

NetLets for End-To-End Delay Minimization in Distributed Computing Over Internet Using Two-Paths *

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

International Journal of High Performance Computing Applications, 2002, in press

Abstract

In a number of distributed computing applications, messages must be transmitted on demand between processes running at different locations on the Internet. The end-to-end delays experienced by the messages have a significant “random” component due to the complicated nature of network traffic. We propose a method based on delay-regression estimation to achieve low end-to-end delays for message transmissions in distributed computing applications. Two-paths are realized between various communicating processes in a transparent manner. Our scheme is implemented over the Internet by a network of NetLets, which communicate with one another to maintain an accurate “state” of delay-regressions in the network. NetLets handle all network traffic between the processes and also perform routing at a certain level depending on the underlying network. We present experimental results to illustrate that NetLets provide a viable and practical means for achieving low end-to-end delays for distributed computing applications over the Internet.

1 Introduction

Several distributed computing processes require the massive computational power and/or diversity of capabilities distributed over a network, which can be the Internet or a dedicated wide-area sensor network. Such processes may range from distributed software sensors that collect network data (e.g. as network flows, delays, etc.) to hardware systems consisting of geographically dispersed remote imaging or visual sensors. These applications require on-demand communication between various processes distributed over the network. In general, both the nature and volume of data that needs to be communicated could be quite varied among the applications as well as within a single application. Typically, the synchronization messages could be small whereas data transfers could be quite voluminous. In most existing solutions, the messages are typically sent as single TCP streams. Such solutions do not in general exploit the network to provide the best possible end-to-end delay.

In the present networks, such as the current Internet, the messages are decomposed into datagrams and sent via various routers as per the IP paradigm. At the routers the incoming datagrams

*Research sponsored by 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. A preliminary version of the results of this paper have been presented at the International Conference on Internet Computing, 2001.

from all sources are serviced in a best-effort paradigm. 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 *NetLets* that perform delay estimation and two-path computation. Depending on the underlying network, various levels of routing are also performed by NetLets. For the Internet, NetLets perform source-based routing via the other NetLets. NetLets run as daemon processes and exchange measurements among themselves. Data to be transmitted over the network will be handed over to the local NetLets by the computation process. In a nutshell, NetLets perform all the networking work needed to achieve the end-to-end performance, and thereby relieve the user from having to explicitly account for networking. Our main emphasis is on *two-paths* which provide several advantages compared to single paths.

NetLets provide networking performance and/or guarantees beyond any of the currently available mechanisms. A number of QoS projects [12], in particular the Detour project [11], propose concepts similar to ours. But, our system differs in a number of ways: (i) our solution is based on explicitly realizing two-paths for data transmission, (ii) our method is analytically justified under very general conditions (Section 3.2), and (iii) preliminary working implementations of NetLets on the Internet (Section 4), local area networks [8], and simulation [5] show very promising results. Compared to an earlier implementation [8], the present work provides stronger analytical guarantees based on more refined measurements and estimation. Another related work is the parallel TCP streams for large memory transfers [10, 4], or web traffic [1], wherein the objective is to achieve high “effective” bandwidth data transfers. Our work differs in providing analytical guarantees, and our streams, routed via other NetLets, achieve more spread-out traffic compared to parallel TCP streams.

The concept of NetLets is described in Section 2. Theoretical results that underpin NetLets for two-paths are described in Section 3. Experimental results based on NetLets implementation on the Internet are presented in Section 4.

2 NetLets and Applications

The NetLets are designed to perform all the network related activities in a manner that relieves the communicating processes from accounting for the details of the network. NetLets run as daemons at the processing nodes and communicate over the network to perform three main functions shown in Figure 1: (a) measurement and state estimation (b) path computation for message transmission, and (c) routing at a suitable level. The actual realization of the computed paths in a network depends on the exact details of the application as well as the type of routing provided by the underlying network. At present, the routing paths for datagrams in IP networks are decided by the best effort method. Very little support for source-based routing is provided by the Internet routers, which are controlled by service providers. In our scheme, the messages between the NetLets are source-routed via the other Netlets ¹.

Consider a simple scenario of distributed computing over the Internet, where a computational task is distributed over several nodes. Messages are communicated between the nodes as per the

¹Present Internet is one of the most challenging scenarios for implementing NetLets. Our implementation is made possible by cooperative agreements with universities that run NetLets on their hosts. NetLets, however, can also be run on the free telnet sites that are available across the Internet.

task structure as in Figure 2(a). For example, the communication between the processes P_1 and P_2 could be handled by a process-to-process TCP connection. In the present methods, the same TCP stream is utilized to send all the messages between them – the individual datagrams, however, could travel via different physical paths. If this connection happens to have high traffic so that delay or the packet loss is high, no rerouting action is taken by TCP.

Consider that NetLets are executed at each of the processes as shown in Figure 2(b). The processes communicate with others through the NetLets which compute the best path for a message of given size. NetLets maintain peer-to-peer connection among themselves as shown in Figure 2(b). Furthermore, each NetLet maintains estimates of various delays as regression functions of the message size. If the IP connection between P_1 and P_2 via the routers R_1 and R_2 has high delays, then the NetLet at P_1 uses its regression to compute an alternate path such as R_1 - R_3 - R_2 , which is utilized by sending the message to the NetLet at R_3 instead of R_2 . Furthermore, if a large message has to be sent, a two-path consisting of $R_1 - R_4 - R_2$, and $R_1 - R_3 - R_2$, may be utilized in parallel.

3 Networks of NetLets

In this section, we provide an analytical justification for the design of the NetLets in terms of performance guarantees and design of two-paths. Detailed derivations for the performance guarantees can be found in [5].

3.1 Network Model

A network of NetLets is represented by a graph $G = (V, E)$ with n nodes and m edges or links. Here each node represents a NetLet and link represents a communication channel such as TCP connection (as in the previous section). It is important to note that a node does not necessarily correspond to an Internet router but a host where NetLet daemon can be run. A message of size r must be transmitted from a source node s to a destination node d . A message 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. Let \mathbf{P}_{Q_V} denote the joint distribution of Q_{v_1}, \dots, Q_{v_n} for all $v_1 \dots v_n \in V$. For any given message size $R_v = r$, let $\bar{q}_v(r)$ denote the *regression* of Q_v onto R such that $\bar{q}_v(r) = E[Q_v | R_v = r]$. 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 NetLets distributed on the Internet. In Figure 3, the measurements are based on sending small messages. The lower plot represents a TCP connection between between Oak Ridge National Laboratory (ORNL) and University of Tennessee

(UTK), and the top plot corresponds to a connection between ORNL and Old Dominion University (ODU). The physical distance between sites for first and second connections are approximately 20 and 500 miles, respectively. In Figure 4, we consider messages with widely ranging sizes between ORNL and University of Oklahoma (OU) in the left figure, and ORNL and University of California at Riverside (UCR) in the right figure. In terms of physical distance, the sites of first and second pairs are approximately 1000 and 2500 miles apart, respectively. Although the mode and type of communication is quite different in the above cases, our model captures the essence: in each plot, the “slope” corresponds to the bandwidth and the additional variation corresponds to Q_v .

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 *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 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 Q_v that contains the regression \bar{q}_v . 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 distributions \mathbf{P}_R and \mathbf{P}_{Q_v} .*

We wish to emphasize here that this guarantee is possible because the measurements correspond to the actual delays experienced by the messages – stated otherwise, the measurement mechanism is same as the one used for actually sending the messages. Traditionally, mechanisms such as ping and traceroute are used to collect measurements, and it is unclear if end-to-end guarantees such as in Eq (3.3) can be provided based on such data. The difficulty here is that the communication mechanism used by the distributed processes can be quite different from that used in ping or traceroute, and there is no simple way of relating the two. Also note that due to firewalls, in some cases ping and traceroute responses may be disabled, controlled (e. g. ICMP rate control), or deliberately set to incorrect values.

The expected end-to-end delay of Eq (3.2) for the message size R can be written as

$$\bar{T}(P, R) = g(R, b(P)) + d(P) + \sum_{j=0}^{k-1} \bar{q}_{v_j}(R),$$

which is a random variable since it is a function of R distributed according to P_R . Let $\hat{q}_v(R) \in \mathcal{Q}_v$ be an estimator of the regression $\bar{q}_v(R)$ computed using the measurements. Based on the estimator $\hat{q}_v(\cdot)$, we consider the *empirical end-to-end delay* given by

$$\hat{T}(P, R) = g(R, b(P)) + d(P) + \sum_{j=0}^{k-1} \hat{q}_{v_j}(R). \quad (3.4)$$

Let the *empirical best path* based on the regression estimate be given by $\hat{P}_R = \arg \min_{P \in \mathcal{P}} \hat{T}(P, R)$. Since \hat{q}_v is based only on the measurements and \mathcal{P} is finite, the best empirical path \hat{P}_R can be computed entirely based on the sample.

The performance of \hat{P}_R is related to that of \hat{q}_v in a simple and elegant manner *both* in terms of sample and computational complexities [5]: the condition (3.3) can be assured by ensuring

$$\mathbf{P} \{ \mathbf{E}_R[\hat{q}_v(R) - \bar{q}_v(R)] > \epsilon_1 \} \leq \delta_1,$$

for suitably chosen ϵ_1 and δ_1 , and thus relating the sample complexity of the regression estimator to that of \hat{P}_R . The sample complexities for a number of estimators including vector space methods, feedforward neural networks and monotone regressions are derived in [5].

3.3 Single Path Computation

Given that a suitable regression estimator has been selected, we now present an algorithm [5] to compute the best empirical path \hat{P}_R . We define an *augmented delay* of an edge $e = (v_1, v_2)$ as $d_A(e) = d(e) + \hat{q}_{v_1}(r)$. Let b_1, b_2, \dots, b_c denote the distinct values of the bandwidths $b(e)$, $e \in E$. Let $G(a) = (V, E(a))$ denote the subnetwork where $e \in E(a)$ if and only if $b(e) \geq a$. Let a $s - d$ *shortest path* in $G(a)$ denote the shortest augmented delay path based *only* on the augmented delay of the edges (i.e. with minimum $d_A(\cdot)$ value). Our algorithm Min-Path is based on the algorithm of [9] which was originally proposed for the special case when $\hat{q}_v(r) = 0$, and $g(r, b(P)) = r/b(P)$. Let $\hat{q}(P, r) = \sum_{v \in P} \hat{q}_v(r)$, which yields $d_A(P) = d(P) + \hat{q}(P, r)$.

algorithm Min-Path(r)

1. for $j = 1, 2, \dots, c$, compute $s - d$ shortest path P_j in $G(b_j)$;
 2. compute index k which minimizes $\{g(r, b(P_j)) + d(P_j) + \hat{q}(P_j, r) | j = 1, 2, \dots, c\}$;
 3. return P_k as the path with the minimum end-to-end delay;
-

Algorithm 1. *Computation of 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 $\hat{q}(r)$ at a given value r . Thus, a polynomial-time (in l) regression estimator results in a polynomial-time (both in n and l) routing method.

3.4 Two-Paths

A *two-path* from s to d , denoted by MP , consists of two simple 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 3 and 4. The *end-to-end delay*, denoted by $T(MP, r)$, of a two-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.

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. More generally we have: for $r > 200$, P_2 is the choice; for $r < 200$, P_1 is the choice; and for $r = 200$ either can 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).

In general, the message size determines if a two-path $\{P_1, P_2\}$ achieves a lower delay. We have

$$T(MP, r) \leq \min\{T(P_1, r), T(P_2, r)\}$$

if and only if $D(P_1) + r/B(P_1) \geq D(P_2)$ and $D(P_2) + r/B(P_2) \geq D(P_1)$ [7]. This condition can be used to check if a two-path is better than a single path for a given message size r .

The two-paths are computed as follows. First the quickest path is computed as in the previous section, then it is removed from the graph by reducing the appropriate bandwidths of the links. Then the next quickest path is computed in the residual graph. Then these two paths are combined using the above equation. While the resultant path (single or two-path) is not guaranteed to be an optimal two-path in a strict sense, this method yielded very good results in actual implementations as shown in the next section.

4 Experimental Results

The network shown in Fig. 5 is utilized in our preliminary implementation of NetLets, which illustrates the practical effectiveness of the proposed method. The delay regression estimation is based on potential function method as in [6]. We consider two distributed computing scenarios involving the exchange of: (a) small size messages with a few tens of bytes, and (b) large messages with widely varying sizes in the range of few bytes to megabytes. The first case is typical in web-based database query servers and web instrument control, and the second case is typical of remote visualization and distributed simulations.

For the first case, our results are based on three NetLets system shown in Figure 5(c), where the server is located at OU and client is located at ORNL. The end-to-end delays incurred in sending and receiving a number of queries is represented on Y-axis in Figure 6. The upper curve represents queries processed as a single TCP stream ORNL-OU as is usually done. The lower curve corresponds to a three NetLets system located at ORNL, ODU and OU, wherein two-paths are employed, consisting of direct TCP stream ORNL-OU and one via ODU. The messages are divided into two parts as per the delay curves of both the paths and sent along the respective paths. Note that the overall the end-to-end delay is much lower when NetLets are employed, except for some smaller sizes. The two-paths based on NetLets resulted in an average improvement of about 25% in the end-to-end delay. We believe that more improvements are possible when more than two paths are considered and NetLets are more customized to the sites.

The second case, corresponds to large memory transfers of randomly generated sizes. In Figure 7, we show the most and least favorable cases we observed in experiments conducted over several

days. In each case the data was collected by executing both the programs (with and without NetLets) within seconds of each other. Our results are again for transfers between ORNL and OU as described above. The upper curve corresponds to TCP stream ORNL-OU and the lower curve corresponds to the two-path via the NetLet at ODU. Notice again that the two-path yielded a significant reduction in the end-to-end delay. Similar results were obtained for the other cases (b) and (c) shown in Figure 5, although results were not as significant as in the best case in Fig. 7. Nevertheless, these results show that NetLets hold a promise for significant performance savings in Internet implementations.

5 Conclusions

We presented the concept of NetLets that enable efficient communication between distributed computing processes over wide area networks including Internet. NetLets communicate with each other to compute the delay-regressions of various links, and compute two-paths to send messages. They also perform routing via other NetLets to realize the two-paths. The computing processes simply handover the messages to local NetLets which will deliver them to the destination. NetLets are based on sound finite sample estimation theory in that the paths are computed using measurements without requiring probabilistic traffic models. The end-to-end delays of computed paths are guaranteed to be close to optimal paths with a specified probability. Note that optimal paths are computable only under a complete knowledge of delay distributions, which involves a significantly more difficult task. Our preliminary implementation of NetLets shows that they are a viable method for the Internet, and a more extensive implementation of NetLets is currently underway.

NetLets are upward compatible with a number of next generation networking technologies that are expected to be available in future. At present, there is no easy means of controlling the route taken by a particular datagram. In the next generation networks that enable dynamic route specification, NetLets can specify the planned routes along with the messages. Multiple grade services, such as DiffServe, IntServe, and MPLS [3, 2], are expected to enable messages to be sent under different priorities, with higher priorities incurring higher costs. NetLets can be configured to monitor and utilize the delays under different grades of service. Also any improvements in the underlying TCP implementation, such as auto-tuning, can be transparently exploited by NetLets by simply replacing the existing implementation with a new one. Active networks enable executable code to be sent along with the messages. In such networks, NetLet daemons at the source nodes can send code for routers to collect the measurements and also to assist in routing. In this paper, we assumed local stationarity of the delay distributions. It would be interesting to design NetLets for time-varying distributions using martingale or mixingale methods.

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] B. E. Carpenter and D. D. Kandlur. Diversifying Internet delivery. *IEEE Spectrum*, pages 57–61, Nov 1999.
- [3] A. Dutta-Roy. The cost of quality in Internet-style networks. *IEEE Spectrum*, pages 57–62, September 2000.
- [4] W. Matthews and L. Cottrell. The PingER project: Active Internet performance monitoring for the NENP community. *IEEE Communications Magazine*, pages 130–136, May 2000.

- [5] N. S. V. Rao. End-to-end delay guarantees in computer networks: Analysis and NetLet implementation, 2000. ORNL Draft, <http://saturn.epm.ornl.gov/~nrao>.
- [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 and S. G. Batsell. QoS routing via multiple paths using bandwidth reservation. In *IEEE INFOCOM98: The Conference on Computer Communications*, volume 1, pages 11–18. 1998.
- [8] 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. 184-193.
- [9] J. B. Rosen, S. Z. Sun, and G. L. Xue. Algorithms for the quickest path problem and the enumeration of quickest paths. *Computers and Operations Research*, 18(6):579–584, 1991.
- [10] D. Salamoni and S. Luitz. High-performance throughput tuning/measurements. In *PPDG Collaboration Meeting*. 2000.
- [11] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vanden dat, G. Voelker, and J. Zahorjan. Detour: Informed internet routing and transport. *IEEE Micro*, pages 50–59, January-February 1999.
- [12] Z. Wang. *Internet QoS*. Morgan Kaufman Pub., 2001.

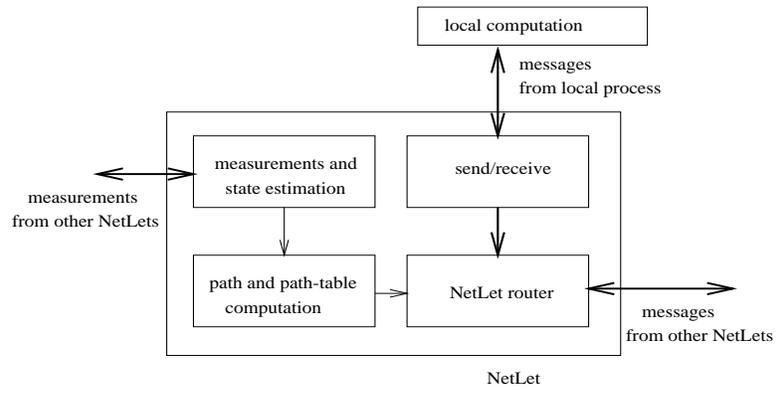


Figure 1: *Functional block diagram of a NetLet.*

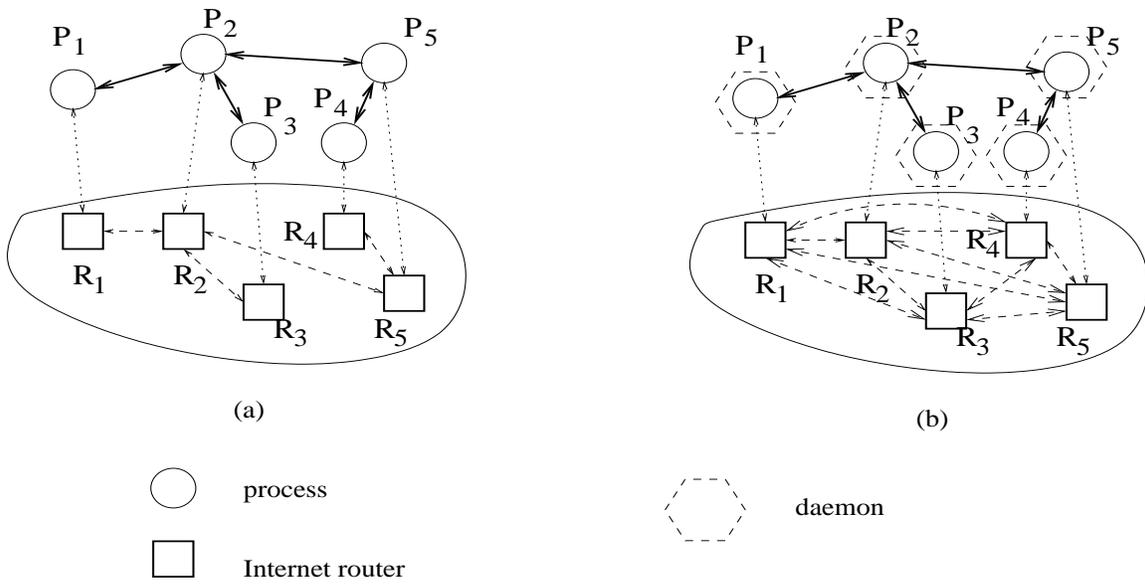


Figure 2: *Typical scenario of distributed computing environment over the Internet.*

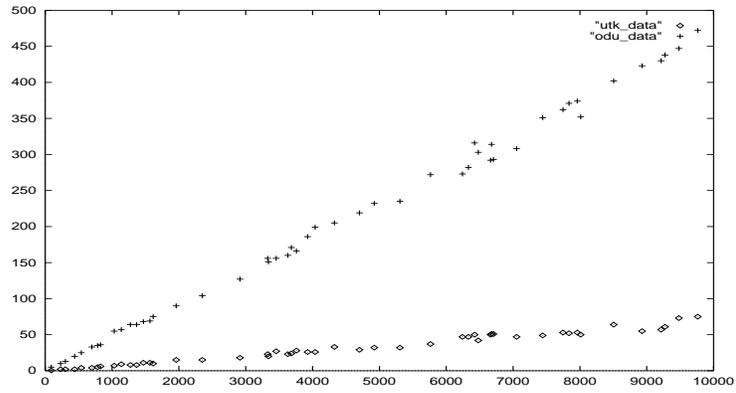


Figure 3: *End-to-end delay measurements for small repeated messages. X-axis: number of messages. Y-axis: end-to-end delay in seconds.*

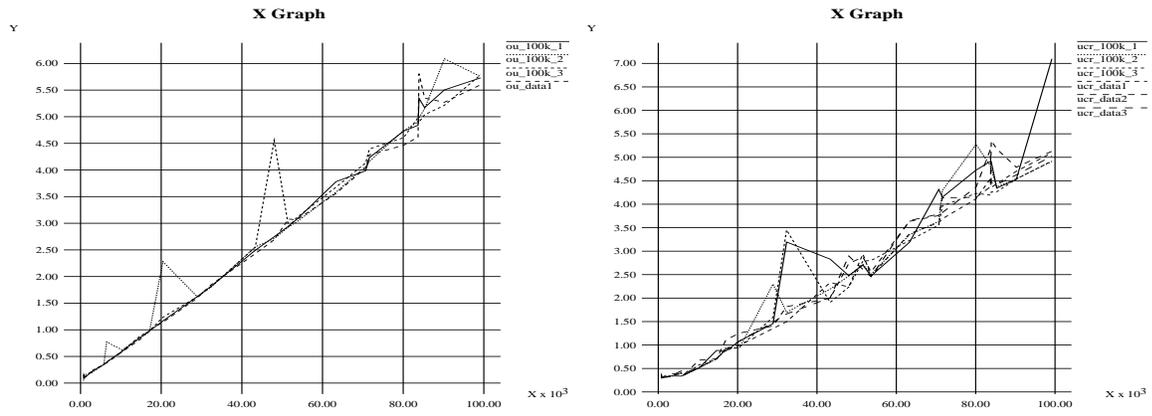


Figure 4: *End-to-end delay measurements for large and varying messages X-axis: message size in bytes. Y-axis: end-to-end delay in seconds.*

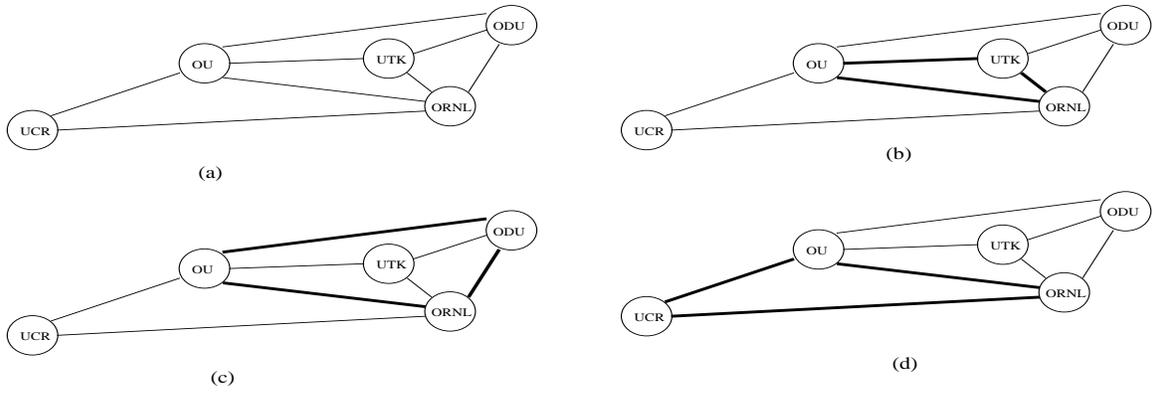


Figure 5: *Network of NetLets.*

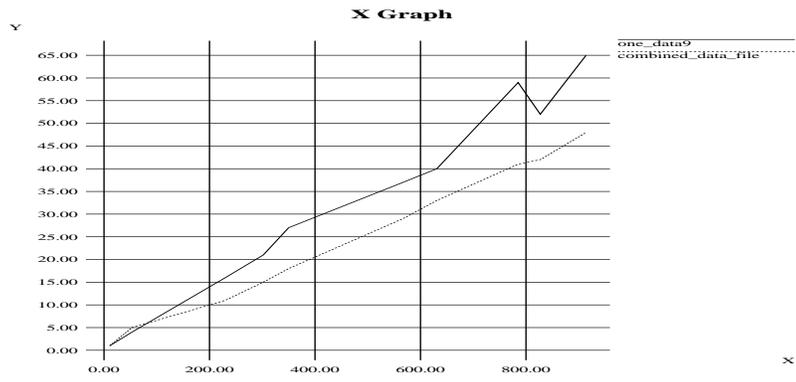


Figure 6: *ORNL-OU: End-to-end delays for small messages. X-axis: number of messages; Y-axis: end-to-end delay in seconds.*

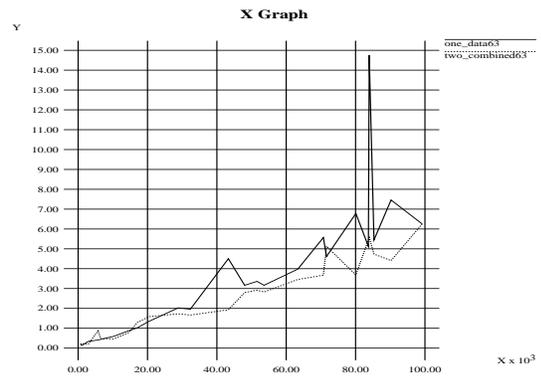
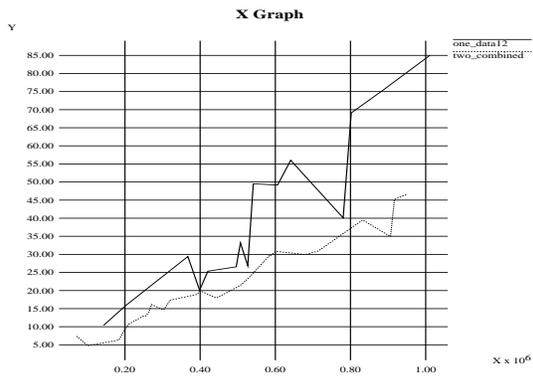


Figure 7: *ORNL-OU: End-to-end delays for large messages. X-axis: message size in bytes; Y-axis: end-to-end delay in seconds.*

Author Contact Information

Nageswara S. V. Rao
Mailstop 6355, Bldg 6010, Rm 220
1000 Bethel Valley Rd.
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6355
raons@ornl.gov
phone: (865) 574-7517
fax: (865) 241-0381

Biography: Nageswara S. V. Rao received B. Tech in electronics and communications engineering from Regional Engineering College, Warangal, India, in 1982, M. E. from School of Automation, Indian Institute of Science, Bangalore, India in 1984, and Ph.D. in computer science from Louisiana State University, Baton Rouge, Louisiana in 1988. He is a Distinguished Research Staff Member at the Computer Science and Mathematics Division, Oak Ridge National Laboratory, where he joined in 1993 as a staff member. He was Assistant Professor in the Department of Computer Science, Old Dominion University, Norfolk, Virginia from 1988 until 1993.

His research interests include multiple sensor systems, computer network protocols, neural networks, fault diagnosis, and adaptive algorithms. He has published over 150 research articles in various journals, conference proceedings and books. He is an editorial board member for the Journal of Franklin Institute.

Running Title: NetLets for End-To-End Delay Minimization