

Synchronizing Network Probes to avoid Measurement Intrusiveness with the Network Weather Service

Benjamin Gaidioz

RESAM

Université Claude Bernard Lyon 1

Rich Wolski

Computer Science Department

University of Tennessee

Bernard Tourancheau

RESAM

Université Claude Bernard Lyon 1

Abstract

In this paper we present a scalable protocol for conducting periodic probes of network performance in a way that minimizes collisions between separate probes. The goal of the protocol is to enable active performance monitoring of large-scale distributed computational systems and networks. We use the protocol to generate time series of measurement data that are then exposed to numerical forecasting models when a prediction of network performance is required. We present the protocol and demonstrate its effectiveness using the Network Weather Service — a tool for dynamically predicting network, CPU, memory, and storage performance.

1. Introduction

Thanks to the increasing availability of high quality networks, it is now possible to combine distributed computing and data resources in service of high-performance applications. Computational Grid [7] software infrastructures like Globus [6], Legion [9], Condor [18], and NetSolve [4] provide abstractions designed to give the user the ability to run applications on a heterogeneous set of machines as he or she would on a single high-performance platform. The analogy that is relevant is with the electrical power grid. Applications are to draw computational “power” from a collection of resources in the same way that appliances draw electrical power from a power utility –ubiquitously and seamlessly.

Any Computational Grid system must include a *scheduler* (either human or automatic) the goal of which is to select the most appropriate resources (hosts, network links, disk storage, etc.) that are going to be used by an application. The scheduler must choose the best resources according to the network performance that will be available

if some data has to be transmitted, the CPU load in order to determine how long the computation will take, and the amount of real memory that will be obtainable on each processor. Since this choice is made before the application loads each resource, it is clear that it should be based on what the performance of CPU and network *will* be instead of what the performance levels are at the time the decision is made. That is, any scheduler decides on resource usage for a future time frame. It is a prediction of resource availability for that time frame, and not current conditions, that must be considered.

The Network Weather Service [19, 20, 17](NWS) is a distributed system that regularly takes load and availability measurements from a collection of Grid resources and uses them to generate short-term performance forecasts. To monitor network links, the NWS conducts end-to-end network probes (which it uses to measure available network performance) and then applies fast statistical models to probe histories to make performance forecasts [19]. To conduct a probe, one host opens a connection with another and sends a small message to measure the link round-trip time, and a large one to measure the throughput. At the time a scheduler needs a prediction of the performance, mathematical forecasting models are applied to a time-series of consecutive measurements to generate a prediction of what the next value in the series will be.

In this paper, we describe a protocol for controlling the intrusiveness of NWS network measurements while, at the same time, ensuring their consistent periodicity. The new protocol allows the system to ensure minimal intrusiveness if network performance conditions permit, but automatically adapts the system’s behavior when conditions deteriorate to ensure consistent periodicity.

It is important for network probes not to interfere with each other. Otherwise, the NWS will make its predictions based on the combined load introduced by several simul-

taneous probes rather than on what is available to a single probe. In [21] we note that the impact of probe collisions can be substantial, causing an inaccuracy of as much as 50%. Thus, probe collisions have to be avoided as much as possible. To prevent probe interference, we ensure mutual exclusion between network probes using a token passing protocol. Only the host holding the token is allowed to probe the network. While a host is holding the token, it conducts a single network probe between itself and all other end points in its collection (called a *clique*). The arrival of the clique token triggers each network probe.

To allow the token-passing scheme to scale, we do not put all monitored hosts into a single clique. Rather, we organize hosts into a hierarchical set of cliques. Hosts at one level of the hierarchy pass a clique token only with other hosts at the same level.

When time-series models are applied to the measurements, their effectiveness depends on the how regularly the measurements are taken in time. Most models assume that the time between measurements is constant when generating a forecast. Perhaps more importantly, the duration during which a forecast is valid depends on the periodicity with which the measurements are gathered. A one-step-ahead forecast is valid for only a single period. If that period is not constant, it is difficult to assign a meaningful lifetime to any given forecast. Under the current NWS token-passing protocol, if the time required for the token to circulate varies, the individual probe periodicity will vary as well. Our work described in this paper attempts to ensure that tokens are passed periodically and, at the same time, that probes collide as little as possible. *In this paper, we describe a new token-protocol and discuss how well it ensures both consistent periodicity and mutual exclusion as compared to the old protocol.*

2. The Network Weather Service

The Network Weather Service is a distributed system, the aim of which is to generate accurate forecasts of resource performance for a collection of CPU, network, memory, and storage resources. The forecasting engine applies mathematical models to series of measurements that are taken at the application level so that the predictions are close to the performance available to applications. To generate a forecast, the NWS operates several different models simultaneously¹ and computes a forecast from each. At the next time step, when a measurement is taken, the measurement is compared to each forecast and the forecasting error is recorded for each forecasting model. When an NWS client requests a forecast, the forecasting engine examines the cumulative error measures recorded for each forecasting

¹The results reported in this work were generated using 17 separate models.

model, and chooses the one that has the lowest cumulative error up to that point to make the forecast.

2.1. NWS component processes

NWS is also a modular system which is built from different *component processes* that run on the hosts the user wants it to monitor. According to its needs, the administrator can dynamically remove or add components to its system. Four components are available:

name server The name server component implements a highly portable name location binding. It records the set of time series that are actively being gathered by the system and the host locations where they are stored.

memory Measurements that have been taken are then remotely stored as time stamp-measurements pairs in memory hosts. The data is immediately written in a file so that it can survive a failure of the process.

sensor A sensor periodically takes measurements of the available performance of some resource and sends it to a memory component. Sensors can be dynamically re-configured through specific messages sent by the user.

forecaster When a forecaster is asked for a prediction, it contacts the name server to learn the location of the data, downloads it from the specified memory host, applies time-series models to the data received, and delivers the forecast to the client.

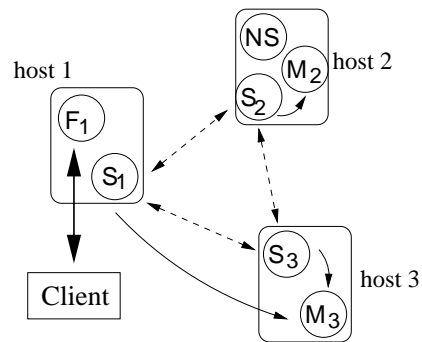


Figure 1. An NWS system monitoring three machines

Figure 1 depicts an NWS system monitoring three hosts. The name server process is running on host 2 and network sensors S_1 , S_2 and S_3 have been started on each host. S_1 and S_3 store their results in the memory M_3 while S_2 stores its measurements in M_2 . The sensors periodically takes measurements, represented by the dashed arrows, of the

available performance of the network. A forecasting process F_1 is running on host 1 and delivers forecasts of the system to the client.

2.2. The TCP experiment

In order to get results on the available network performance, the network sensor periodically probes the network by moving some data between sensors, and timing the move. Both latency (measured as round-trip time divided by 2) and bandwidth are measured. The structure of the NWS probing mechanism is depicted in figure 2 where host A opens a connection with B , starts a timer and measures round-trip time and bandwidth to host B as follows:

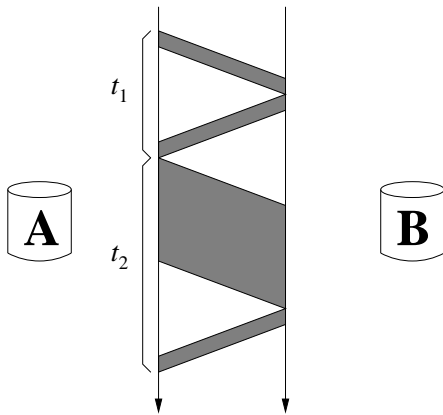


Figure 2. The TCP experiment

- A sends a small (four bytes) message, B sends it back to A and the timer is stopped. Let t_1 be the time the round-trip took, latency is calculated as $t_1/2$.
- Right after, A starts its timer again and sends a large message to B . After having received the message, B closes the connection and A stops the timer. Let t_2 be the data transfer time and s be the size of the message, bandwidth is calculated as $s \div (t_2 - t_1)$.

Both values are then remotely stored in a memory host.

2.3. The token-protocol

Instead of having all network sensors of a system asynchronously conducting experiments with each other, the administrator organizes the system as a hierarchy of hosts sets called *cliques*. A clique consists of a set of hosts where a given network probe is conducted by each machine with other members of the clique with a given period. Figure 3 shows a system that has been set up on fifteen machines.

The three domains that are monitored include five hosts each on which cliques (cliques 1, 2 and 3) have been started. Network performance inside these domains is known end-to-end. Two others cliques (cliques 4 and 5) have been set up. Both include only two hosts from different domains. The administrator can approximate the network performance between machines of different domains using the measurements that have been taken between the single pair of hosts in the cliques that span domains. This hierarchical organization permits the NWS intrusiveness control mechanisms to scale.

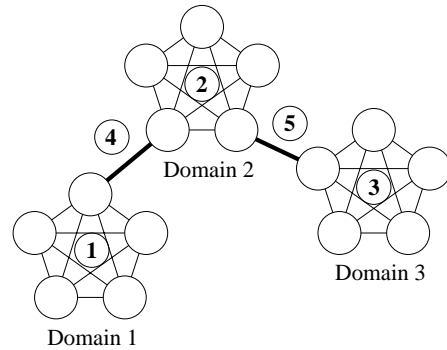


Figure 3. A system organized as a hierarchy of cliques.

The NWS uses time series analysis, [8, 1, 10] to predict the performance of the network. Each monitored host is expected periodically to conduct several network probes that give a measure of the latency and the throughput of the link between it and all other hosts. The more periodical the series is, the more accurate and statistically valid (in a theoretical sense) the forecasts will be.

One way to achieve a consistent periodicity of probes is to run each sensor on its own clock as gloperf [13] does, but when there is complete overlap, the measurements change dramatically (see figure 4). Thus, NWS uses a token-protocol that ensures mutual exclusion of the probes.

3. Token Passing and Probe Periodicity

In order to initiate the token protocol, the user sends a control message to one of the hosts. It includes a list of hosts to probe, the periodicity with which to probe, and parameters to the network probe itself (buffer size, message size, etc.). All of these values are carried in the token which is passed from host to host. When a host receives the token, it probes all of the links between itself and all of the hosts listed in the token, and then forwards the token to the next host. As soon as the next host receives the token, it probes the network links between itself and the other hosts listed in

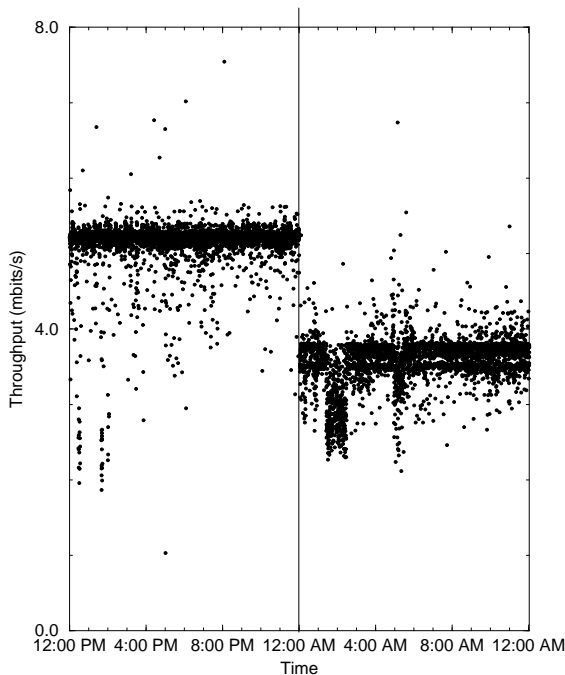


Figure 4. Probe collisions can effect probe accuracy. On the left of the vertical line are readings taken when probes do not collide. On the right are reading taken when a probe intentionally collides with another [21].

the token, and then forwards the token. After all hosts have seen the token, it is returned to the original initiator (termed the leader). The leader calculates the time required to circulate the token, subtracts that from the desired periodicity, and re-initiates the token at the next period point. For example, if the token is to be circulated every thirty seconds, and the last complete circuit took 7 seconds, the leader will re-initiate the token 23 seconds after it is returned.

There are two factors that affect the periodicity of each host's probes.

- The time required by each host to contact all other hosts listed in the token can vary.
- If a network link fails, the hosts can become partitioned, and if a host crashes, the token can be lost.

To handle both of these cases, each host maintains a timer and, after it times out, manufactures a token which it then initiates. When a new token is initiated by a host after a time-out, the initiating host declares itself the leader, and increments a token-carried sequence number. If (after a network restoration, for example) a host encounters two tokens, it annihilates the one with the lower sequence number. If the sequence numbers match, then the one with the highest numbered leader is chosen to break ties. Until all

old tokens are annihilated, however, multiple tokens may be circulating causing the measurement periodicity to vary.

To set the time-out for a token arrival, each host tracks the token circulation time and the variance in that time. When the token is received by a host, its circuit time is recorded, and the NWS forecasting models are invoked to predict the next circuit time and the variance in circuit time. The time-out is then set to the predicted circuit time plus twice the predicted standard deviation. That is, the NWS adapts its time-outs automatically based on forecasts of token circulation time. This scheme is good at controlling the number of tokens that are circulating but it does little to ensure consistent periodicity.

4. A New Token Protocol

In this section, we present a new clique-protocol which works almost the same way as the previous one with a modification to the time-out mechanism. We will see that this protocol leads to more regular measurements since we provide to the user a way to specify a upper bound value for the period instead of dynamically calculating a time-out value.

If the token arrives before the higher limit, the protocol is unchanged and hosts use the token arrival to trigger a spate of network probes. The lower end of the range dictates the desired periodicity. That is, the user specifies what periodicity is desired, and how long each host should wait before it times-out. Instead of generating a new token at time-out, however, the new protocol considers it as *late*, and the host begins probing the links to other hosts in its list under purely local control (possibly causing collisions). Even though no token is created, the host keeps waiting for a token to arrive during the time it is probing. If the token is received while the probes are in progress, it is kept until the host has finished probing its list and forwarded normally. If the token is not received, however, by the beginning of the next period, a new token is immediately generated and forwarded. Multiple tokens may be circulated until the newest one annihilates the older ones, and during this phase, probe collisions may occur. The main idea, however, is that hosts revert to locally (non-mutually exclusive) operation *automatically* when network performance prevents a periodicity that is within the user's specified range. If the network variance allows the token to be passed while maintaining the user's desired periodicity, then the system enforces mutual exclusion, and it switches between these phases based on the observed performance conditions.

This algorithm is illustrated using a three host clique in Figure 5. In the figure, each host is associated with a horizontal time line in which thick parts represent time conducting probes. Dotted lines shows the clique token pass between hosts.

During the first complete circuit (labeled 1 in the figure),

hosts conduct their probes one-by-one and the token protocol synchronizes them so there are no collisions. While a host is holding the token, it probes the network connection between itself and the other two hosts, measuring the available bandwidth and the round-trip duration. When host 1 gets the token back, it waits before starting the new set of probes. Assume that the network performance become suddenly worse so that the time to probe all hosts (thick lines) is longer in round 2. In the figure, during the second circuit, assume that the user's upper range has been violated causing host 2 and host 3 both start their probes without holding the token. Note that in this condition, host 2 and host 3 will collide if they probe the same network link. Notice also that the user's range delays the time at which host 2 and host 3 start as they will time-out when the upper end of the range is reached. That is, if the user specified a periodicity of 30 seconds, and an upper range of 35 seconds, host 2 and host 3 will start 5 seconds late. In the following round (labeled 3 in the figure), the delay causes host 2 to receive the token within the user's specified range and it is synchronized with host 1 again. Host 3 is still shifted compared to host 2 (since it started a little later) but it becomes synchronized in the next round in the same way that host 2 did.

The user now fixes the "stretchability" of the period as a range of time values within which the periodicity must be maintained. By enforcing the periodicity range, the new protocol potentially sacrifices mutual exclusion if the network conditions will not permit the token to ensure the desired periodicity.

- If network performance is relatively stable like it may be in a local area network, the user could specify a small stretchability. If the network is suddenly slowed so that the token is delayed, collisions may occur but the amount of collisions should not be high.
- If the system is set up on widely dispersed hosts, network performance may significantly vary. Delayed tokens may more often lead to probe collisions affecting the accuracy of the forecasts.

The new protocol gives the user the ability to control the tradeoff between mutual-exclusion and periodicity. In the next section, we will study the effectiveness of this new protocol.

5. Results

We implemented the protocol that we have described. Our study will focus on the performance of both protocols in terms of periodicity and percentage of wall-clock time that probes collide.

5.1. Description of the experiment

Since network performance varies from one moment to the next, the only way to compare both implementations of the clique-protocol is to use them at the same time on the same hosts. When a component of a NWS system is run, the user can specify a port number that the program will use if the default port number is inappropriate. Using this functionality, two NWS systems (one running the old protocol and the other running the new) were started on the same set of hosts, each one running independently of the other, both including two hosts located in ENS, Lyon in France and two hosts located in UT in Knoxville. Cliques were configured for a 240 second periodicity with a five second stretchability (2% of the period) for the one running the new protocol. The transoceanic link experiences wide performance varia-

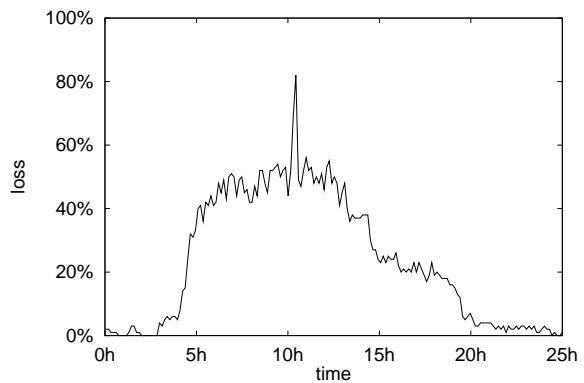


Figure 6. Ping statistics from Lyon to Knoxville

tion (both latency and bandwidth) during a typical day so that our study focuses on 24 hours of the experiment where the systems saw both good and bad network performance. During this experiment, we recorded the loss statistics given by the `ping` command (see figure 6).

5.2. Periodicity Results

In the figures 7 and 8, we show the evolution of the period of probes for one of the four hosts during the experiment. In the figures, the y axis denotes the period between probes for the given host, and the x axis shows the time it was recorded. A value substantially above 240 on the y axis indicates a late set of probes and one below 240 indicates probes that were conducted early. Both graphs can be correlated with the loss statistics shown on figure 6.

- We see that between 0h and 5h, loss statistics are sufficiently low to obtain a good periodicity of the mea-

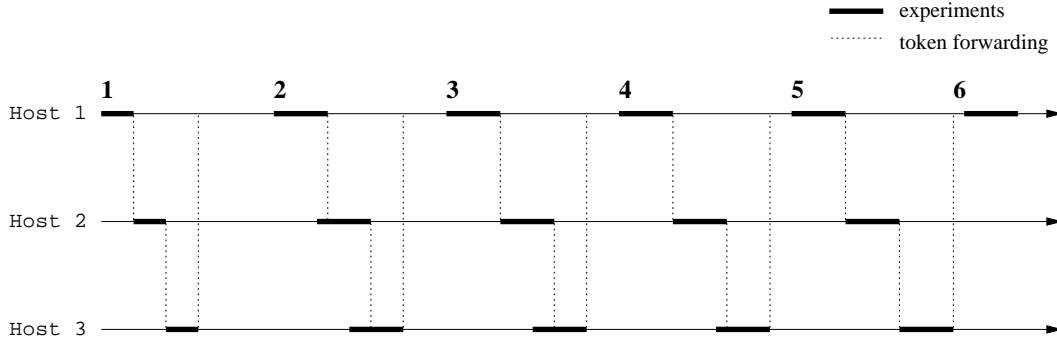


Figure 5. Timing Diagram for the New NWS Clique Protocol

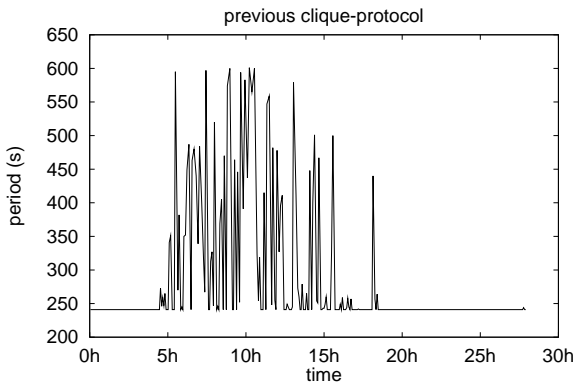


Figure 7. Periods of the jobs using the previous protocol

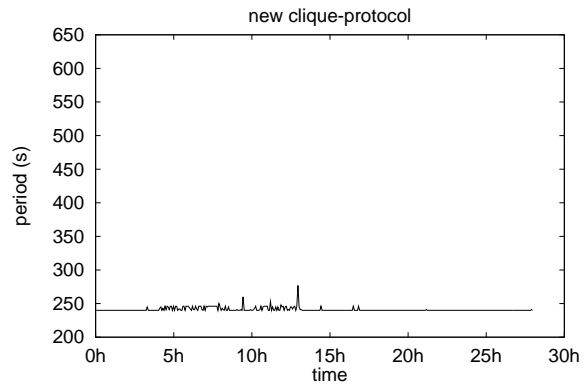


Figure 8. Periods of the jobs using the new protocol

surements. Both protocols yield a periodicity near 240 seconds.

- Then, until 15h of execution, the ping statistics reports a much higher loss of around 50%. Hosts running the old protocol calculate large time-outs to ensure mutual exclusion of their probes, leading to periods of 2.5 times what was specified by the NWS administrator. The new protocol yields a less varying period. Most of time, it is kept between 240 and 245 seconds. Infrequently, it happens to suddenly spike to five seconds (see the peak between 10h and 15h on the x axis). This delay is an artifact of the NWS non-multithreaded implementation. Occasionally, a host becomes “locked” while attempting to re-register itself with the name server and cannot initiate its own probes. The time-out for this name server communication is set to a maximum of 5 seconds, hence the 5 second spike. The new token protocol does not alleviate this source of delay, although we discovered the problem as a result of this research. It appears to happen far less frequently than

delays caused by token slowness, however, and we are currently working on modifying the implementation to address this issue.

- Loss becomes low again and both protocol manage to follow the specified period.

5.3. Collision Percentage

Table 1 compares the collision frequency between the old protocol and the new. Since the network probes could overlap for different amounts of time (*i.e.* a collision is not a simple “yes” or “no” condition), we report the percentage of collision time over several periods during the execution.

0h–5h The first two rows compare the old NWS protocol (the first row) with the new protocol (the second) during the first five hours of the experiment where, according to the ping statistics, the network performance is fairly good. The old protocol experiences no collisions. No probes were underway 88.7% and one probe was underway 11.3% of

the time. This behavior is what was expected since the old protocol is designed to ensure mutual exclusion of probes. Note, also, that these results indicate that the old protocol never experienced a spurious a clique time-out. If one of the hosts had erroneously timed out, it would have generated a second token resulting in probe collisions. The automatic time-out discovery mechanism used by the old protocol successfully ensured mutual exclusion over the duration of the test. The new protocol ensures less mutual exclusion. Due to network performance variation, the case of several hosts conducting probes at the same time occurred, but the amount of time it represents is less than 1% of the total time.

5h–15h The next two rows show the performance of both protocols during a part of the execution where network loss between Lyon and Knoxville is much higher. We first notice that 45% of the wall-clock time is spent conducting experiments, showing that an experiment takes more time to complete than before. The old protocol generates collisions for 2.8% of the total test time thanks to its automatic clique time-out. However, the figure 7 shows that the periodicity is completely lost during that period of the experiment. Most of the overlap obtained by the new protocol is for two hosts conducting an experiment simultaneously (11.1% of the total time) and three or four hosts probing the network at the same time occurs only 1.9% of the time. Overlap occurs because the protocol keeps ensuring the specified periodicity of 240 seconds with a five second cap on the lateness of any probe.

17h–24h The network returns to good performance as it is shown by the ping statistics and the periodicity of the probes is restored. The following two rows shows that during that period, no collisions occurred for both protocols.

Table 1. Collisions in the four NWS systems

		% time probing simultaneously				
		0	1	2	3	4
0h — 5h	old NWS	88.7	11.3	0	0	0
	new NWS	91	8.1	0.8	0.1	0
5h — 15h	old NWS	54.6	42.6	2.4	0.4	0
	new NWS	51.5	35.5	11.1	1.8	0.1
17h — 24h	old NWS	88.7	11.3	0	0	0
	new NWS	90	10	0	0	0
Total	old NWS	76.2	22.8	0.8	0.1	0
	new NWS	75.9	19.3	4.1	0.6	0.1

6. Related Work

There are a wide variety of network performance monitoring tools [16, 5, 12, 13, 3, 15], excellent surveys of which is available from [11] and [2]. Network monitoring can be categorized as either active or passive. An active network monitor probes network resources by loading them, and then observing and recording the resulting performance. The NWS TCP/IP sensor is an active sensor, but it differs from other active monitoring tools such as [16] and [3] in that it uses end-to-end TCP/IP probes and not ICMP echo packets to measure network performance. The disadvantage of this approach is that the NWS requires an execution presence at each end-point of the network segment being measured. In contrast, ICMP echo is returned by most Unix systems, so measurement does not require collaboration between hosts at the end-points. The advantage, however, is that TCP/IP probes reflect flow-control and congestion control effects that are not observable with ICMP echo.

Other systems, such as [13] use TCP/IP measurement probes, but make no attempt to coordinate probes across hosts that are being monitored. That is, each gloperf sensor operates on its own periodic clock and no mechanism is provided for preventing experiment collisions. In [21] we describe the effect of probe collisions (reproduced in figure 4). In some settings, colliding probes can cause a 50% loss of accuracy. Certainly, if the system is to scale to moderate numbers of hosts, the intrusiveness of the probe traffic must be controlled explicitly and sensors must be organized to avoid N^2 measurements (for N hosts) at every period.

Lastly, it possible to combine active probing with passive packet traffic readings that can be gathered via the Simple Network Management Protocol (SNMP), as described in [15]. While packet traffic may be self-similar [14], it is possible to correlate observed application performance with gateway traffic if the number of gateways traversed is small. It is not clear how this approach will scale to the wide-area, where SNMP traffic data may not be provided or may be erroneously reported. The advantage of such an approach, however, is that it is non-intrusive. We believe that it can be combined with controlled, periodic measurement as part of a scalable network forecasting tool.

7. Conclusion

The main goal of the Network Weather Service is to provide accurate measurements of the performance of specific resources so that effective statistical forecasting based on time-series analysis is possible. Since the network probes conducted by hosts to measure network performance introduce some load of the network, mutual exclusion is ensured by a token-protocol. The fact is that the quality of the fore-

casts depends on the periodicity of the measurements and that the previous token-protocol can lead to a bad periodicity because of its time-out system.

We have designed and implemented a new token protocol to ensure mutual exclusion when network conditions permit, but which automatically reverts to locally controlled probes (with no synchronization) when conditions deteriorate. We allow the user to specify a range of periodicity which, if violated, will cause unsynchronized measurements. Using this new protocol, the periodicity of the network probes is better controlled at the possible expense of a greater number of collisions. We measure the increased collision time for a combination of local and wide area networks, and find that it is acceptably low, even when the periodicity range is set to 2% of the expected period.

References

- [1] G. Box, G. Jenkins, and G. Reinsel. *Time Series Analysis, Forecasting, and Control, 3rd edition*. Prentice Hall, 1994.
- [2] Caida web page at <http://www.caida.org/tools>.
- [3] R. Carter and M. Crovella. Dynamic server selection using bandwidth probing in wide-area networks. Technical Report TR-96-007, Boston University, 1996. available from <http://cs-www.bu.edu/students/grads/carter/papers.html>.
- [4] H. Casanova and J. Dongarra. NetSolve: A Network Server for Solving Computational Science Problems. *The International Journal of Supercomputer Applications and High Performance Computing*, 1997.
- [5] N. L. for Applied Network Research. Amp. <http://amp.nlanr.net>.
- [6] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 1997.
- [7] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., 1998.
- [8] C. Granger and P. Newbold. *Forecasting Economic Time Series*. Academic Press, 1986.
- [9] A. S. Grimshaw, W. A. Wulf, J. C. French, A. C. Weaver, and P. F. Reynolds. Legion: The next logical step toward a nationwide virtual computer. Technical Report CS-94-21, University of Virginia, 1994.
- [10] R. Haddad and T. Parsons. *Digital Signal Processing: Theory, Applications, and Hardware*. Computer Science Press, 1991.
- [11] Idmaps web page at <http://idmaps.eecs.umich.edu>.
- [12] R. Jones. <http://www.cup.hp.com/netperf/netperfpag.html>. Netperf: a network performance monitoring tool.
- [13] C. Lee, R. Wolski, J. Stepanek, C. Kesselman, and I. Foster. A network performance tool for grid environments. In *Proc. of SC99*, November 1999.
- [14] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic. *IEEE/ACM Transactions on Networking*, February 1994.
- [15] B. Lowekamp, D. O'Hallaron, and T. Gross. Direct network queries for discovering network resource properties in a distributed environment. In *Proc. 8th IEEE Symposium on High-Performance Distributed Computing (HPDC-8)*, Redondo Beach, CA, Aug. 1999.
- [16] A. Network and I. Services. Surveyor. <http://www.advanced.org/surveyor/>.
- [17] The network weather service home page - <http://nws.npaci.edu>.
- [18] T. Tannenbaum and M. Litzkow. The condor distributed processing system. *Dr. Dobbs Journal*, February 1995.
- [19] R. Wolski. Dynamically forecasting network performance using the network weather service. *Cluster Computing*, 1998. also available from <http://www.cs.utk.edu/rich/publications/nws-tr.ps.gz>.
- [20] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems (to appear)*, 1999. available from <http://www.cs.utk.edu/~rich/publications/nws-arch.ps>.
- [21] R. Wolski, N. Spring, and C. Peterson. Implementing a performance forecasting system for metacomputing: The network weather service. In *Proceedings of Supercomputing 1997*, November 1997.