A Network Status Predictor to Support Dynamic Scheduling in Network-Based Computing Systems*

JunSeong Kim and David J. Lilja
Electrical and Computer Engineering and Minnesota Supercomputing Institute
University of Minnesota, Minneapolis, MN 55455

Abstract

The management of networks has often been ignored in network-based computing systems due to the difficulty of estimating application programs' network latency and bandwidth requirements, and the difficulty of predicting the system network load. To help address this deficiency, and thereby support dynamic network resource scheduling, we propose the Network Status Predictor (NSP). *This tool is a general and extensible network load monitor that introduces lower and upper latency prediction bounds. We evaluate its ability to dynamically predict TCP/IP end-to-end latency with varying network loads using a cluster of SGI multiprocessors interconnected with a Fibre Channel network. Our results show that a combination of numerical predictors can be dynamically selected based on the network's recent state to produce better predictions than when using a single predictor alone.*

1. Introduction

The importance of reducing communication overhead in network-based computing cannot be overemphasized since communication delays can often become the performance bottleneck in parallel application programs [3, 7, 10]. While substantial research efforts have focused on improving various aspects of the communication performance of network-based systems [2, 3, 5, 7, 8, 10], network resources typically have not been actively scheduled. However, understanding dynamic variations in both network bandwidth and latency is critical to maximize an application's overall performance and resource utilization.

This paper presents an efficient, accurate, and extensible network load detection and prediction method that can be used for dynamically scheduling network resources in a distributed parallel computing system. The *Network Status Predictor (NSP)* tool directly monitors network load status by continually measuring network latencies. Several different numerical models are evaluated for their ability to predict lower and upper bounds for the network load, which is more useful information than a single predicted latency value. Of course, we could achieve perfect prediction accuracy simply by making the predicted interval very large. A large interval is essentially useless for allocating network resources, however. Very small intervals, on the other hand, tend to be inaccurate in the sense that the actual value that occurs is outside the predicted interval.

To quantify this tradeoff, we introduce two new performance metrics. The *prediction accuracy* is the ratio of the number of correct predictions (i.e. the latency value that actually occurs is within the predicted interval) to the total number of predictions. The *prediction width* then quantifies the aggressiveness of a prediction method by normalizing the difference between the upper and lower bounds to the difference of the maximum and minimum values of a given window of latency measurement history.

2. Network Status Predictor

The *Network Status Predictor (NSP)* is a distributed system that measures current network load to predict future load. When using network latency history to predict future latencies, older measured values are likely to be less important for predicting future latencies than more recent values. To accommodate this changing level of importance in measurements, *NSP* uses a *sliding window* mechanism that saves only the previous K latency measurements.

Let $\mathsf{Swin}_K(i,t)$, $0 \leq i \leq (K-1)$, be the $(i+1)^{th}$ element in the window at time t. Then the average, standard deviation, and median of the window are calculated as follows.

$$\mathsf{avg}_K(t) \quad = \quad \frac{1}{K} \sum_{i=0}^{K-1} \mathsf{Swin}_K(i,t)$$

*Supported in part by National Science Foundation grant no. CDA-9414015. Authors' e-mail addresses: {jskim,lilja}@ece.umn.edu.

$$\mathsf{stdev_K}(t) = \sqrt{\frac{1}{K}\sum_{i=0}^{K-1}(\mathsf{Swin_K}(i,t) - \mathsf{avg_K}(t))^2}$$

$$\mathsf{med_K}(t) = \begin{cases} \mathsf{Sort_K}((K-1)/2, t) & , K = odd \\ \frac{\mathsf{Sort_K}(K/2-1,t)+\mathsf{Sort_K}(K/2,t)}{2} & , K = even \end{cases}$$

where $\mathsf{Sort_K}(i,t)$ is the $(i+1)^{th}$ element of the sorted sequence of the previous K measurements in ascending order.

2.1. Prediction Bounds and Metrics

Bounds for predicting future network latencies are calculated using the sliding window of measured network latency values. That is, at time t we calculate lower and upper prediction bounds, $LB_f(t)$ and $UB_f(t)$, respectively, such that

$$LB_f(t) = \mathsf{Method}_f(value(t), value(t-1), ..., value(0))$$
$$UB_f(t) = \mathsf{Method}_f(value(t), value(t-1), ..., value(0)).$$

In these expressions, $value(t)$ is the measured latency value at time t. Then the actual network latency value for time $(t+1)$ is $value(t+1)$ where Method_f predicted that $LB_f(t) \leq value(t+1) \leq UB_f(t)$.

To test the effectiveness of a prediction method we introduce the *prediction accuracy* and *prediction width*. A prediction is correct when the actual measured value at time $(t+1)$ is between the prediction bounds $LB_f(t)$ and $UB_f(t)$. The *prediction accuracy*, *PA*, is defined to be the ratio of the number of correct predictions to the total number of predictions:

$$PA_f(n) = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions}.$$

We could always ensure a correct prediction at time t by choosing $LB_f(t) = 0$ and $UB_f(t) = \infty$. Very wide intervals are not particularly useful when trying to estimate future network latency values, however. Smaller intervals provide more precise information, and so would be preferred. However, there is an obvious tradeoff between the interval width and the prediction accuracy. To quantify the usefulness and aggressiveness of a prediction method, we introduce the *prediction width* metric, *PW*. This metric is defined to be the average of the ratios of the difference between the lower and upper bounds of the predictions to the difference between the minimum and maximum values among all of the elements in the sliding window. Thus,

$$PW_f(n) = \frac{1}{n}\sum_{t=0}^{n-1}\left[\frac{UB_f(t) - LB_f(t)}{\mathsf{Max_K}(t) - \mathsf{Min_K}(t)}\right]$$

Using these metrics, the predictor with the combination of highest prediction accuracy and smallest prediction width is the best predictor of future network latencies.

2.2. Prediction Models

MIN-MAX based prediction

The most straight-forward prediction bounds use the minimum and maximum values of the sliding window directly for the lower and upper prediction bounds, respectively, giving

$$LB_{MM}(K,t) = \mathsf{Min_K}(t)$$
$$UB_{MM}(K,t) = \mathsf{Max_K}(t)$$

MIN-MAX prediction is expected to show good prediction accuracy since the latency at time $(t+1)$ is likely to be between the minimum and maximum of the previous K measured latency values. In practice, however, the MIN-MAX prediction is too conservative in that the bounding interval becomes too large to be useful. This method is a good reference for the other methods, though, due to its high prediction accuracy and its reflection of the measurement history.

A simple modification of the MIN-MAX prediction method is a *trimmed MIN-MAX* prediction. Instead of considering all of the window elements, we consider only the central $(K-2T)$ elements from the sorted list ignoring the largest and the smallest group of values. Trimmed MIN-MAX prediction compensates for the drawback of the complete MIN-MAX predictor by eliminating the window's extreme values from consideration for the prediction bounds. The trimmed MIN-MAX predictor that ignores $2T$ elements in the window is defined to be

$$LB_{TMM}(K,T,t) = \mathsf{Sort_K}(T,t)$$
$$UB_{TMM}(K,T,t) = \mathsf{Sort_K}(K-T-1,t)$$

where $T = \lfloor \alpha * K \rfloor$ and $0 \leq \alpha < 1$ is the fraction of the elements in the window that are ignored.

Standard deviation based prediction

It seems reasonable to take the average distance from the mean (i.e. the standard deviation) for the prediction bounds, especially when the distribution is unimodal and symmetric. This gives

$$LB_{STD}(K,t) = \mathsf{Max}[0, \mathsf{avg_K}(t) - \mathsf{stdev_K}(t)]$$
$$UB_{STD}(K,t) = \mathsf{avg_K}(t) + \mathsf{stdev_K}(t)$$

Note that we make the lower bound always non-negative with the Max operation.

While the mean value provides equal weight to all the elements within the sliding window, the median value may be more representative when the events tend to show great variability with an asymmetric distribution [11]. Consequently, our alternative standard deviation prediction method uses

the median value instead of the mean value as a reference point, giving

$$\begin{aligned} LB_{MSTD}(K,t) &= \mathsf{Max}[0, \mathsf{med_K}(t) - \mathsf{stdev_K}(t)] \\ UB_{MSTD}(K,t) &= \mathsf{med_K}(t) + \mathsf{stdev_K}(t) \end{aligned}$$

Confidence interval based prediction

Another predictor we considered is based on confidence intervals, which assume that the measured values come from a normal distribution. Let $T = t_{[1-\alpha/2;K-1]}\mathsf{stdev_K}(t)/\sqrt{K}$ where $t_{[1-\alpha/2;K-1]}$ is the $(1-\alpha/2)$-quantile of a t-variate with $(K-1)$-degrees of freedom. Then, the $100(1-\alpha)\%$ confidence interval for the prediction bounds is given by

$$\begin{aligned} LB_{CI}(K,\alpha,t) &= \mathsf{Max}[0, \mathsf{avg_K}(t) - T] \\ UB_{CI}(K,\alpha,t) &= \mathsf{avg_K}(t) + T \end{aligned}$$

As we have in the standard deviation prediction, we also examine the use of the median value as the basis in the confidence interval prediction giving

$$\begin{aligned} LB_{MCI}(K,\alpha,t) &= \mathsf{Max}[0, \mathsf{med_K}(t) - T] \\ UB_{MCI}(K,\alpha,t) &= \mathsf{med_K}(t) + T \end{aligned}$$

Weight based prediction

The last method we consider for calculating prediction bounds applies a fixed weight to the mean and median values of the window elements. While we expect good results when samples are uniformly distributed, it is based on no special theory. Instead, its attractiveness is its simplicity. The prediction bounds when using the mean are

$$\begin{aligned} LB_{WT}(K,\alpha,t) &= \mathsf{avg_K}(t) * (1-\alpha) \\ UB_{WT}(K,\alpha,t) &= \mathsf{avg_K}(t) * (1+\alpha) \end{aligned}$$

and for the median are

$$\begin{aligned} LB_{MWT}(K,\alpha,t) &= \mathsf{med_K}(t) * (1-\alpha) \\ UB_{MWT}(K,\alpha,t) &= \mathsf{med_K}(t) * (1+\alpha) \end{aligned}$$

where $0 \leq \alpha < 1$.

Figure 1 shows an example of the bounds calculated when using the various prediction methods described above. These methods provide bounds for the network latency predicted to occur at time $(t+1)$. For instance, the standard deviation method predicts that the latency that will occur at time $(t+1)$ is between LB_{STD}=1.7 msec and UB_{STD}=6.9 msec.

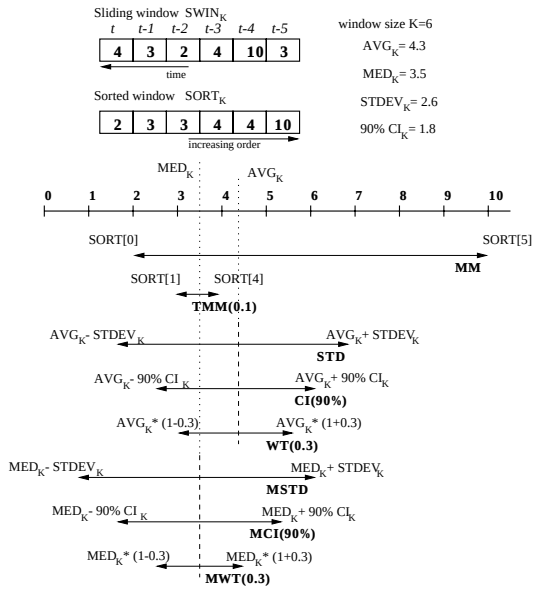

Figure 1. The various methods evaluated for calculating the prediction bounds in *NSP* can produce a wide range of prediction widths.

3. Experimental Results

3.1. Methodology

A distributed cluster of four Silicon Graphics Challenge L shared-memory multiprocessors was used to evaluate the various *NSP* latency prediction methods. Each of the nodes in this system contains at least three R10000 processors running at 196MHz on a shared bus. All nodes run version 6.2 of the IRIX operating system and communicate with each other over a Fibre Channel network using an Ancor CXT 250 16-port switch. We measured the TCP/IP round trip delay between nodes over a 24-hour period. Figure 2 shows a 24-hour sample with the X-axis representing the elapsed time from 10:00 a.m. on Thursday, June 4, 1998 to 10:00 a.m. the following day sampled at 10 second intervals. The Y-axis is the round trip delay of an $8K$ byte sampling message sent between two fixed nodes.

By comparing the actual latency value at time $(t+1)$ with the prediction bounds of the various predictors, we can evaluate the prediction ability of each method. Note that each predictor uses as its input K consecutive latency values from time t to $(t-K-1)$ of the 24-hour sample to predict the latency at time $(t+1)$. These predictions are calculated for each value in the 24-hour sample to compute the prediction accuracies and widths in the following sections.

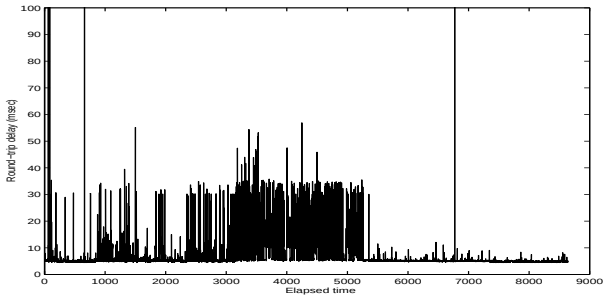

Figure 2. The round trip time required to send an $8K$ byte message between two processors was measured every 10 seconds for a 24-hour period.

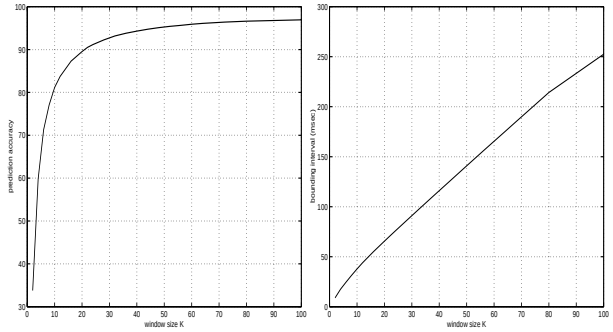

Figure 3. The prediction accuracy and bounding interval of a MIN-MAX predictor as a function of the sliding window size, K.

3.2. Baseline MIN-MAX Prediction

Figure 3 shows the prediction accuracy and bounding interval as a function of the sliding window size, K, for the MIN-MAX predictor. The *bounding interval* is the average difference between the lower and upper bounds. As expected, the prediction accuracy improves as the window size K increases since the maximum value of a larger window is not less than that of a smaller window and the minimum value of a larger window is not greater than that of a smaller window. However, the more accurate prediction is at the cost of a larger bounding interval.

While the prediction accuracy saturates near 97%, the bounding width continues to increase as K increases. With a window size of $K = 20$, we can expect a prediction accuracy as high as about 90% with a corresponding bounding interval of 65.51 msec. Reducing the window size to $K = 10$ reduces the expected prediction accuracy to 81.2%, while increasing the window to $K = 30$ causes the bounding interval to increase by nearly a factor of 50% while increasing prediction accuracy by only 3.6%. Thus, we use a window size of $K = 20$ in these experiments to find a method whose prediction accuracy is close to that of this MIN-MAX method, but whose prediction width is as small as possible.

3.3. Comparing the Predictors

Figure 4 compares the prediction accuracy and prediction width of the different prediction methods discussed in Section 2.2 when applied over the 24 hours of sample data. The trimmed MIN-MAX method (TMM) uses three different parameter settings, α=0.05, 0.1, and 0.15, which results in 2, 4, and 6 elements being trimmed from the window, respectively. The confidence interval methods (CI and MCI) are calculated for 80%, 90%, and 95% intervals. Finally, the weight based methods (WT and MWT) employ weights of α=0.1, 0.2, and 0.3.

The simple MIN-MAX method (MM) shows the best prediction accuracy of 89.5%. Since it is used as the prediction width normalization basis, its prediction width is always 100%. Surprisingly, the modified weight method (MWT) shows the second best prediction accuracy. When α=0.3 it shows 82.6% prediction accuracy, but with a 93.9% higher prediction width than that of the MIN-MAX method. The standard deviation based methods (STD and MSTD) have around 77% prediction accuracy with only 51% of the prediction width of the MIN-MAX method. From the low prediction accuracies of the confidence interval based methods (CI and MCI), we conclude that the assumption of network latency following a normal distribution is not valid.

3.4. A Dynamic Composition Predictor

To exploit the best behavior of the different predictors for a wide range of network traffic situations, we first divide the 24 hour samples into three different traffic classes – *low*, *medium*, and *high*. By dynamically combining the results of several of the the predictors, we expect to have better predictions over the entire 24-hour sample period than using a single predictor for all traffic situations. We choose 500 samples, which corresponds to more than a one hour measurement period, to characterize each traffic class shown in Figure 5. The period from $t = 7500 \sim 8000$ corresponds to low traffic, $t = 2500 \sim 3000$ to medium traffic, and $t = 4200 \sim 4700$ to high traffic. We introduce the *traffic class*, TC, to classify the current network state, where TC = (the trimmed $\mathsf{avg}_K(t)$ + the trimmed $\mathsf{stdev}_K(t)$) /(the average latency of the *NSP* sampling message during an idle period).

Figure 6 shows the calculated *TC* parameter values for the different traffic classes. We identify the network as

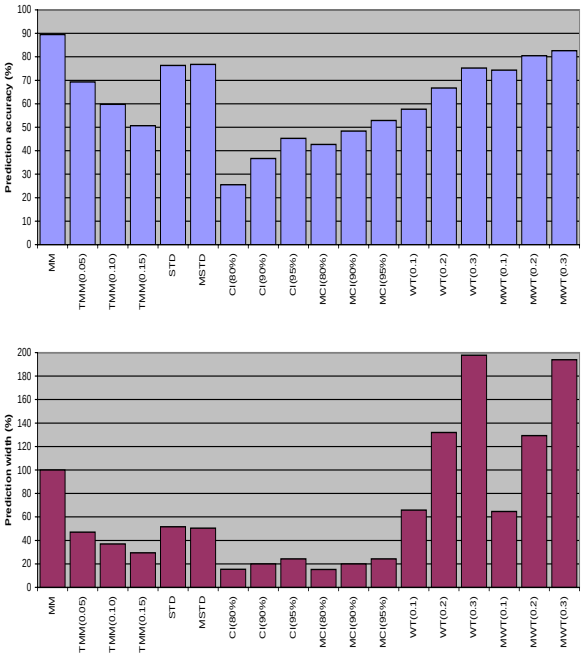

Figure 4. A comparison of the prediction accuracy (PA) and prediction width (PW) of the different predictors over the 24-hour sample.

being in the *low* traffic situation when $TC \leq 1.1$, in the *medium* traffic situation when $1.1 < TC \leq 2.0$, and in the *high* traffic situation when $TC > 2.0$.

Figure 7 shows the prediction accuracy and prediction width of the different prediction methods over the three different traffic classes. In *low* traffic, which shows a very stable pattern, the MIN-MAX (MM) and weight based (WT and MWT) methods all show good performance with prediction accuracies of greater than 90%. The weight method (WT) with α=0.1 shows a prediction accuracy of 96% with only an 11% wider prediction width than that of the baseline MIN-MAX method (MM). In *medium* traffic, which shows only occasional bursts of traffic, the MIN-MAX (MM), standard deviation based (STD and MSTD), and modified weight (MWT) methods all achieve good prediction accuracies. When considering the prediction width, however, the modified weight method (MWT) with α=0.2 shows the best result. In the *high* traffic case, the pattern tends to be very bursty which makes good prediction difficult. In this situation, the weight based methods (WT and MWT) become almost useless having a prediction accuracy of less than 50%. The MIN-MAX method (MM) shows good prediction accuracy, but with a high prediction width. The trimmed MIN-MAX method (TMM) with a small value of α and the standard deviation based methods (STD and MSTD) do show

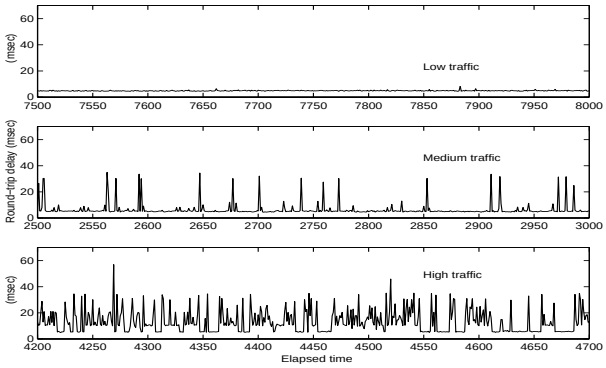

Figure 5. Three periods showing the different traffic classes extracted from the 24-hour measurement period.

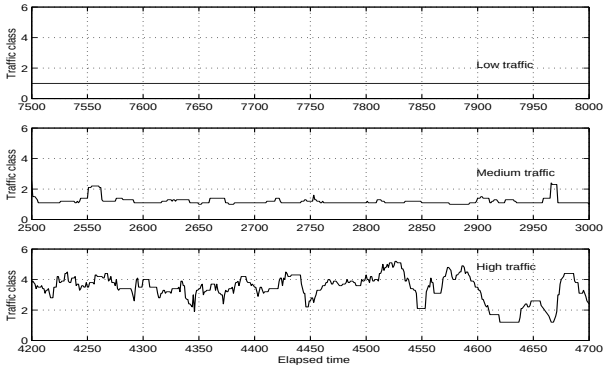

Figure 6. The value of the traffic class parameter, *TC*, for the three different observed traffic classes.

reasonable prediction accuracies with moderate prediction widths, though.

From these observations, we expect to obtain better predictions by combining the weight based (WT and MWT), the trimmed MIN-MAX (TMM), and the standard deviation based (STD and MSTD) methods into a single predictor that selects one of these methods according to the currently observed network load. Figure 8 shows the prediction performance obtained when dynamically applying different prediction methods based on the traffic class, *TC*. Also shown for comparison are the prediction performance results of the MIN-MAX (MM) and the modified weight (MWT) methods. Table 1 shows how the three combined predictors use different combinations of the proposed methods. Combined predictor *dynamic3*, which uses the weight method (WT) with α=0.1 for low traffic, the modified weight method (MWT) with α=0.2 for medium traffic,

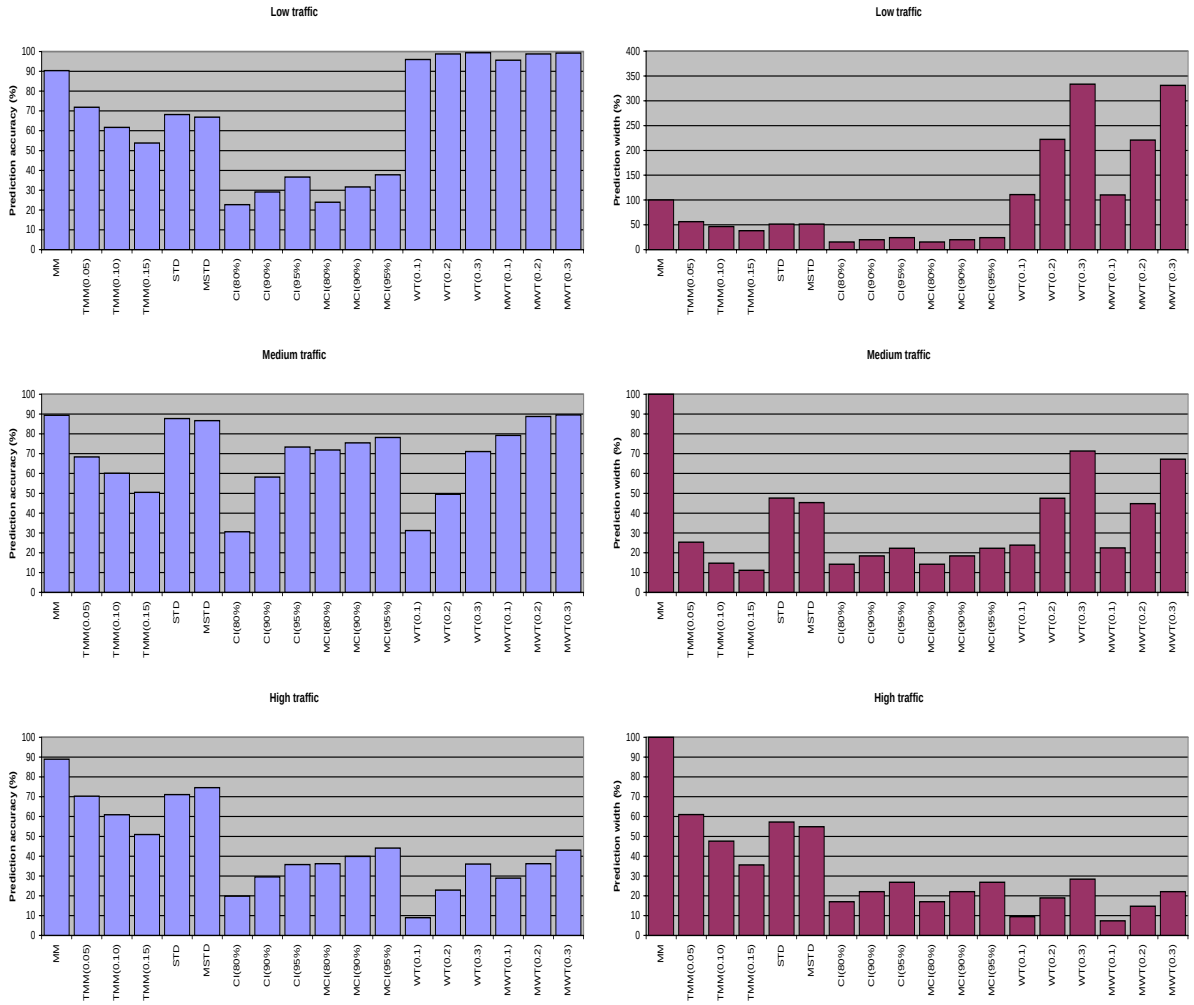

Figure 7. Prediction accuracy and width of the predictors for the different traffic classes.

and the modified standard deviation method (MSTD) for high traffic, produces an overall 85.2% prediction accuracy with an 81.7% prediction width, which is an 18.3% reduction over the baseline MIN-MAX method (MM). The other combinations of the different methods show similar performance. Thus, we conclude that, by dynamically combining different methods, we can predict network status very well throughout different network operating regimes.

	Traffic Class		
	Low traffic	Medium traffic	High traffic
Dynamic1	WT(0.1)	MWT(0.2)	TMM(0.05)
Dynamic2	WT(0.1)	MWT(0.2)	STD
Dynamic3	WT(0.1)	MWT(0.2)	MSTD

Table 1. The combined predictors of Figure 8 use different combinations of prediction methods based on the *Traffic Class*.

4. Related Work

The *application-level scheduling (AppLeS)* [1] project provides a mechanism for scheduling individual applications on distributed metacomputing systems. Its *application-centric programming* evaluates everything in the system in terms of its impact on the application. *AppLeS* agents utilize a *network weather service (NWS)* [1, 11] to monitor the varying performance of the resources potentially usable by their applications. *NWS* periodically samples the values of various important parameters, such as processor-to-processor latency and bandwidth, to dy-

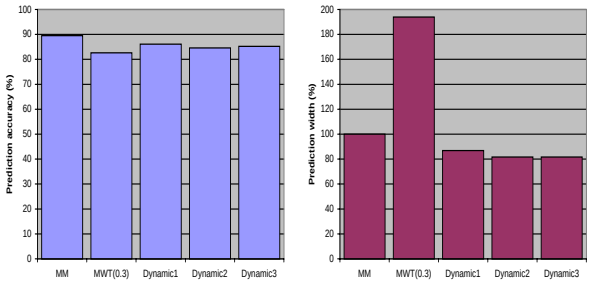

Figure 8. The prediction accuracy and prediction width obtained by combining different prediction methods based on the traffic class over the 24-hour sample period.

namically forecast their expected future values using several numerical models. The *Legion* [6] and *Globus/Nexus* [4, 5] metacomputing systems also employ *NWS* for network scheduling.

The *Network Status Predictor (NSP)* presented in this paper is a distributed system that periodically detects and predicts network load. The specific implementation of *NSP* we tested is similar to *NWS* [11] in that the network load is sensed directly with a user-level daemon that runs on each network-attached node. The daemon periodically samples network latency by sending and receiving some fixed-size messages. However, the size of the sampling message and the sampling period can differ among the systems. For instance, *NWS* uses a $64K$ byte message sent every 30 seconds.

Although the average network latencies measured using different sizes of sampling messages could be similar, their variances could be quite different. Typically, larger sampling messages lead to smaller variances in measured network latency. After measuring the characteristics of the Fibre Channel network, we chose an $8K$ byte sampling message size for *NSP*. This size corresponds to around 75% of its saturation bandwidth with a 10 second sampling interval over TCP socket connections.

While *NSP*'s basic sampling technique is similar to *NWS*, it also has some important differences. In *NWS*, average and median values of some subset of samples are used to predict the future expected performance of network resources. However, because of the burstiness of network traffic, a single number is not very reliable unless the network stays in a relatively stable state [9]. *NSP*, however, predicts lower and upper *bounds* for the network latency using samples of recent latency measurements. In addition, *NSP* dynamically selects among several different predictors based on the network's current status, which is quantified using the *Traffic Class* parameter.

5. Conclusion

To predict the expected latency of messages sent on an interprocessor communication network in a distributed computing system, the *network status predictor (NSP)* uses direct, periodic measurements of network latency as the inputs to several numerical models to calculate lower and upper prediction bounds. We evaluated the performance of *NSP* using the *prediction accuracy* and the *prediction width* metrics to quantify the trade-off between the accuracy and the size of the predicted interval.

We found that no single numerical model provided good performance for all network situations. Instead, we introduced a combined predictor that dynamically selected among prediction models based on the *traffic class* parameter, which quantifies the network's recent state. This combined predictor produced consistently good performance over all traffic classes. This *dynamic composition* predictor should prove helpful in effectively allocating network resources in distributed computing systems.

References

[1] F. Berman, *et al*, "Application-Level Scheduling on Distributed Heterogeneous Networks," *Supercomputing*, 1996.

[2] D. E. Culler, *et al*, "Parallel Computing on the Berkeley NOW," *9th Joint Symp. on Parallel Processing*, 1997.

[3] T. von Eicken, *et al*, "Low-Latency Communication over ATM Networks using Active Messages," *IEEE Micro*, February 1995, pp. 46-53.

[4] I. Foster, C. Kesselman, and S. Tuecke, "The Nexus Task-parallel Runtime System," *Intl. Workshop on Parallel Processing*, 1994. pp. 457-462.

[5] I. Foster, *et al*, "Multimethod communication for high-performance metacomputing applications," *Supercomputing*, 1996.

[6] A. Grimshaw, *et al*, "Legion: The Next logical step toward a nationwide virtual computer," *Report no. CS-94-21, Computer Science, U. of Virginia*, 1994.

[7] J. Kim and D. J. Lilja, "Exploiting Multiple Heterogeneous Networks to Reduce Communication Costs in Parallel Programs," *Heterogeneous Computing Workshop*, April 1997, pp. 83-95.

[8] J. Kim and D. J. Lilja, "Utilizing Heterogeneous Networks in Distributed Parallel Computing Systems," *Intl. Symp. on High Performance Distributed Computing*, August 1997, pp. 336-345.

[9] R. Mraz, "Reducing the Variance of Point-to-Point Transfers for Parallel Real-Time Programs," *IEEE Parallel and Distributed Tech.*, 1994, pp. 20-31.

[10] S. S. Mukherjee and M. D. Hill, "Making Network Interfaces Less Peripheral," *Hot Interconnects V*, August 1997.

[11] R. Wolski, "Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service," *Intl. Symp. on High Performance Distributed Computing*, August 1997, pp. 316-325.