another example is an application used by someone who would prefer today's weather forecast, but who would accept yesterday's. We expect adaptive client applications, usually using MSHN's libraries, to query MSHN servers to determine the current status of resources. Adaptive applications would then begin to use an application format that was well-suited to the available resources, but might need to adjust their decision if higher priority applications need the resources.

This paper documents our investigation into one aspect of MSHN. The particular problem that this paper deals with is determining how accurate an estimate of resource availability is necessary in order to implement a successful strategy for adapting. Knowing how accurate an estimate is needed will help us when we refine MSHN's architecture, because it will help us trade off the amount of time required to get better estimates against the performance benefits that applications will receive from such estimates.

## 3 Relationship to Other Work on Adaptive Systems

There are many systems that are currently exploring adaptivity as a response to resource load changes in the areas of high performance computing, real-time network protocols, and in systems that support mobile computing.

Siegel [1] investigates adaptation by reconfiguring a SPMD application into a SIMD application. Researchers at UCSD are examining general approaches to reconfiguring applications based upon the types and numbers of processors available [2].

Xie [9] has developed a new network architecture, called Burst Scheduling, to guarantee the real-time quality of service needs for multimedia applications when the loads on the networks are continually changing. RLM [11] examines receiver initiated adaptation for network applications that execute in a heterogeneous multicast environment.

Several researchers are investigating support for mobile applications [12], [6], [3]. In fact, the term "agility," used in the title of this paper, is defined by Satyanarayanan when describing the Odyssey system, as the speed and accuracy with which an adaptive system detects and responds to changes in resource availability. There are two major differences between Odyssey and MSHN: (i) Odyssey is aimed at mobile applications whereas MSHN is not; and (ii) Odyssey does not consider supporting applications with hard deadlines, whereas many of MSHN's applications, if not completed by their deadlines, are useless. Additionally, most of these systems, have, as a key component some mechanisms for encapsulating data typing information. Finally, QuO [18] tackles the software engineering problem of building applications that are capable of adapting.

The previous work most closely related to this paper is design work on the JTF-ATD Communications Server. The Communications Server is one of the servers provided by the the Defense Advanced Research Projects Agency's (DARPA) Joint Task Force Advanced Technology Architecture [4, 5]. It is the job of the Communications Server to predict the current network loads. The Communications Server obtains its estimates by actively sending different sized packets over the network and recording the amount of time required for a round trip. Using this timing information, it estimates the average bandwidth that is available, along with the latency for packets sent along the network. When queried, the Communications Server reports the instantaneous reading, without anchoring it through the use of historical data. In addition to placing an additional load on a possibly heavily loaded resource, the Communications Server, then, is also subject to inaccuracies caused by the bursty nature of network traffic. Along this same line, Wolski's Network Weather Service [17, 16] implements a method for monitoring current network use and CPU load, using historical information to predict future use and load. Like the Communications Server, whose accuracy we report on in this paper, also estimate bandwidth availability by sending different sized data sets over the network and recording the time required for the transfer.

Since this paper examines the performance of adaptation policies, as a function of the accuracy of bandwidth predictions, we include the expected accuracy of the Communications Server as one data point in our simulations. In order to mimic the Communications Server accuracy in these simulations, we ran several file transfer experiments. While the Communications Server monitored the bandwidth between two computers, we transferred files and retrieved readings of the Communications Server's predictions. We plotted the difference between the actual average bandwidth and the Communications Server's predicted bandwidth and found it to be very close to an exponential distribution with a mean of 1Mbit, offset by 340Kbits.

# 4 Our Simulation Model

We built discrete event simulations of communication-intensive applications that exchange information with one another over a fully-connected network. In this section, we enumerate the values that are fixed in our simulations and describe the parameters that we varied. Before enumerating these values and parameters, we define several terms:

**Node.** A location that contains many computers that generate network traffic, such as a business office.

**Client.** An application that generates its own messages. There are typically many clients at a single node.

**Best Case Latency.** The amount of time it would take for a message to arrive if it could use all of the bandwidth on a channel. We will denote the best case latency as $L_b$.

The assumptions we use below are similar to those used in simulations of an initial Communications Server design performed by Teknowledge Federal Systems [15]. The characteristics of the simulation experiments that we will describe reflect the following scenario. Two sites were used, one afloat and the other ashore. Three classes of slightly modified application clients were considered: a time-critical electronic mail system, a time-critical database client, and a time-critical web client. In each case, if data is not received on time, lives may be at stake. Similarly, two of the situations used, when we generate messages on average every two and every three seconds, we consider to be crisis situations. Such a crisis situation might exist if there is a weather emergency (e.g., a tsunami) in the Pacific and decisions to evacuate civilians must be made correctly and quickly. We are all familiar with the fact that in such emergencies phone lines become inaccessible due to overuse, if not due to weather conditions. The same will soon be true for network resources. It is exactly in these emergencies that we wish to ensure that

- the highest priority information is delivered on time, and that
- the users of the applications receive sufficient information to make wise decisions.

### 4.1 Characteristics Common to all of our Simulation Experiments

Given the definitions above, our simulation models a network with the following properties.

- Our simple network has two nodes. No routing is therefore needed; all messages are sent over direct connections.
- The connection between the two nodes is full duplex with a throughput of 10 Mbits/second. Half of the network's bandwidth, 5 Mbits/second, is available for each direction.
- Each client using the network receives an equal share of the bandwidth.

During the simulation, non-adaptive clients generate **ordinary messages** according to the inter-arrival distribution associated with that particular simulation. Ordinary messages differ from **adaptable messages** in that ordinary messages are available in only a single format. The adaptable messages are generated using the same inter-arrival distribution. All messages have the following attributes.

- A **priority**, $P$, that ranges between 0 (high priority) and 9 (low priority). We generate the priority using a uniform distribution.
- The **tolerated latency** is the amount of time that the application is willing to wait for the data to arrive, before it considers it to be late. The **deadline** is derived by adding the current time to the tolerated latency. As might be expected, we set up the experiment such that the higher priority messages have smaller tolerated latencies; that is, the higher priority messages needed to arrive at their destination sooner.

  For (ordinary) messages from non-adaptive applications, we set the tolerated latency to:

$$\begin{array}{rl} 16 \times L_b & \text{if } P \in \{8, 9\} \\ 12 \times L_b & \text{if } P \in \{5, 6, 7\} \\ 10 \times L_b & \text{if } P \in \{2, 3, 4\} \\ 7 \times L_b & \text{if } P \in \{0, 1\} \end{array}$$

  The tolerated latencies for adaptive messages are chosen from a uniform discrete distribution of $\{L_b$ + 30 minutes, $L_b$ + 60 minutes, $L_b$ + 90 minutes$\}$. As these tolerated latencies ultimately represent the maximum amount of time available to get some required information to a human decision maker, say some commander in the field, they are highly arbitrary. Depending upon the decision being made, the tolerated latency can range anywhere from seconds to days. The distribution used above was selected as "reasonable" given a tactical planning scenario.

- Non-adaptive clients send ordinary messages that have different lengths depending upon their class.
  - Class A messages are exponentially distributed around 1 MByte and are generated 60% of the time;
  - Class B messages are exponentially distributed around 1.4 MBytes and are generated 25% of the time; and
  - Class C messages are exponentially distributed around 50 MBytes and are generated 15% of the time.

Finally, we ran each experiment using 10 different sets of random seeds. In each set, a different seed was used for each of the following distributions:

- the inter-arrival rate,
- the message class type generation,
- the distribution that determined whether a client was adaptive or non-adaptive,
- each different message size distribution, and
- the priority.

Also, we verified that, in every case, we ran our simulation long enough to reach a stable state. This is especially important because in some of our experiments, the network was very heavily loaded due to a small average inter-arrival time.

### 4.2 Adaptation Strategies

In different simulations, we varied the percentage of *adaptive clients* between 1.25% and 100% of all clients. Non-adaptive clients exchange data that is available only in a single format. On the other hand, all data that an adaptive client needs to send is available in any of five formats. The actual sizes for each of the five data formats are chosen from exponential distributions with means 3000 MBytes (8 minutes of color video), 300 MBytes, 30 MBytes, 3 MBytes, and .3 MBytes (simple text description). We assume that our adaptive clients' preference for the various formats decreases with size. That is, the most important format for each client to exchange is the largest one and the smallest format is of least importance.

We ran our simulations using three different client adaptation strategies. In the first strategy, **Strategy 1**, the client first requests a performance prediction from the server; that is, it requests that the server respond with its current estimate of the available bandwidth of the network. The client[1] then calculates whether, based on this predicted bandwidth, it should be able to transmit the largest size format of the required data. If the calculation indicates that this format can be transmitted in its entirety before its deadline, the client begins to send the data. If the client's calculation indicates that this transmission cannot be completed before the deadline, the client iterates through the various size formats from larger to smaller, until it determines the largest one that the client can expect to send in its entirety prior to that deadline, and begins to send it. Periodically the server updates the client with new estimates of available bandwidth. If, based upon a new estimate, the client calculates that it cannot complete sending the format that it is currently transmitting prior to its deadline, it stops transmitting that format and searches for a smaller format that it can expect to complete prior to the deadline and begins transmitting that format. [2]

**Strategy 2** is very similar to Strategy 1. The only difference is that in Strategy 2, both adaptive and non-adaptive clients take action if their deadline arrives and they have not completed transmitting their current format; they stop transmitting when the deadline arrives. In Strategy 1, such "late" formats were sent to completion, despite the deadline having passed. Strategy 1 seems to be a very inconsiderate strategy. However, similar strategies, known as "application centric" strategies, are used in many high performance applications [2].

**Strategy 3** acts as a control. In this strategy, the client does not really adapt. It always attempts to send the largest format and continues sending it until that format has been transmitted in its entirety. We note that this strategy, though seemingly more wasteful even than Strategy 1, is the default strategy in most Internet web servers.

### 4.3 Simulation Parameters

In this section we identify our simulation parameters and how we varied them for different simulations. The majority of our experimental space (Figure 1) is focused on the three parameters that we discuss first.

We ran different simulation experiments for different average inter-arrival rates. In each simulation, the amount of time between client message generation is exponentially distributed around the mean. The means for the different experiments are set at 2 seconds, 3 seconds, 15 seconds, and 60 seconds. Given the amount of data being sent in our experiments, the 2- and 3-second average inter-arrival times

[1] In this paper, we attribute much of the work of adapting to the client application. However, in the architecture that we are currently building most of the adaptation work is to be done via common libraries that are linked with the client. In our architecture, the client simply supplies the list of acceptable formats along with a ranking indicating its preference for each format.

[2] The adaptive clients in our current simulation do not ever start sending a larger format.

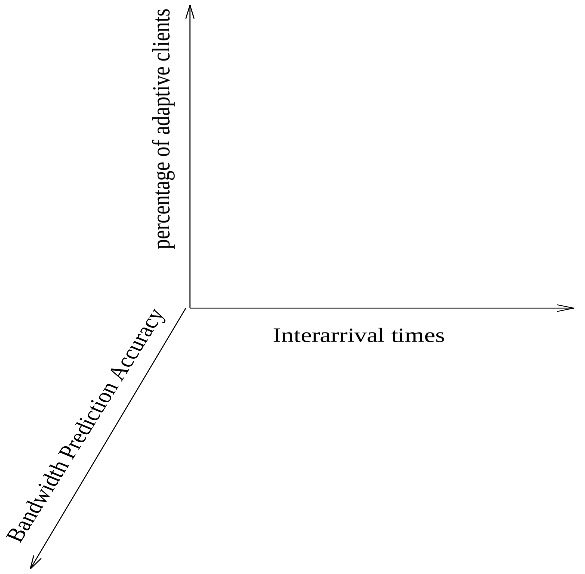

Figure 1: Experimental space using the parameters of inter-arrival times, percentage of adaptive clients, and accuracy of bandwidth estimates.

simulate a critical or busy mode while the 15- and 60-second mean inter-arrival times simulate a lightly loaded mode.

In each experiment, some of our client applications are adaptive, while the remainder are non-adaptive. We ran some experiments where all of the clients (100%) are adaptive and others where very few (1.25%) are adaptive. We also examined some ratios in between these extreems: 20%, 10%, 5%, and 2.5%.

We ran different experiments varying the accuracy with which our server could predict the instantaneously available bandwidth. Since we are running a simulation, instantaneous and exact values of the bandwidth are available. To simulate reality, we added "noise." This noise was sampled from exponential distributions with various different means. Since an exponential distribution produces only positive numbers, we changed the sign of the noise by multiplying it by a sample taken from the discrete uniform distribution {-1,1}, i.e., we flipped a coin. We call this skewed bandwidth estimate the *Instantaneous prediction*. We ran experiments using means, measured in Kbits/second, of 0, 2.5, 5, 7.5, 10, 20, and 50 with the (.85,.15) and (.15,.85) weights described below.

Also, because network traffic is bursty, and because, like all good control systems, we wanted to avoid unstable situations, we had our simulated server use two different sets of **weights** when producing its bandwidth estimation. In the first case it used $Prediction_i = 0.15 \times Instantaneous$ $prediction + 0.85 \times Prediction_{i-1}$, where $Prediction_i$ was the estimate used to determine whether the current format could be completed. We also performed experiments using $Prediction_i = 0.85 \times$ $Instantaneous\ prediction + 0.15 \times Prediction_{i-1}$. When we discuss results pertaining to using these different weights in Section 5.2, we will denote them as the weights $(x, y)$, where $x$ refers to the weighing of the $Instantaneousprediction$ and $y$, to the weighting of the estimate $Prediction_{i-1}$. We note that the Communications Server uses a weighting pair of (1.0, 0.0), which we use for generating data values that correspond to the performance of a server such as the Communications Server.

# 5 Results

In this section we present results from our simulations and summarize our conclusions from these results. For each experiment, we measured both the average size of the adaptive messages that arrived before their deadline and the percentage of adaptive messages for which no format arrived on time.

## 5.1 The Need for Adaptation

We first present results that demonstrate the need for adaptation. After these results and our conclusions from them, we restrict our attention to only Strategies 1 and 2 (Section 4.2). As mentioned above, Strategy 3 was our control case wherein adaptive applications simply tried to send the largest format of each data item.

In these experiments, 5% of the processes were adaptive; that is, they attempted to deliver a message of a size drawn from an exponential distribution with mean 3000 MBytes. For all inter-arrival times (means of 2 seconds, 3 seconds, 15 seconds, and 60 seconds), 98% of these large messages did not arrive before their deadline, even when we removed messages immediately if they exceeded their deadlines. In order for these applications to meet critical deadlines, it is apparent that both a method of estimating the network resource load and a strategy for adapting to bandwidth availability is needed.

## 5.2 The Effect of Varying Weights

In this section, we show that using the weight pair (.15, .85) is substantially better than the pair (.85, .15). In later sections we restrict our discussion to experiments involving only the weight pair (.15, .85).

Using Strategy 1, we again ran simulations where 5% of the messages were adaptive. Table 1 shows results from these simulations. The weighting scheme (.15,.85) is much better when the messages have an inter-arrival mean time of 2 and 3 seconds, and slightly better than (.85, .15) for the 15- and 60-second mean inter-arrival times. The difference between the weighting schemes is a matter of reaction time. Using (.85, .15), adaptive applications will tend to react more quickly to an instantaneous reading which can cause resource thrashing, especially in a heavily loaded environment. On the other hand, the (.15, .85) scheme allows adaptive applications to make better informed decisions based on statistics gathered over a period of time. Ideally, one would want to dynamically adjust these weights depending upon the observed behavior of the data being sampled. That is, the weights would dynamically change based upon detection that the underlying statistics of the sample data is changing. Techniques used to build filters that accomplish this dynamic weighting can be found in Singer [13], LeMay and Brogan [10], and Sworder [14]. While a real system would likely use such filters, we chose to keep our simulation simple and therefore we ran the rest of our simulations, excluding those that mimicked the Communications Server, using the (.15,.85) weighting scheme. As mentioned earlier our simulations that mimicked the Communications Server were executed using the (1.0, 0) weighting because that server does not anchor its predictions with historic information.

| Inter Arrival Time (secs) | Size for (.85, .15) Weighting (KBytes) | Size for (.15, .85) Weighting (KBytes) |
|---|---|---|
| 2 | 761 | 1271 |
| 3 | 1833 | 2546 |
| 15 | 45793 | 45929 |
| 60 | 297269 | 299379 |

Table 1: Average size of messages received using different weighting schemes under Strategy 1 (5% applications adaptive).

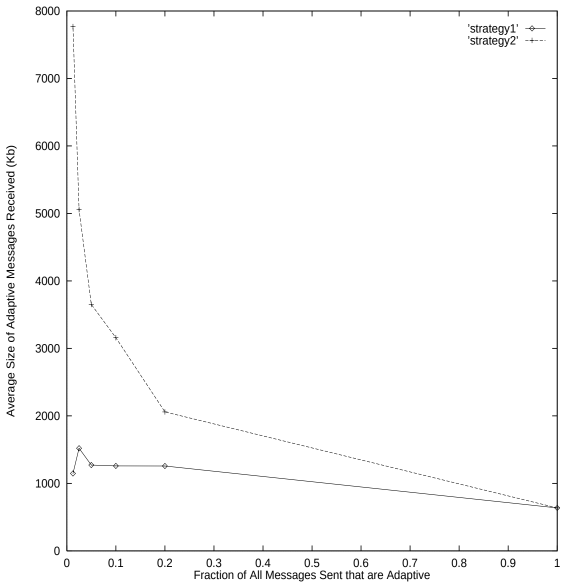

Figure 2: Average size of adaptive messages received for an inter-arrival time of 2 seconds.

## 5.3 Strategy 1 vs. Strategy 2

In this section, we see that there is some benefit to be gained from dropping messages that exceed their deadline. In comparing these two strategies, we use an *Instantaneous prediction* with a mean error rate of 5.0 Kbits.

Figure 2 shows that when the network resource is very busy (mean inter-arrival time of 2 seconds), the benefit of dropping messages that exceed their deadline is substantial.[3] The results for a 3 second inter-arrival time are very similar.

When the network resource becomes less loaded, there is less benefit from dropping late messages. This is due to the fact that there are fewer messages that are late, and hence eligible to be dropped, because the network resource is not in high demand. Figure 3 shows the results for mean inter-arrival time of 15 seconds. When the mean inter-arrival time is 60 seconds, applications receive sufficient bandwidth the

[3] At this point we note that in Figure 2 which we have already discussed, as well as in both Figure 3 and Figure 4 which we will soon discuss, it may at first appear that performance is decreasing as the fraction of adaptive applications increases. Actually, these graphs must be carefully considered. As the fraction of adaptive applications increases, so does the amount of traffic that the applications collectively attempt to place on the network, since the average size of a message from an adaptive application is larger than the average size of a message from a non-adaptive application.

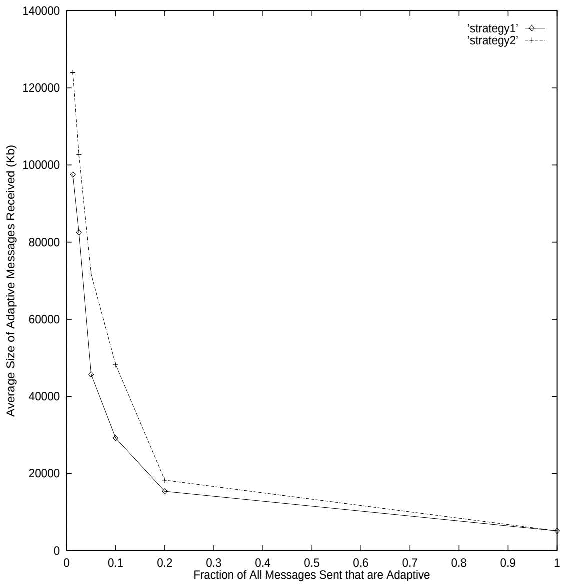

Figure 3: Average size of adaptive messages received for a mean inter-arrival time of 15 seconds.

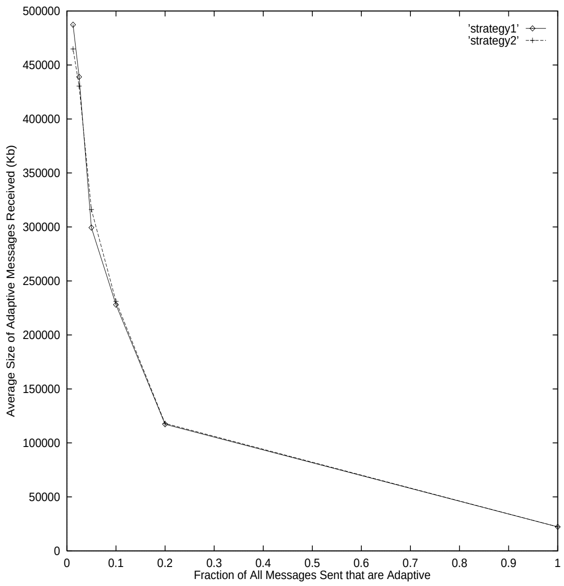

Figure 4: Average size of adaptive messages received for an inter-arrival time of 60 seconds.

majority of the time. Figure 4 demonstrates that in this case there is no benefit derived from, nor penalty paid for, dropping late messages.

### 5.4 Determining How Accurate Server Estimates Should Be

In this section we examine a multitude of operating points to determine under what conditions:

1. A simple server, such as the JTF-ATD Communications Server, will suffice, and
2. When a more accurate assessment of resource load is needed.

As we stated in the previous section, we performed experiments with the JTF-ATD Communications Server, wherein we recorded the Communications Server's *estimate* of latency based upon its intrusive occasional messages. At the same time as the Communications Server was "snooping" on the network, we ran applications that sent round-trip messages and as accurately as possible measured the *actual* latency and bandwidth. We found that the difference between the estimated and the actual bandwidth, that is, the error in the estimate, very closely approximated an exponential distribution with mean of 1 Mbit, offset 340 Kbits in the negative direction [8]. We therefore used this distribution to predict adaptive performance based upon advice from this intrusive server. The data points in the graph labeled Communications Server correspond to this performance.

Before discussing actual results for different accuracies of server estimates, we note that simulations that use 2 and 3 seconds as their mean message inter-arrival time model a crisis situation. In a military environment, such inter-arrival rates occur in an emergency, such as during a sudden biological attack. In this case, the network resource will be in high demand, but the priority messages must make it to their destinations before their deadlines. However, when the mean inter-arrival times are 15 and 60 seconds, the network resource is under normal use, and not many applications are competing for the same network resource.

In order to determine how accurate a server must be when estimating a resource's load, we collected data for each of the *Instantaneous prediction* error means enumerated above. The first criteria analyzed was the number of messages that did not make their deadlines under Strategies 1 and 2. Figures 5 and  6 display the results of Strategy 2 when there are 100% and 1.25% adaptive messages, respectively. The results showing the number of late messages for Strategy 2 show a trend similar to that of Strategy 1. We note that the points labeled "Communications Server" model the accuracy of the JTF ATD Communications Server as discussed above.

For each of the *Instantaneous prediction* means, including the Communications Server, there were no

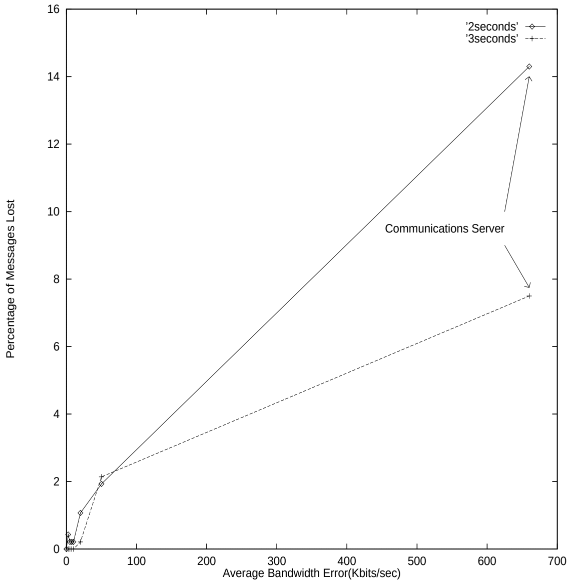

Figure 5: Percentage of messages not received by deadline when using Strategy 2 and 100% of messages are adaptive.

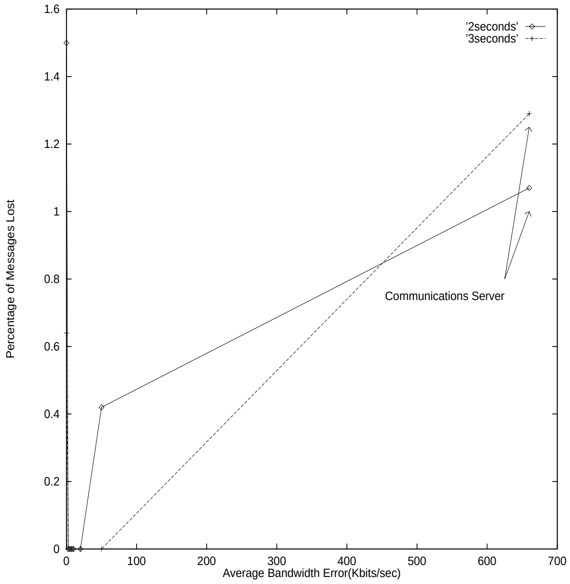

Figure 6: Percentage of messages not received by deadline using Strategy 2 and 1.25% of messages are adaptive.

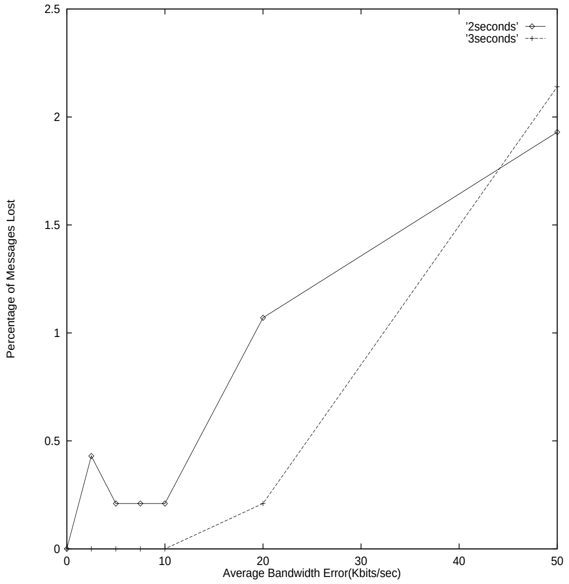

Figure 7: Percentage of messages not received by deadline using Strategy 2 and 100% of messages are adaptive.

late messages for mean inter-arrival times of 15 and 60 seconds. Therefore, we now focus on the crisis situations (2 and 3 second mean inter-arrival times) and determine how accurate a server's prediction must be. Figure 7 and Figure 8 eliminate the Communications Server data and focus on more accurate assessments. Again, the results for Strategy 1 show similar trends. Before continuing further, we wish to point out an interesting phenomena that is seen on close examination of these figures: the small spike at 2.5 KB/sec error in Figure 7 and the downward slope prior to 2.5 KB/sec error in Figure 8. These phenomena result from our applications being too optimistic initially. When many adaptive applications decide almost simultaneously, to send a message, and they have an accurate assessment of the available bandwidth they may all attempt to send very large formats. When they notice that the other applications are also sending large formats, they throttle back, but sometimes it is too late, because they have already put so much data on the network that they prevent one another from completely transmitting any size of message.

After examining these results closely, we determined that, in a crisis situation, being within 20 Kbits/second of the actual network throughput allows most messages to meet their deadlines. Using a less accurate server may result in an unacceptable loss of data, possibly meaning loss of life in some ap-

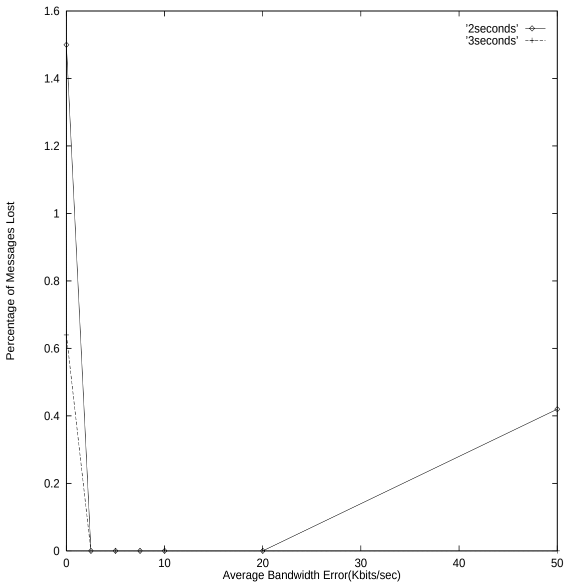

Figure 8: Percentage of messages not received by deadline using Strategy 2 and 1.25% of messages are adaptive.

plications.

The second criteria we examine is the average size of the message that does arrive on time. Figure 9 shows the results for Strategy 2 when 100% of the applications are adaptive. We note that when the server estimates are less accurate, only smaller messages are successfully received. We observed this trend for all inter-arrival times for these simulations.

In order to better understand the circumstances under which the average size received is maximized, we refer to Figure 10. The figure indicates that being within 5 Kbits/second will get the best results in a crisis situation under most loads and being within 10 Kbits/second may suffice in many circumstances. The only exception to this rule is seen when only 1.25% of the applications are adaptive, and the mean inter-arrival rate is 60 seconds. Figure 11 shows that, in this case, a server such as the Communications Server will allow for larger adaptive messages to arrive on time. Since there is little competition for the network resource, a less accurate picture of the load is acceptable. Overall though, as Figure 12 shows, when a crisis situation occurs, it is better to have an accurate server, one that can predict the network bandwidth within 10 Kbits/second. The results for Strategy 1 again are similar to the results presented here.

As can be seen from the results, most cases require an accurate estimate of the network resource

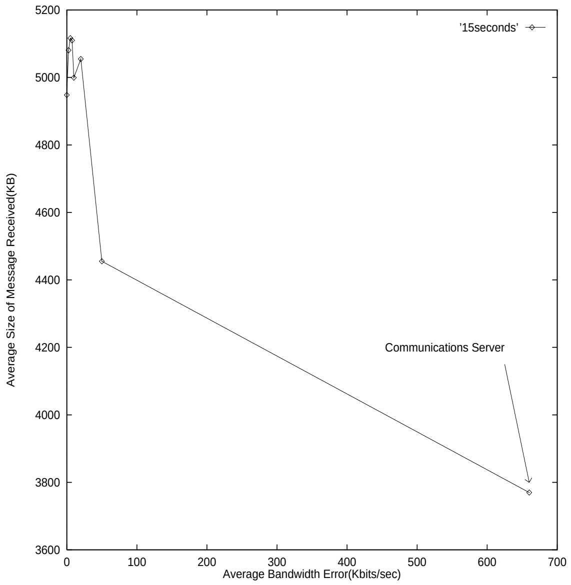

Figure 9: Average size of successful adaptive messages using Strategy 2 when 100% of the messages are adaptive and the mean inter-arrival time is 15 seconds.

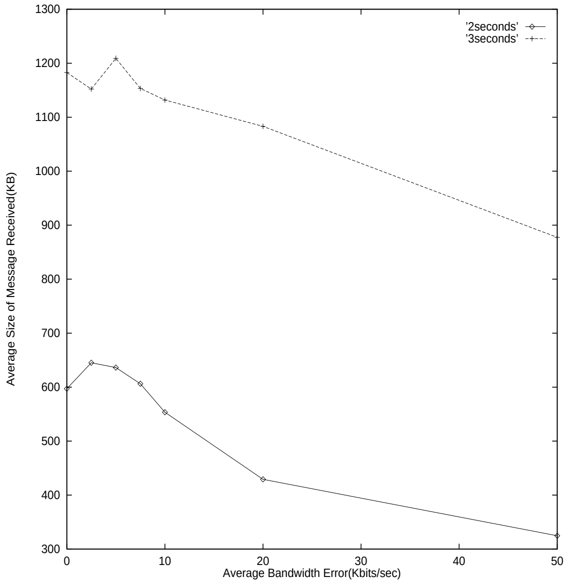

Figure 10: Average size of successful adaptive messages using Strategy 2 when 100% of the messages are adaptive and the mean inter-arrival times are 2 and 3 seconds.

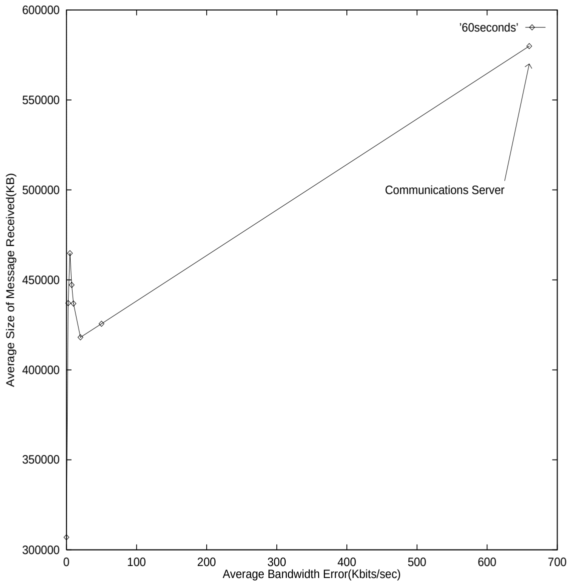

Figure 11: Average size of successful adaptive messages using Strategy 2 when 1.25% of the messages are adaptive and the mean inter-arrival time is 60 seconds.

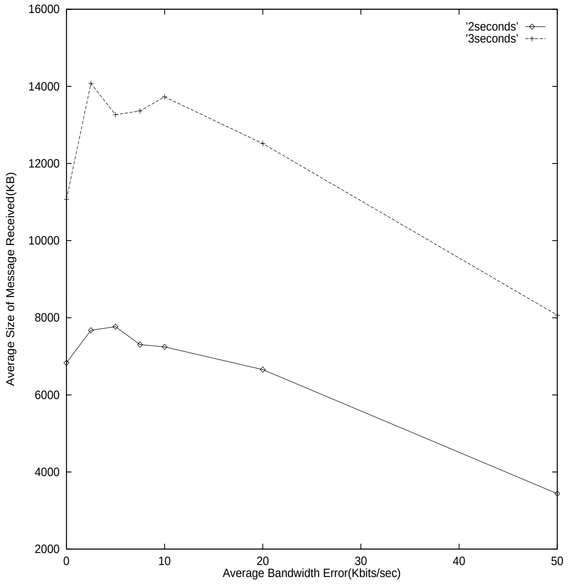

Figure 12: Average size of successful adaptive messages using Strategy 2 when 1.25% of the messages are adaptive and the mean inter-arrival times are 2 and 3 seconds.

(within 10 Kbits/second), especially in a crisis mode. However, when there is little competition for the resource and the percentage of adaptive applications was small, a less accurate estimate may be useful.

# 6 Current Work

In addition to expanding the portion of the parameter space covered by our simulations for the problem described in this paper, we are currently in the process of refining both the generalized scheduling problem that MSHN addresses as well as our software architecture. The scheduling problem, for which MSHN provides support, is a stochastic pseudo-Boolean programming problem. MSHN's initial architecture is made up of wrappers for system calls; an adaptation client library; two database supporting servers, one maintaining a quickly changing database and another maintaining a fairly rapidly changing database; and a SmartNet-like scheduling advisor. Further description of the MSHN architecture is beyond the scope of this paper and will be documented elsewhere.

# 7 Conclusions

In this paper we saw that, for situations similar to those examined by Teknowledge, an adaptive client requires a better network resource load assessment than can be furnished by an intrusive server that occasionally examines the state of the network. Specifically, estimating the network resource within 10 Kbits/second allows applications to adapt very well in most cases and in some cases an estimate within 20 Kbits/second suffices. While we make no claim that these results apply for situations with very different parameters than we examined in this paper, we chose these parameters both because they represented realistic scenarios and because they allowed us to compare various adaptation strategies under conditions which had already been examined by others against those same strategies when more information is available.

Finally, this paper only focuses on estimating the network resource load, but other resources, such as CPU and hard drive use, must also be monitored. In order to allow easy development of adaptive applications, an architecture that provides these applications an interface to accurate information, such as that presented in this paper, is essential.

## References

[1] Armstrong, J. B., Siegel, H. J., Cohen, W., Tan, M., Dietz, H. G., and Fortes, J. A. B. Dynamic Task Migration from SPMD to SIMD Virtual Machines. In *Proceedings of the International Conference on Parallel Processing* (August 1994), vol. 2, pp. 160–169.

[2] Berman, F., and Wolski, R. Scheduling From the Perspective of the Application. In *High Performance Distributed Computing(HPDC '96)* (August 1996).

[3] Fox, A., Gribble, S. D., Brewer, E. A., Amir, E., Dietz, H. G., and Fortes, J. A. B. Adapting to Network and Client Variability via On-Demand Dynamic Distillation. In *Proceedings of the Seventh International ACM Conference on Architectural Support for Programming Languages and Operating Systems* (October 1996), pp. 160–170.

[4] Hayes-Roth, F. The JTF-ATD Architecture for Agile, Mobile, Collaborative Crisis Response: A Case Study in Architecture Driven Development. Teknowledge Federal Systems Technical Report, available at http://www.teknowledge.com, March 1995.

[5] Hayes-Roth, F., and Erman, L. Joint Task Force Architecture Specification (JTFAS). Teknowledge Federal Systems Technical Report, available at http://www.teknowledge.com, April 1994.

[6] Inouye, J., Cen, S., Pu, C., and Walpole, J. System Support for Mobile Multimedia Applications. In *Proceedings of the 7th International Workshop on Network and Operating Systems Support for Digital Audio and Video* (May 1997), pp. 143–154.

[7] Kidd, T., Hensgen, D., Freund, R., and Moore, L. Smartnet: A Scheduling Framework for Heterogeneous Computing. *Proceedings of the IEEE International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'96)* (June 1996), 514–521.

[8] Kresho, J. Quality Network Load Information Improves Performance of Adaptive Applications. Naval Postgraduate School Technical Report, Master's Thesis, September 1997.

[9] Lam, S., and Xie, G. Burst Scheduling Networks: Flow Specification and Performance Guarantees. In *Proceedings of the Workshop on Network and Operating Systems Support for Digital Video and Audio* (April 1995), pp. 289–292.

[10] LeMay, L. J., and Brogan, W. L. Kalman Filtering: Extended Kalman Filter. St. Joseph Sciences, Inc., 1984.

[11] McCanne, S., Jacobson, V., and Vetterli, M. Receiver-driven Layered Multicast. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM) '96 Conference* (August 1996), pp. 117–130.

[12] Noble, B. D., Satyanarayanan, M., Narayanan, D., Tilton, J. E., and Walker, K. R. Agile Application Aware Adaptation for Mobility. In *Proceedings of the 16th ACM Symposium on Operating System Principles* (October 1997).

[13] Singer, R. A. Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets. *IEEE Transactions on Aerospace and Electronic Systems AES-6*, 4 (July 1970), 473–483.

[14] Sworder, D. D., Clapp, J., and Kidd, T. Model-Based Prediction of Decisionmaker Delays and Uncertainty. *Cybernetics and Systems 24*, 1 (January 1993), 51–68.

[15] Terry, A., Barnes, T., and Hayes-Roth, R. JTF ATD Communications Server: Architectural Refinement Through Simulation Experiments. Teknowledge Federal Systems Technical Report, available at http://www.teknowledge.com, September 1995.

[16] Wolski, R. Forecasting network performance to support dynamic scheduling using the network weather service. In *Proceedings of the 6th High-Performance Distributed Computing Conference* (August 1997).

[17] Wolski, R., Spring, N., and Peterson, C. Implementing a performance forecasting system for metacomputing: The network weather service. Tech. Rep. TR-CS97-540, UCSD, May 1997.

[18] Zinky, J. A., Bakken, D. E., and Schantz, R. D. Architectural Support for Quality of Service for CORBA Objects. *Theory and Practice of Object Systems 3* (January 1997).