On Measurement-Based Transport Method for Message Delay Minimization Over Wide-Area Networks

Qishi Wu, Nageswara S. V. Rao
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831
Tele: 865-576-5603
Email: {wuqn,raons}@ornl.gov

S. Sitharama Iyengar
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803
Tele: 225-587-1252
Email: iyengar@bit.csc.lsu.edu

Abstract – **Data transport methods with end-to-end performance guarantees are required by the next generation distributed computing applications. Transport methods based on default best-effort IP networks are not capable of providing such guarantees due to complex traffic distributions in wide-area networks and highly dynamic network conditions. In this paper, we present a measurement-based transport method using source routing strategy to minimize message delays via multiple quickest paths. This method is based on a linear approximation using effective bandwidth, and is computationally efficient and also analytically tractable under fairly general conditions. This method is implemented using application-level daemons and tested at Internet nodes. The results illustrate its superior performances over the default TCP/IP method.**

Keywords: transport method, overlay network, network measurements, end-to-end delays

I. Introduction

New generation of large-scale distributed computing applications require unprecedented end-to-end performance levels from long-haul networks. One of the important problems in this area is to minimize the message delays over wide-area networks. In the present wide-area networks, including the Internet, messages are typically decomposed into data packets, which are then forwarded by the routers as per the best-effort IP paradigm. Typically, the packets are routed along paths with minimum number of hops from source to destination. However, such paths are not necessarily uniformly quickest for all message sizes because the "available" path bandwidth and network congestion are also vital factors in determining the delays. In general, the delays experienced by data packets have been shown to have highly complicated statistical distributions, which make the end-to-end performance highly unpredictable. Since an accurate prediction of delay distributions is not feasible over wide-area networks, measurement-based approaches have been developed to identify close-to-optimal quickest paths [9-11]. Furthermore, by utilizing application-level daemons, multiple paths via the daemons can be used to overcome the limitations of the default TCP/IP paradigm. These works establish the feasibility and underlying principles of such an approach but they only provide proof-of-the-concept experimental results. In particular, the analytical methods in [10] are not very practical, whereas the results in [11] are based on simulation and are not analytically justified.

In this paper, we propose a measurement-based transport method and describe its complete functional framework with practical implementations of various components. The proposed method actively collects network measurements to estimate the effective bandwidth and minimum delay for each link using a linear regression estimator, whose probabilistic performance guarantees are shown under fairly general conditions. Based on the measurement results, the proposed method computes multiple quickest paths to achieve low end-to-end message delays using a source-based routing strategy supported by application-level routing daemons. We also describe a systematic approach to optimizing the needed measurements based on the statistical design [4,5].

The rest of the paper is organized as follows. We describe in Section II the general framework of network daemons that estimates link properties and supports message transfers. Section III discusses in detail the technique used for measuring effective bandwidth and minimum delay as well as its performance analysis. Section IV presents the algorithms for computing multiple quickest paths. Section V is devoted to the minimization of measurements. The implementation details and experimental results are given in Section VI.

II. Network Daemon of Measurement-Based Transport Method

We design network transport daemons that provide a measurement-based solution ensuring end-to-end delay performance. The framework of such transport daemons illustrated in Fig. 1 is based in part on earlier works [9-11]. A transport daemon consists of two main components, namely, measurement and transport modules. The measurement module has four functional units for measurement minimization, end-to-end delay measurement, regression estimate of link properties, and routing table construction. The transport module is responsible for setting up a transport channel by propagating the source routing control information along a pre-computed routing path and providing data transfer service to user applications through API functions.

In addition to the refinements in various components, this framework also incorporates the measurement optimization compared to the earlier works [9-11]. In general, not all transport daemons deployed at available nodes are activated in order to save computing and communication resources. Instead, we strategically select a subset of nodes to conduct link measurements at optimal measuring rates that are statistically determined. The measurement results are broadcast over the entire overlay network of transport daemons so that every node is able to build the network topology and perform source routing via multiple quickest paths for its applications.

During message transmission, the daemons operate in one of three modes: sending, routing, and receiving. A sending node reads data from user applications and prepares the source-routing header information. A receiving node forwards incoming data to host applications and handles the acknowledgements if necessary. A routing node acts as a virtual router at the application level providing routing functions

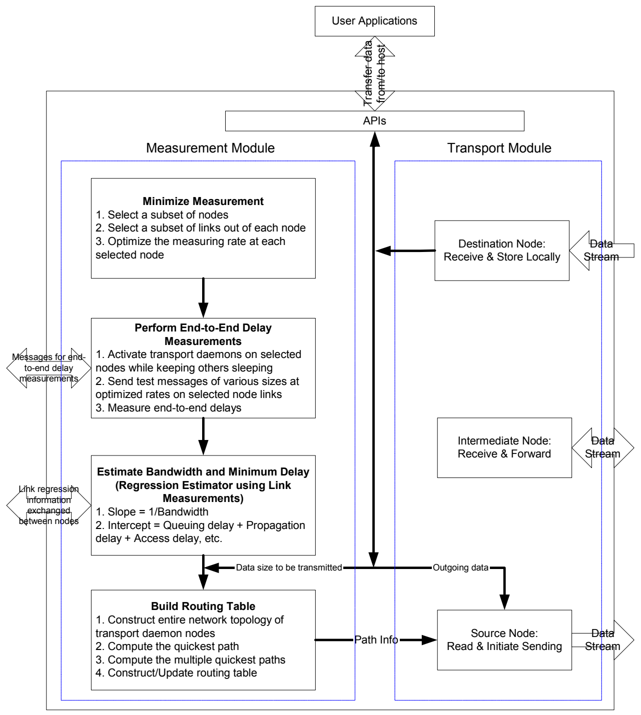

User Applications
Transfer data from/to host
APIs
Measurement Module
Transport Module
Minimize Measurement
1. Select a subset of nodes
2. Select a subset of links out of each node
3. Optimize the measuring rate at each selected node
Perform End-to-End Delay Measurements
1. Activate transport daemons on selected nodes while keeping others sleeping
2. Send test messages of various sizes at optimized rates on selected node links
3. Measure end-to-end delays
Messages for end-to-end delay measurements
Estimate Bandwidth and Minimum Delay (Regression Estimator using Link Measurements)
1. Slope = 1/Bandwidth
2. Intercept = Queuing delay + Propagation delay + Access delay, etc.
Link regression information exchanged between nodes
Data size to be transmitted
Outgoing data
Build Routing Table
1. Construct entire network topology of transport daemon nodes
2. Compute the quickest path
3. Compute the multiple quickest paths
4. Construct/Update routing table
Path Info
Destination Node: Receive & Store Locally
Intermediate Node: Receive & Forward
Source Node: Read & Initiate Sending
Data Stream
Data Stream
Data Stream

Fig. 1. Framework of measurement-based transport daemon

III. BANDWITH AND DELAY MEASUREMENT

The link bandwidth is the fastest rate at which data can be generated and sent along the link, while the available link bandwidth is the spare bandwidth of the link "left over" after the cross traffic [2,3]. Due to complex traffic distribution over wide-area networks, and the non-linear nature of transport protocol dynamics (in particular TCP), the throughput achieved in actual message transfers is typically different from both the link and available bandwidths, and typically contains a random component The *effective path bandwidth* is defined as the maximum throughput the path provides to a flow given the current cross traffic load on the path. The notion of effective bandwidth is specific to the transport protocol employed by the transport daemon, and is related to both link and available bandwidth perhaps in a complicated way. The active measurement technique we apply here is to estimate the effective path bandwidth and minimum delay for each virtual link.

There are three main types of delays involved in the message transmission over computer networks, namely, link propagation delay d_p imposed at the physical link level, equipment-associated delay d_q mostly incurred by processing and buffering at the hosts and routers, and bandwidth-constrained delay d_{BW}. Due to the time-varying network cross traffic, the delay d_q often experiences a high level of randomness. Also, since the transport protocol reacts to the competing traffic on the links, the delay d_{BW} may also exhibits randomness particularly over congested wide-area connections. We have the following expression for the end-to-end delay or message delay in transmitting a message of size r on a path P with l links or hops:

$$d(P,r) = d_{BW}(P,r) + \sum_{i=1}^{l}(d_{p,i}(P) + d_{q,i}(P,r)) \tag{1}$$

Note that d is a random variable for fixed P and r. If the message size r is smaller than the smallest MTU on the path, and the network is lightly loaded, the bandwidth-constrained delay in Equation (1) is negligible so that the sum of the queuing and propagation delays mostly account for the end-to-end delay[1]. We denote this lower bound of message delay by a fixed quantity: $d_{\min}(P) = \sum_{i=1}^{l}(d_{p,i}(P) + d_{q,i}(P,r)), \quad r < MTU$.

On the other hand, if the message size r is considerably large, the message delay is mainly contributed by the bandwidth-constrained delay together with a somewhat smaller but random quantity d_q. Let $EBW(P)$ denote the effective bandwidth of the path P. The message delay $d(P,r)$ for transmitting a message of size r can be approximated by a linear model:

$$d(P,r) \approx \frac{1}{EBW(P)} r + d_{\min}(P) \tag{2}$$

In a circuit-switching connection such as a light path or a dedicated bandwidth channel, the maximum transmission rate is fixed and is determined by the minimum effective link bandwidth $EBW(link)$ of the path P

$$EBW(P) = \min_{link \in P}\{EBW(link)\}. \tag{3}$$

In a purely packet-switching network wherein all data packets are stored and forwarded at intermediate nodes. Thus, the bandwidth-constrained delay needs to be counted at every component link. For the transmission of a packet of length less than path MTU, the effective path bandwidth is approximated in [11] as:

$$EBW(P) = \frac{1}{\sum_{link \in P} \frac{1}{EBW(link)}} \tag{4}$$

However, when transmitting a message of large size in a packet-switching network, the pipelining of data packets along component links actually makes the effective path bandwidth practically close to Equation (3).

We use an active measurement technique to estimate the effective bandwidth of a link or a physical path in an overlay network. A measurement node generates a set of test messages of various sizes, sends them over an outgoing link through a

[1] From the transport layer's viewpoint, the minimum delay for a message smaller than the path MTU may also contain a small component introduced by the timeout a transport protocol uses to wait for more data from applications. This waiting period is usually dependent on the implementation of a transport protocol.

transport channel such as a TCP flow, and measures the end-to-end transmission delay. This process is repeated several times for each message size and the average delay is calculated. Once the average end-to-end delays for messages of different sizes are determined on an outgoing link, we apply a linear regression to fit the measured points of message size r and end-to-end delay d pair. The first order approximation of the effective path bandwidth EBW and the minimum end-to-end delay $d_{\min}$ are then estimated by the slope and intercept of the regression line $d = \frac{1}{EBW} r + d_{\min}$, respectively. It is worth pointing out that the intercept of the regression line is sometimes sensitive to the message transmission delay measurements. We replace the estimate of $d_{\min}$ with a UDP-based minimum end-to-end delay measurement if the intercept does not yield a valid value.

The message delay measurements between two transport daemons deployed at Louisiana State University (LSU) and Oak Ridge National Laboratory (ORNL) as well as its corresponding linear regression estimate are illustrated in Fig. 2. The measurement of the transmission delay for each message size is carried out three times. From this figure, we estimate that the effective path bandwidth EBW of this virtual link to be about 1.0 Mbps and the minimum message delay to be about 35 ms.

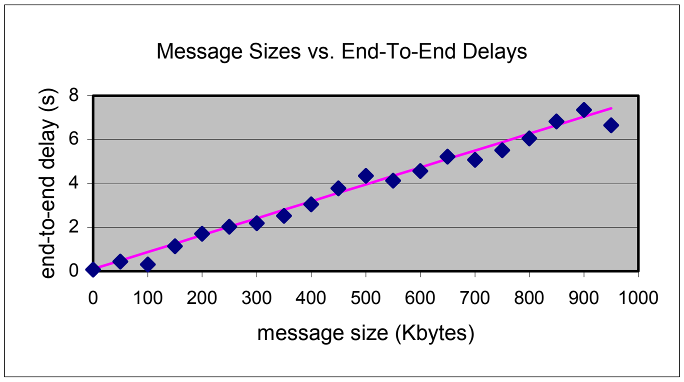


Fig. 2. End-to-end message transmission delay measurements between LSU and ORNL

We use the following result from [11] to obtain the linear regression estimate for the effective path bandwidth and the minimum end-to-end delay:

Theorem: Given a set of test messages with various sizes $R = \{r_i \mid i = 1,2,...,k\}$, and the corresponding message delays $D = \{d_i \mid i = 1,2,...,k\}$, the following formula gives the coefficient vector of a polynomial regression estimate in the least squares sense.

$$\vec{a} = (X^T X)^{-1}(X^T \vec{y}) \qquad (5)$$

where, $\vec{a}$ is the coefficient vector of a polynomial regression estimate: $d = a_{n-1}r^{n-1} + a_{n-2}r^{n-2} + ... + a_1 r + a_0$. Column vector $\vec{y} = D$, and matrix X is constructed as follows:

$$X = \begin{bmatrix} r_1^{n-1} & r_1^{n-2} & ... & r_1 & 1 \\ r_2^{n-1} & r_2^{n-2} & ... & r_2 & 1 \\ \multicolumn{5}{c}{\dots\dots\dots\dots\dots} \\ r_k^{n-1} & r_k^{n-2} & ... & r_k & 1 \end{bmatrix} \qquad (6)$$

We now analyze the effectiveness of this method under fairly general conditions. For a given message of size r, let the corresponding delay d be distributed according to a probability distribution $P_{d/r}$. Note that the expression $\frac{1}{EBW} r + d_{\min}$ for d is an approximation that estimates d for a given value r. While this equation captures the dominant relationship between r and d, it does not capture the random effects. Recall that such random components could be due to a number of factors such as delays at router or host buffers, packet retransmissions due to buffer overflows or other losses, and time variations in transferring the packets from application buffers to kernel buffers then to Network Interface Card (NIC) buffers. In general such variations are not major contributors to the delays, particularly for large messages sent over wide-area connections. Nevertheless, it is important to assess the effectiveness of the above method compared to the one that takes the randomness into effect. Given a linear estimator $a_1 r + a_0$ for the delay d, its expected square error is given by

$$I_{(a_1,a_0)} = \int (d - a_1 r - a_0)^2 P_{d,r}. \qquad (7)$$

The *best estimator* according to Eq (7) is given by $I_{(a_1^*,a_0^*)} = \min_{(a_1,a_0)} I_{(a_1,a_0)}$. For this cost measure, coefficients a_0 and a_1 cannot be computed even in principle since it requires the knowledge of $P_{d/r}$. Typically this is an infinite dimensional quantity and cannot be accurately estimated using a finite sample.

Consider that $a_1^1 r + a_0^1$ is an estimator for the delay d computed using Theorem 1. We will show that for any distribution $P_{d,r}$ we have

$$P\left\{ I_{(a_1^1,a_2^1)} - I_{(a_1^*,a_2^*)} < \varepsilon \right\} > 1 - 8\left(\frac{64e}{\varepsilon} \ln \frac{64e}{\varepsilon} \right)^2 e^{-\varepsilon^2 k/512}, \qquad (8)$$

which implies that the expected error of the computed estimate is within ε of the optimal with probability approaching 1 as the sample size k increases. In fact, for any given values of k and ε, the right hand side of the above expression can be used to compute probability with which the error of the estimate is within ε of the optimal. This is best type of result possible when the distribution $P_{d,r}$ is completely unknown. Similar results are shown in [12] using a more detailed expression for the delay, which results in estimators that are significantly more complicated than the above linear estimate. There are two important aspects of the above performance guarantee. On the positive side, it is entirely distribution-free in that it is valid independent of $P_{d,r}$, although stronger guarantees may be possible for certain distributions. On the negative side, it only ensures the closeness of estimator error to the best possible linear approximation, but it is quite possible that the latter itself is unsatisfactory. Note that the linear approximation model has been supported by the domain-specific considerations above.

We will now prove the above performance result. Consider the empirical error of the delay estimator $a_1 r + a_0$ given by

$$I^1_{(a_1,a_0)} = \frac{1}{k}\sum_{i=1}^{k}(d - a_1 r_i - a_0)^2 \qquad (9).$$

Now note that the above delay estimator $a_1^1 r + a_0^1$ minimizes this empirical error, that is $I^1_{(a_1^1,a_2^1)} = \min_{(a_1,a_0)} I^1_{(a_1,a_0)}$. The estimators class $\{a_1 r + a_0\}$ forms a vector space of dimensionality 2. Then by using the results from Vapnik [14], we have the following

$$\begin{aligned} & P\left\{ I_{(a_1^1,a_2^1)} - I_{(a_1^*,a_2^*)} < \varepsilon \right\} \\ & > P\left\{ \sup_{(a_1,a_0)} \left| I^1_{(a_1,a_0)} - I_{(a_1,a_0)} \right| < \varepsilon/2 \right\} \qquad (10) \\ & > 1 - 8\left(\frac{64e}{\varepsilon} \ln \frac{64e}{\varepsilon} \right) e^{-\varepsilon^2 k/512} \end{aligned}$$

where the last bound is due to the dimension 2 of the estimator class (details are standard can be found, for example in [7]).

IV. Multiple Quickest Path Computation

A set of nodes selected to deploy transport daemons and virtual links connecting them form an overlay network [15]. A typical overlay network with estimated available path bandwidths and minimum end-to-end delays is shown in Fig. 3 for illustrative purposes.

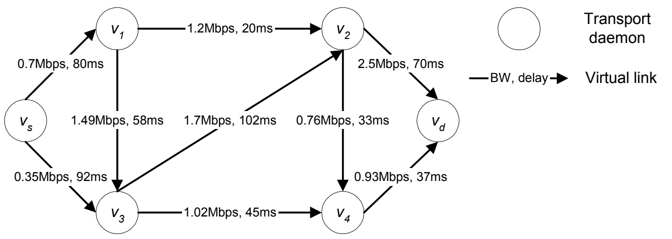


Fig. 3. Overlay network of transport daemons with estimated path bandwidths and minimum end-to-end delays

An overlay network can be represented by a graph $G(V, E)$ with vertices V denoting transport daemons and edges E denoting virtual links. Such an overlay network graph can be seen as a combination of a weighted graph and a flow network: each edge in an overlay network is associated with both the minimum delay, which corresponds to the edge weight in a weighted graph and the bandwidth, which corresponds to the capacity in a flow network. The quickest path problem is to find a routing path in an overlay network graph G such that the end-to-end delay time required to send a message of size r from a source node v_s to a destination node v_d is minimum. A routing path in an overlay network is usually made up of one or more virtual links or physical paths in the underlying network. Since the end-to-end delay for message transmission over a routing path does not only depend on the minimum delays of component virtual links, but also the associated available bandwidths, apparently the well-known single-source shortest path algorithm, Dijkstra's algorithm, cannot be directly applied to such graphs.

We design an approximate algorithm *MQP*(overlay network graph $G(V, E)$, message size r, path number m, source node v_s, destination node v_d) to compute multiple quickest paths from source node v_s to destination node v_d in an overlay network graph $G(V, E)$. This algorithm is modified from Dijkstra's algorithm [1,6,7] with the new key value of node $v_i, i = s, 1, 2, \ldots, d$ defined by $\frac{r}{EBW[v_i]} + d_{\min}[v_i]$, where $EBW[v_i]$ is the effective path bandwidth of the route from source node v_s to node v_i, and $d_{\min}[v_i]$ is the sum of the minimum virtual link delays along the route from source node v_s to node v_i. For simplicity, we use the minimum of the component virtual link bandwidths to approximate the effective path bandwidth $EBW[v_i]$.

The multiple quickest paths algorithm *MQP*(G, r, m, v_s, v_d) is briefly presented in Fig. 4. Note that every time a quickest path is found, its path bandwidth is subtracted from the bandwidths of its component virtual links before we search for the next quickest path.

```
MQP(G, r, m, $v_s$, $v_d$)
Begin
For each quickest path numbered from 1 to m
   Initialize resolved/unresolved node set, predecessor node list, and key value of source node $v_s$;
   Compute key value $\frac{r}{EBW[v_i]} + d_{\min}[v_i]$ for node $v_i \neq v_s$;
   While unresolved node set is not empty
      Select the node with the minimum key value;
      Relax its neighbor nodes;
      Remove this node from unresolved node set and add to resolved node set;
   Build the quickest path from $v_s$ to $v_d$ using the predecessor node list;
   Subtract path bandwidth from all component virtual link bandwidths of the current quickest path;
End
```

Fig. 4. Algorithm *MQP*(G, r, m, v_s, v_d) for computing multiple quickest paths

The relaxation procedure in the *MQP* algorithm is similar to the one used in Dijkstra's algorithm except that *MQP* uses the new key value and the path bandwidth is recomputed for each neighboring node to make sure that its path bandwidth is always the minimum of the component virtual link bandwidths.

As for the data transmission, the source node first retrieves multiple quickest paths from the routing table, divides the data into multiple parts appropriately, and then sends them to the second nodes through different paths concurrently. Suppose that m quickest paths have been found: $P_1, P_2, \ldots, P_m$. The partitioning of user data of size r into m parts: $r_1, r_2, \ldots, r_m$, requires that the transmission times along all routing paths are equal:

$$\begin{cases} \frac{r_1}{EBW(P_1)} + d(P_1) = \frac{r_2}{EBW(P_2)} + d(P_2) \\ \frac{r_2}{EBW(P_2)} + d(P_2) = \frac{r_3}{EBW(P_3)} + d(P_3) \\ \ldots\ldots \\ \frac{r_{m-1}}{EBW(P_{m-1})} + d(P_{m-1}) = \frac{r_m}{EBW(P_m)} + d(P_m) \end{cases} \qquad (11)$$

and

$$\sum_{i=1}^{m} r_i = r \qquad (12)$$

Particularly, when the number of quickest paths $m = 2$, we use the following equation to partition the data of size *r* [9]:

$$\begin{cases} r_1 = \frac{EBW(P_1)\cdot r}{EBW(P_1)+EBW(P_2)} + \frac{EBW(P_1)\cdot EBW(P_2)\cdot[d(P_2)-d(P_1)]}{EBW(P_1)+EBW(P_2)} \\ r_2 = r - r_1 \end{cases} \qquad (13)$$

A source node performs user data partition based on the measurement results and sends data components via multiple quickest paths. As an overlay router, an intermediate transport daemon performs routing functions. It receives incoming data components, extracts routing path information, and forwards them to the very next hop on the routing path. The destination node simply receives data components and stores them locally. If all data components have arrived, an acknowledgement is sent from the destination node to the source node to notify the sending daemon of the completion of the transmission task.

V. Measurement RATE Minimization

In high-speed networks, it is impractical to collect delay measurements for every message for the purpose of computing the effective bandwidth of every link. Even if all measurements are available, estimating *EBW* may consume computational resources on data that does not contribute useful information. In this section, we adapt the general method of [7] based on the statistical design of experiments [5] to the present *EBW* estimation problem. We allocate suitable rates for various links and re-compute the estimates using the measurements collected so far. Informally, such measurement rate optimization results in links with higher variances being measured more often than others.

We now briefly describe the general formulation of the statistical design of experiments [5], to which the network measurement problem can be mapped in more than one ways. Consider the model of [4] given by

$$Y_t = U_t(x_i) + \varepsilon_t(x_i) \qquad (14)$$

where Y_t is the observation taken at time *t* corresponding to the variable x_i and ε_t is the measurement error. In our case x_i corresponds to a path or link and Y_t is its *EBW* estimate. Let $U = \left(u(x_1), u(x_2), ..., u(x_s)\right)^T$ denote estimators at all nodes. Let $E_u\left[U\right] = U_0$ and

$$Var_u\left[\left(U-U_0\right)\left(U-U_0\right)^T\right] = K . \qquad (15)$$

We consider the design problem given by $\xi = \{(p_1, x_1), \cdots, (p_n, x_n)\}$ such that $p_i = r_i / N$ and $N = \sum_{i=1}^{n} r_i$. Let $U^1(\xi)$ be a predictor of U, and its quality is given by the matrix of expected squared residuals

$$D(\xi) = E_{u,\varepsilon}\left[\left(U^1(\xi)-U\right)\left(U^1(\xi)-U\right)^T\right]. \qquad (16)$$

The observational errors are assumed to have zero means and be uncorrelated such that:

$$E_{\varepsilon|u}\left[\varepsilon_t(x_i)\right] = 0 \qquad (17)$$

$$E_{\varepsilon|u}\left[\varepsilon_t(x_i)\varepsilon_{t^1}(x_i)\right] = \sigma^2\delta_{tt^1} \qquad (18)$$

Let φ be a criterion of optimality. One of the simplest problems is to select a design ξ^* that minimizes $\varphi\left[D(\xi)\right]$ where *D* is the residual matrix. Several methods for computing the optimal design ξ^* are presented in [5] for estimating the delays. In this formulation, we fixed the sites and found the optimal measuring rates at each of the chosen sites so that total rate of measurement is no more than *N* during the time window *[0,T]*. At fixed measurement rates, the solution here optimizes the allocation of measurement rates among the chosen sites, which yields more accurate estimation than equally distributing the measurement rates. There could be other potential ways to apply statistical design methods to improve the measurements in computer networks [4].

VI. Implementation in WANs and Experimental Results

The transport daemon for end-to-end delay minimization is implemented in C++ on Linux operating system. TCP connections are used for both end-to-end measurements and user data transmissions. We construct a simple overlay network solely for performance evaluation purposes by deploying transport daemons on the following three sites: LSU, ORNL, GaTech (Georgia Institute of Technology). The topology of this test overlay network is shown in Fig. 5. Since there is always a physical path connecting any two hosts in the Internet, the overlay network is essentially a complete graph.

We conduct two sets of transport experiments between ORNL and LSU in this overlay network: one using a single default TCP stream, and the other using two quickest paths. The first quickest path is the direct IP connection to destination while the second one is via the transport daemon deployed at GaTech as a virtual router. A user-defined header containing both data and path information such as data type, data size, path delay, path bandwidth, and a list of routing nodes, is propagated from the source node to the destination node to set up a data channel via virtual routers.

For each experiment, we transfer a certain size of data from ORNL to LSU. The data is partitioned into two parts for two quickest paths according to Equation (13) based on the data size as well as the bandwidth and delay measurements. The throughput performances of these experiments using different transport methods are tabulated in Table 1 for comparison.

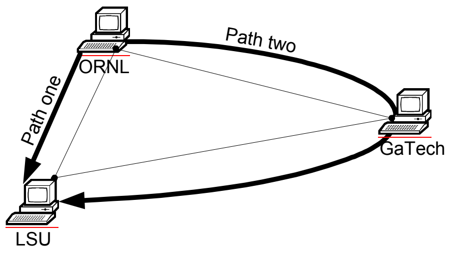


Fig. 5. Overlay network of transport daemons over Internet

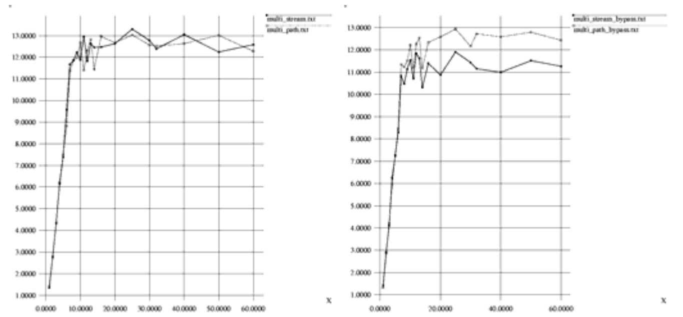
Table 1. Throughput performances of two transport methods from ORNL toLSU

Experiment index	File size (Mbytes)	Single TCP stream (Mbps)	Two quickest paths (Mbps)
1	1	1.63	3.31
2	5	1.40	2.36
3	10	1.38	2.68
4	15	1.35	2.55
5	20	1.16	2.41
6	25	1.07	2.10
7	30	0.84	1.80
8	35	0.92	1.75
9	40	1.44	3.79
10	45	1.88	4.26
11	50	1.01	2.03
12	55	0.98	1.68
13	60	0.66	1.13
14	65	0.82	2.17
15	70	1.11	2.06
16	75	1.03	2.48
17	80	0.79	1.37
18	85	1.26	3.39
19	90	1.21	2.83
20	95	0.92	1.72
21	100	1.16	2.74

Due to recent interest in parallel-TCP method that employs a number of TCP streams, we briefly compare our method with it. The data packets of parallel-TCP travel along the same IP route to the destination and therefore share all communication resources. However, the data packets of the multi-path transport are delivered via different routes with the support of overlay routers and are intended to avoid resource contest between different paths, especially on the bottleneck link. The multi-path transport exhibits similar throughput performance to parallel-TCP when there is no congestion or the congestion only occurs at two ends. The advantage of using the multi-path transport over parallel-TCP becomes evident when the second quickest path bypasses the congested zone experienced by the first quickest path. Fig. 6 shows such a case that the multi-path transport outperforms parallel-TCP when an increasing number of parallel streams saturate the default IP route. The x-axis in Fig. 6 represents the number of parallel streams that run along the default route (or the first quickest path in the multi-path transport), and the y-axis is the corresponding throughput performance measured in Mbps.

VII. Conclusion

We presented a measurement-based source routing method to minimize message delays via multiple quickest paths. Using a linear approximation, we presented a method which is computationally efficient and also analytically tractable under fairly general conditions This method is implemented using application-level daemons and tested at Internet nodes, and it outperformed the default TCP/IP method. There are number possible directions for future investigations. The measurement optimization method may be extended to identify a subset of nodes that at which measurements may be collected while still assuring the performance. Also practical advantages of the measurement optimization methods may be studied in detail.



(a) No congestion is bypassed (b) Congestion is bypassed

Fig. 6 Comparison of throughput performance between multi-stream and multi-path transports

References

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest. Introduction to Algorithms. the MIT Press, 2000.

[2] J. Curtis, T. McGregor. Review of Bandwidth Estimation Techniques. New Zealand Computer Science Research Students' Conference, 19-20th April 2001, University of Canterbury, New Zealand.

[3] C. Dovrolis, P. Ramanathan, D. Moore. Packet Dispersion Techniques and Capacity Estimation. Submitted to the IEEE/ACM Transactions in Networking, 2002.

[4] V. V. Fedorov, D. Flanagan, T. Rowan, and S. G. Batsell, Analysis and monitoring design for networks, Technical Report ORNL-TM-13620, Oak Ridge National Laboratory, 1998.

[5] V. V. Fedorov and P. Hackl, Model-Oreinted Design of Experiments, Springer-Verlag, Berlin 1997.

[6] M. Jain, C. Dovrolis. End-to-End Available Bandwidth: Measurement methodology, Dynamics, and Relation with TCP Throughput. Proceedings of ACM SIGCOMM, August 2002.

[7] N. S. V. Rao, Vector space methods for sensor fusion problems, Optical Engineering, vol 37, no. 2, 1988, pp. 499-504.

[8] N. S. V. Rao, On design of measurements for end-to-end delay minimization in wide-area networks,9th International Conference on Advanced Computing and Communications, 2001.

[9] N. S. V. Rao, NetLets for end-to-end delay minimization in distributed computing over Internet using two-Paths, International Journal of High Performance Computing Applications, 2002, vol. 16, no. 3, 2002.

[10] N. S. V. Rao, Overlay Networks of In-Situ Instruments for Probabilistic Guarantees on Message Delays in Wide-Area Networks, IEEE Journal on Selected Areas of Communications, vol 22, no.. 1, 2004.

[11] N. S. V. Rao, Y. C. Bang, S. Radhakrishnan, Q.Wu, S. S. Iyengar, and H. Cho, NetLets: Measurement-based routing daemons for low end-to-end delays over networks, Computer Communications, vol. 26, no. 8, 2003, pp. 834-844.

[12] N.S.V. Rao, S.G. Batsell. Algorithm for Minimum End-to-End Delay Paths. IEEE Communications Letters, 1999.

[13] N. S. V. Rao, W. C. Grimmell, Y.C. Bang, S. Radhakrishnan, On algorithms for quickest paths under different routing modes, to appear in IEICE Transations on Communications, 2004.

[14] V. Vapnik, Nature of Statistical Learning, Springer-Verlag, 1996.

[15] Q. Wu, , "Control of Transport Dynamics in Overlay Networks", Ph.D. Dissertation, Dept of Computer Science, Louisiana State University, May 2003.