

Multiple Paths for End-To-End Delay Minimization in Distributed Computing Over Internet *

Nageswara S. V. Rao
Center for Engineering Science Advanced Research
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6355
raons@ornl.gov
presented at *Supercomputing 2001*, Denver, Colorado

Abstract

Messages exchanged between processes distributed over the Internet via the usual TCP streams are subject to various delays at the routers and hosts. In addition to being limited by the “effective” bandwidth, the end-to-end delays of the messages have a significant “random” component due to the complicated nature of the network traffic. We propose a measurement-based method to implement multiple paths for achieving low end-to-end delays for message transmissions in distributed computing applications. Our scheme is implemented over the Internet by a network of user-level daemons, which maintain an accurate “state” of delay-regressions in the network. These daemons handle all network tasks between the processes and also perform transport-level routing. They explicitly realize multiple paths via themselves, thereby achieving physical diversity of the transmission paths as well as higher aggregated bandwidth compared to usual and parallel TCP methods. Multiple path routing is performed among the daemons without explicit support from the underlying network routers. Experimental results indicate that our method is a viable and practical means for achieving low end-to-end delays for distributed computing applications over the Internet.

1 Introduction

High performance applications require aggregated capabilities distributed over a network such as the Internet or ESnet. Such capability might be the aggregated bandwidth using multiple streams, or combined computational power of geographically distributed supercomputers. For example, the former typically arises in high performance storage systems and later arises in climate or physics computations. In both cases, on-demand communication between various processes must be effectively supported over the network. In general, both the nature and volume of communication data could be quite varied among the applications as well as within a single application. The synchronization messages that coordinate the computation between various processes could be small whereas the data transfers could be quite voluminous. The variability of the message sizes must be explicitly accounted for because of the non-monotonicity of end-to-end delays: a path with high bandwidth (suited for bulk transfers) is not be optimal for small messages if it has high

*Research sponsored by the Defense Advanced Projects Research Agency under MIPR No. K153 and by the Laboratory Director's Research Project at the Oak Ridge National Laboratory, and by the Engineering Research Program of the Office of Basic Energy Sciences, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

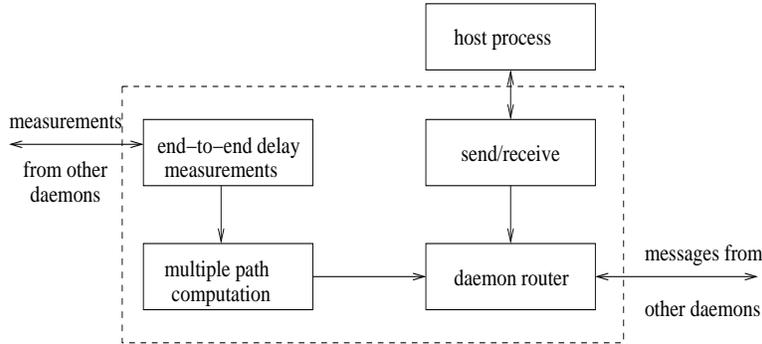


Figure 1: *Functional block diagram of a daemon.*

latency. Indeed, small messages are efficiently sent via smaller bandwidth paths with low end-to-end delay. Such paths are very vital in preventing supercomputers from idling while waiting for a message/data to arrive over the network.

In most existing Internet solutions for achieving low delays, the messages are typically sent as single or parallel Transport Control Protocol (TCP) streams. The TCP stack can be optimized to account for various host and network parameters. Such solutions, however, do not in general exploit the physical diversity of paths in the network. The end-to-end delays are also subject to limitations imposed by the policies and traffic loads at the routers, in addition to the bandwidth limits of the physical links. Furthermore, the implementation of TCP stack can significantly add to the end-to-end delays by limiting the achievable bandwidth. While the latter delays are somewhat measurable and predictable, those at the routers are not so easily accountable for the following reasons. In the present networks, such as the current Internet, the messages are decomposed into datagrams and sent via various routers as per the Internet Protocol (IP). At the routers the incoming datagrams from all sources are serviced using the best-effort method – the datagrams can be delayed or altogether dropped at the routers. Consequently, the end-to-end delay of a message transmission can be highly unpredictable, especially if the datagrams are routed through high traffic regions. While such unpredictability can be tolerated in services such as email, it can cause severe problems in several applications such as data transfers to and from supercomputers, instrument control, and distributed simulation.

We propose a network of daemons that exchange measurements, and perform source-based multiple path routing via themselves. At the hosts, the messages will be simply handed over to the local daemon which performs all the networking tasks needed to achieve the end-to-end performance. The user thus is relieved from having to explicitly account for networking. Our main emphasis is on *multiple paths* via the daemons, which provide several advantages compared to single paths. These daemons are special instantiations of NetLets, which are described in detail in [5, 7].

There are several works that deal with infrastructure- or measurement-based methods for providing certain quality of service, such as overlay networks [2], Detour project [9], and parallel TCP for large memory transfers [8, 3] or web traffic [1]. While our work is focussed exclusively on end-to-end delays, it has several unique features: (i) multiple physical paths are explicitly optimized and realized for the message transport, (ii) the required measurements are collected by internal instruments, (iii) our measurement and path computation methods are analytically justified under very general conditions (Section 3.2) in terms of distribution-free performance guarantees, and (iv) our implementations on the Internet (Section 4 and [5]), local area networks [7], and simulation [4] show very promising and concrete results.

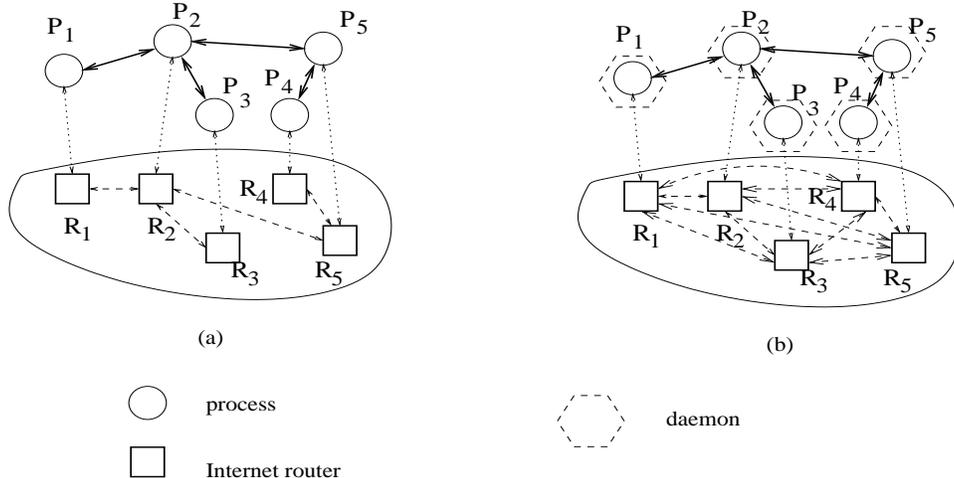


Figure 2: *Typical scenario of distributed computing environment over the Internet.*

The daemons concept for multiple paths is described in Section 2. Theoretical results for the path computation are described in Section 3. Experimental results based on Internet implementation are presented in Section 4.

2 Daemons for Multiple Paths

The daemons perform all networking tasks to relieve the communicating processes of the burden of “accounting for the network”. The daemons run at the processing nodes and communicate over the network to perform three main functions shown in Figure 1: (a) collection of delay measurements, (b) multiple path computation, and (c) routing via other daemons. At present, the routing paths for datagrams in IP networks are decided by the best effort method, and multiple paths are not supported by the Internet routers. In our scheme, the messages between the daemons are source-routed via multiple paths through the daemons.

To illustrate the concept, consider a computational distributed over several nodes over the Internet. Messages must be communicated between the nodes as per the task structure as in Figure 2(a). For example, the communication between processes P_1 and P_2 could be handled by a process-to-process TCP stream(s). At present, a single or parallel TCP streams are utilized to send all messages between them. The actual physical paths taken by the individual datagrams are decided by the network routers. While the datagrams could travel via different physical paths (which are small in number), these paths are not controlled or optimized by the host processes. If a TCP connection stream has high traffic, which results in high delays or packet loss, no rerouting action is taken by TCP, and routers respond by simply dropping the packets at least on the short time scales. Although some level of rerouting might be performed by certain routers, their primary goal is not optimizing the end-to-end performance of any particular host process.

Consider that daemons are executed at each of the processes as shown in Figure 2(b). Daemons maintain peer-to-peer connections among themselves as shown in Figure 2(b), and dynamically estimate various delay regressions of the message size. If a large message has to be sent, a multiple path consisting of $R_1 - R_2$, $R_1 - R_3 - R_2$, $R_1 - R_4 - R_2$ and $R_1 - R_5 - R_2$, can be utilized. Computation of such multiple paths, however, is non-trivial and will be discussed subsequently.

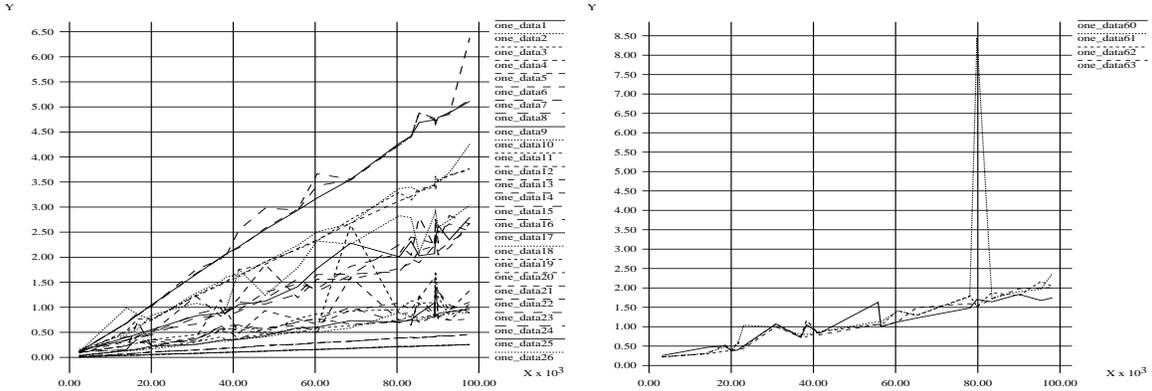


Figure 3: *ORNL-OU: End-to-end delays for large messages. X-axis: message size in bytes; Y-axis: end-to-end delay in seconds.*

3 Networks of Daemons

In this section, we provide an analytical justification for the daemons in terms of performance guarantees, and for the design of multiple paths.

3.1 Network Model

A network of daemons is represented by a graph $G = (V, E)$ with n nodes and m links. Here each node represents a daemon and link represents a communication channel such as TCP connection. A node does not necessarily correspond to an Internet router but to a host where a daemon can be run. A message of size r must be transmitted from a source node s to a destination node d , which incurs three types of delays:

- (a) *Link Delay:* For each link $e = (v_1, v_2)$, there is a *link-delay* $d(e) \geq 0$ such that the leading edge of a message sent via e from node v_1 at time t will arrive at node v_2 at time $t + d(e)$.
- (b) *Bandwidth Constrained Delay:* Each link $e \in E$ has a deterministic “effective” *bandwidth* $b(e) \geq 0$. Once initiated, a message of r units can be sent along link e in $g(r, b(e)) + d(e)$ time, where $g(r, b)$ is non-decreasing in r and non-increasing in b . For a simple bandwidth constraint, we have $g(r, b(e)) = r/b(e)$.
- (c) A message of size R arrives at the source s according to an *unknown* distribution P_R . At any node v , Q_v and R_v are the random variables denoting the queuing delay and message size distributed according to *unknown* distributions \mathbf{P}_{Q_v} and \mathbf{P}_{R_v} , respectively. No information about the distributions of R and Q_v , $v \in V$, is available. Instead, the *measurements* $(Q_{v;1}, R_{v;1}), (Q_{v;2}, R_{v;2}), \dots, (Q_{v;l}, R_{v;l})$ that are independently and identically distributed (iid) according to the distribution \mathbf{P}_{Q_v, R_v} , are known at each node $v \in V$.

We performed end-to-end delay measurements using daemons distributed on the Internet. In Figure 3, we consider messages with widely ranging sizes between ORNL and a number of other universities in the left figure, where each cluster corresponds to a single destination. In the right, we show the delays between ORNL and University of Oklahoma (OU). As portrayed by these measurements, our model captures the essence of end-to-end delays: in each plot, the “slope” corresponds to the bandwidth and the additional variation corresponds to Q_v . Note that in our model the “effective” bandwidth plays a key role and hence is explicitly accounted for, and the rest of the delays can have any distribution (including fractal or self-similar).

3.2 Performance Guarantees

Consider a path P , from source $s = v_0$ to destination $d = v_k$, given by $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$, where $(v_j, v_{j+1}) \in E$, for $j = 0, 1, \dots, (k-1)$. The *bandwidth* of this path is $b(P) = \min_{j=0}^{k-1} b(e_j)$, and the *delay due to bandwidth* is given by $g(r, b(P))$. The *delay* of this path is $b(P) = \sum_{j=0}^{k-1} D(e_j)$. The *end-to-end delay* of path P in transmitting a message of size R is

$$T(P, R) = g(R, b(P)) + d(P) + \sum_{j=0}^{k-1} Q_{v_j|R}, \quad (3.1)$$

where $Q_{v_j|R}$ is the conditional delay at node v_j given that a message of size R arrived at the node. The *expected end-to-end delay* of path P for the given message size R is given by

$$\bar{T}(P, R) = g(R, b(P)) + d(P) + \sum_{j=0}^{k-1} \int Q_{v_j} dP_{Q_{v_j|R}}, \quad (3.2)$$

which is a random variable (of R) for a fixed path P . Let \mathcal{P} denote the set of all paths from s to d . Let P_R^* denote a path with the minimum expected end-to-end delay for the given message size R such that $\bar{T}(P_R^*, R) = \min_{P \in \mathcal{P}} \bar{T}(P, R)$. If the error distributions are known, P_R^* can be computed using deterministic optimization methods. Such an approach is infeasible here since the distributions are not known. We compute an estimator \hat{P}_R of P_R^* using a regression estimator such that

$$\mathbf{P} \left\{ \mathbf{E}_R[\bar{T}(R, \hat{P}_R) - \bar{T}(R, P_R^*)] \geq \epsilon \right\} \leq \delta, \quad (3.3)$$

for a sufficiently large sample size, which depends on ϵ, δ, n , and a suitably chosen function family \mathcal{Q}_v that contains the regression function. Informally, this condition guarantees that: *the expected delay of \hat{P}_R is within ϵ of that of P_R^* with probability $1 - \delta$, irrespective of the delay distributions.* Detailed derivations can be found in [4].

This guarantee is possible because the measurements are the actual delays experienced by the messages based on TCP streams. Traditionally, ICMP-based mechanisms such as ping and traceroute are used to collect measurements, and end-to-end guarantees as in (3.3) cannot be provided based on such data if paths go through firewalls. Some firewalls disable responses to ping and traceroute and sometimes deliberately send incorrect responses. Also, some firewalls enforce rate controls on ICMP traffic but not on TCP, in which case the delay measurements based on ping and traceroute could be highly misleading. Thus our approach not only provides analytical guarantees but also provide guidance for appropriate measurements.

Given that a suitable regression estimator has been selected, an algorithm was presented in [4] to compute the best empirical path \hat{P}_R . The complexity of this algorithm is $O(m^2 + mn \log n + nf(l))$, where $f(l)$ is the complexity of computing the regression at a given value r . Thus, a polynomial-time (in l) regression estimator results in a polynomial-time (both in n and l) path computation method.

3.3 Multiple Path Computation

A *multiple path* from s to d , denoted by MP , consists of a set of simple bandwidth-disjoint paths from s to d . Consider a network $G = (V, E)$ with zero queuing delays $Q_v = 0$ for all $v \in V$ and $g(r, b) = r/b$. Now a message of r units can be sent along the edge P in $r/B(P) + D(P)$ time.

This model is a first order approximation to the measurements shown in Figures 2. The *end-to-end delay*, denoted by $T(MP, r)$, of a multiple path MP from s to d is defined as the time required to send a message of size r from s to d , wherein the message is subdivided and transmitted via the constituent paths. Multiple paths often provide more bandwidth compared to single paths which can result in smaller end-to-end delay. But to minimize the end-to-end delays, the message must be suitably divided into parts to be routed via different paths as illustrated in the next example.

Consider a network consisting of two paths P_1 and P_2 . Let $B_1 = 10$ units/sec, $B_2 = 20$ units/sec, $D_1 = 2$ sec, and $D_2 = 12$ sec. Consider a message of size $r = 100$ units. Then $T(P_1, 100) = 12$ and $T(P_2, 100) = 17$. If a single path is to be used, P_1 will be chosen for this message size. On the other hand, let us say that 99 units are sent on P_1 and 1 unit is sent on P_2 ; the corresponding delays are given by $99/10 + 2 = 11.9$ sec and $1/20 + 12 = 12.05$ sec respectively, resulting in an end-to-end delay of 12.05 seconds. Clearly, it is advantageous *not* to use the two-path $\{P_1, P_2\}$ for this message size. Now consider a message of size $r = 1000$ units. The end-to-end delays of individual paths P_1 and P_2 are 102 sec and 62 sec, respectively. Thus if a single path is to be used, then P_2 will be chosen. Consider using a two-path $\{P_1, P_2\}$ for $r = 1000$ such that 400 and 600 units are sent via P_1 and P_2 respectively, resulting in the individual delays of 42 sec for each path; hence the resultant end-to-end delay is 42 sec which is smaller than that of P_1 or P_2 (given by 102 and 62 sec, respectively).

For two paths P_1 and P_2 , we have $T(\{P_1, P_2\}, r) \leq \min\{T(P_1, r), T(P_2, r)\}$ if and only if the condition $C(P_1, P_2)$ given by

$$D(P_1) + r/B(P_1) \geq D(P_2) \quad \text{and} \quad D(P_2) + r/B(P_2) \geq D(P_1)$$

is satisfied. Under this condition, the minimum end-to-end delay of $\{P_1, P_2\}$ is achieved by dividing the message into two parts of sizes r_1 and r_2 , $r_1 + r_2 = r$, to be sent via P_1 and P_2 respectively. The sizes of the parts are given by

$$r_1 = \frac{B_1 r}{B_1 + B_2} + \frac{B_1 B_2 (D_2 - D_1)}{B_1 + B_2} \quad \text{and} \quad r_2 = \frac{B_2 r}{B_1 + B_2} - \frac{B_1 B_2 (D_2 - D_1)}{B_1 + B_2}.$$

Let $\{P_1, P_2, \dots, P_p\}$ be the paths from which a multiple path with low end-to-end delay must be constructed. Let $D_i = D(P_i)$ and $B_i = B(P_i)$ for $i = 1, 2, \dots, p$ such that $D_1 < D_2 < \dots < D_p$. Let $\xi_i \in [0, 1]$ denote the fraction of r routed via path P_i . Then the end-to-end delay of a multipath with $\xi = (\xi_1, \xi_2, \dots, \xi_p)$ is specified as follows.

Lemma 3.1 Let $\bar{B}_j = \sum_{i=1}^j B_i$ and $\bar{D}_j = \frac{\sum_{i=1}^j B_i D_i}{\sum_{i=1}^j B_i}$. The minimum end-to-end delay is given by

$$T^*(r) = \begin{cases} \frac{r}{B_1} + \bar{D}_1 & \text{if } 0 < r \leq \bar{B}_1(D_2 - \bar{D}_1) \\ \frac{r}{B_2} + \bar{D}_2 & \text{if } B_1(D_2 - \bar{D}_1) < r \leq B_1(D_3 - \bar{D}_2) \\ \dots & \dots \\ \frac{r}{B_j} + \bar{D}_j & \text{if } \bar{B}_{j-1}(D_j - \bar{D}_{j-1}) < r \leq \bar{B}_j(D_{j+1} - \bar{D}_j) \\ \dots & \dots \end{cases}$$

corresponding to $\xi_i^* = B_i(T^* - D_i)/r$. \square

Theorem 3.1 For a sequence of paths $P_{i+1}, P_{i+2}, \dots, P_{i+k}$, $i, i+k \leq q$, let $MP_{i+j} = \{P_{i+1}, P_{i+2}, \dots, P_{i+j}\}$. Consider that $C(MP_{i+j}, P_{i+j+1})$ is true for all $1 \leq j < q - i$. Then the optimal

end-to-end delay of the multipath $\{P_{i+1}, P_{i+2}, \dots, P_{i+k}\}$ is given by

$$\frac{r}{B_{i+1} + B_{i+2} + \dots + B_{i+k}} + \frac{B_{i+1}D_{i+1} + B_{i+2}D_{i+2} + \dots + B_{i+k}D_{i+k}}{B_{i+1} + B_{i+2} + \dots + B_{i+k}}.$$

The constituent paths of a multiple path for a given message size are computed by repeatedly computing the quickest path and removing it from the graph by reducing the appropriate bandwidths of the links. Then the extracted quickest paths are combined using the algorithm Disjoint MTA in $O(p \log p)$ time, for p paths. The resultant multiple path is not always guaranteed to be optimal in a strict sense because of the additional unaccounted randomness in the delays. But this path yielded very good results in actual implementations as shown in the next section. Details of Theorem 3.1 and Lemma 3.1 are provided in the fuller version of this paper.

algorithm disjoint MTA

1. sort the set of paths according to their delays;
 2. compute shortest delay path P_1 ;
 3. $MP \leftarrow \{P_1\}$;
 4. **while** for the next shortest delay path P the condition $C(MP, P)$ is true **do**
 5. $MP \leftarrow MP \cup \{P\}$;
-

Algorithm Disjoint MTA. Algorithm for solving disjoint MTP.

4 Experimental Results

The network shown in Fig. 4 is utilized in our implementation. The delay regression estimation is based on potential function method as in [6]. The server is located at OU and client is located at ORNL. Daemons are executed as user-level socket-based codes at the sever and client, and at two additional locations ¹ at Louisiana State University (LSU) and Old Dominion University (ODU). Typical experimental results are shown in Fig. 4 in the right. The upper curve represents the delays in a single TCP stream ORNL-OU as is usually done. The lower curve corresponds multiple path consisting of TCP streams ORNL-ODU-OU, ORNL-LSU-OU and two direct parallel TCP streams ORNL-OU. The messages are divided into four parts as per the delay curves of the paths as described in the previous section, and are sent along the respective paths. Note that the overall the end-to-end delay is much lower when daemons are employed, except for some smaller sizes. The multiple paths resulted in an average improvement of about 35% in the end-to-end delay in all the cases studied by us. The details of the experimentation and more extensive measurements are provided in the fuller version of the paper. To our knowledge, ours is the first implementation of physically diverse multiple paths over the Internet (without additional support from the Internet routers) that achieve concrete reductions in the end-to-end delays. Note that parallel TCP streams do not necessarily implement physically diverse paths, and methods such as DiffServ and MPLS require support from the network routers. We believe that further improvements of our method are possible when the daemon implementations are more customized to the host sites.

¹The daemons, being user-level programs, can be executed at free telnet sites at various locations on the Internet.

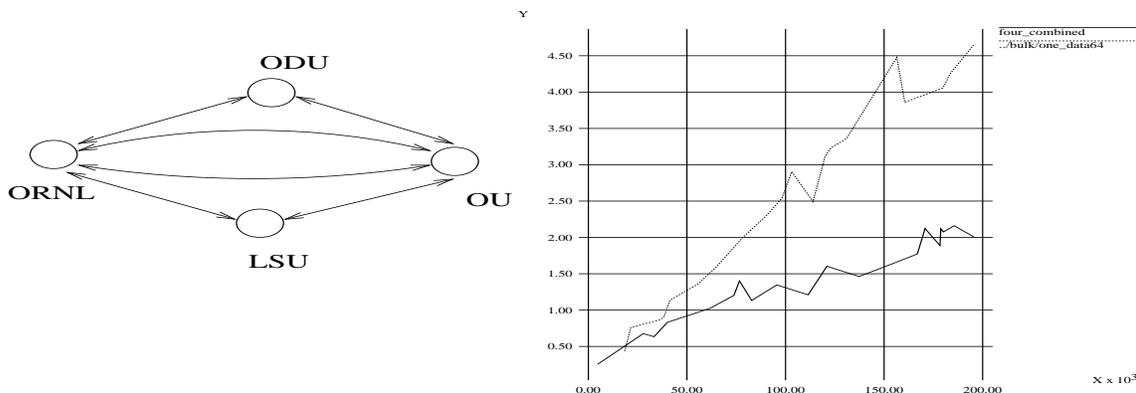


Figure 4: *ORNL-OU: End-to-end delays for large messages. X-axis: message size in bytes; Y-axis: end-to-end delay in seconds.*

References

- [1] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, M. Stemm, and R. H. Katz. TCP behavior of a busy Internet server: Analysis and improvements. In *Proceedings of INFOCOM*. 1998.
- [2] G. Huston. *Internet Performance Survival Guide: Basic Mechanisms and Directions*. John Wiley and Sons, 2000.
- [3] W. Matthews and L. Cottrell. The PingER project: Active Internet performance monitoring for the NENP community. *IEEE Communications Magazine*, pages 130–136, May 2000.
- [4] N. S. V. Rao. End-to-end delay guarantees in computer networks: Analysis and NetLet implementation, 2000. ORNL Draft, <http://saturn.epm.ornl.gov/~nr Rao>.
- [5] N. S. V. Rao. NetLets: End-to-end QoS mechanisms for distributed computing in wide-area networks using two-paths. In *Proceedings of International Conference on Internet Computing*, 2001. ORNL Draft, <http://saturn.epm.ornl.gov/~nr Rao>.
- [6] N. S. V. Rao and S. G. Batsell. On routing algorithms with end-to-end delay guarantees. In *IC3N: International Conference on Computer Communications and Networks*, pages 162–167. 1998.
- [7] N. S. V. Rao, S. Radhakrishnan, and B- Y. Bang. NetLets: Measurement-based routing for end-to-end performance over the Internet. In *Proc. IEEE Int. Conf. on Networking*, 2001.
- [8] D. Salamoni and S. Luitz. High-performance throughput tuning/measurements. In *PPDG Collaboration Meeting*. 2000.
- [9] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vandat, G. Voelker, and J. Zahorjan. Detour: Informed internet routing and transport. *IEEE Micro*, pages 50–59, January-February 1999.