IEEE International Performance Computing and Communications Conference, April 7-9, 2005
Phoenix, AZ.

On Transport Daemons for Small Collaborative Applications over Wide-Area Networks

Qishi Wu, Nageswara S. V. Rao
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831
{wuqn, raons}@ornl.gov

S. Sitharama Iyengar
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803
iyengar@bit.csc.lsu.edu

Abstract

A number of science applications employing collaborative computations require transport methods that guarantee end-to-end performance at the application level. Throughputs achieved by the traditional transport methods are limited to single default best-effort IP paths, which are often insufficient for the application tasks. In this paper, we present a measurement-based approach that utilizes application-level daemons at the collaborating sites to enhance the transport performance by utilizing multiple quickest paths. This method is based on a linear approximation of the effective bandwidth, and is computationally efficient and analytically tractable under fairly general conditions. We implemented and tested this method at Internet nodes, and the experimental results show significant performance improvements over the default TCP.

1. Introduction

Several recent large-scale science applications are carried out by collaborative teams of geographically distributed researchers, for example, the Terascale Supernova Initiative [16]. These computations are typically distributed over a small number (typically 4-5) of sites over the Internet, and they require effective transport methods for exchanging datasets and messages of varying sizes on demand. Particularly for large datasets, these applications require sustained high throughputs over long-haul connections. Often, the throughputs provided by single default paths are not sufficient even when special transport methods such as parallel TCP or rate controlled-UDP methods are used. One of the main limitations is that these methods may achieve high bandwidth utilization on the default paths but are not able to exploit the unused bandwidth on other disparate paths; note that all streams of parallel TCP typically share the same default IP path. While separate dedicated networks with sufficient bandwidth will meet these requirements, the small collaborative scope of these applications makes Internet-based methods more practical due to the high cost of the former. Thus, our objective is to leverage the access to hosts at the collaborating sites to augment the throughputs achieved on default IP paths by using additional paths amongst the sites. Furthermore, solutions that do not require changes to the infrastructure, such as router/switch reconfigurations or kernel rebuilds, are always preferable since these deployments are typically handled by application scientists and not systems/network administrators. In particular, networking solutions that can be deployed at the application level are highly desirable. While there are performance limitations that cannot be overcome without infrastructure changes, we will illustrate that several significant and practical performance improvements can be achieved by leveraging the multiple hosts at the collaborating sites instead. Our goal in this sense is much more modest than developing better transport methods for the Internet at large.

One of the important challenges in the above class of applications is to minimize the message delays over wide-area networks. Over the Internet, messages are decomposed into data packets and 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. But such "static" paths are not necessarily the quickest ones because the dynamic nature of the "available" path bandwidth may necessitate choosing different paths for different messages at different times. In general, the delays experienced over the wide-area connections by data packets have been shown to have highly complicated statistical distributions, which make the end-to-end performance highly unpredictable. Furthermore, for high bandwidth levels, the delays measured at the application level may include host-related components due to factors such as kernel-application and kernel-NIC (Network Interface Card) packet transfers, priorities among competing application processes, and CPU load levels. As a result, ensuring performance at the transport or lower levels of IP stack does not always

suffice to ensure the application-level performance, particularly at very high speeds. Since an accurate analytical prediction of delay distributions at the application level is not feasible over wide-area networks, measurement-based approaches have been developed albeit with an easier goal of identifying close-to-optimal quickest paths [9-11]. Additionally, by utilizing application-level daemons to realize alternative or multiple paths, these works illustrated that the throughputs of the default TCP connections could be improved. By strategically routing packets amongst the application-level daemons, these methods were able to utilize augmentations and/or alternatives to default IP paths. However, these works can only be considered as a proof-of-principle of such approaches; in particular, the analytical methods in [10] are not readily deployable in practice, and the results in [11] are based on simulation only and are not analytically justified.

In this paper, we describe the functional framework of a measurement-based transport method that utilizes application-level daemons. In this method, active measurements are used to estimate the effective bandwidth and minimum delay for each link using the linear regression technique that provides probabilistic performance guarantees under fairly general conditions. Based on the link estimates, multiple quickest paths are computed to achieve low end-to-end message delays. These paths are then realized using a source-based routing strategy supported by the application-level routing daemons. We also describe a systematic approach to optimizing the amount of measurements collected based on the statistical design [4,5]. The scope of our work is limited to the above mentioned collaborative environments with a *small* number of sites typically operated by application scientists, and is not intended for large-scale Internet deployment.

The rest of the paper is organized as follows. We describe in Section 2 a general framework of network daemons that estimate the link properties and support message transfers. Section 3 discusses the estimation of effective bandwidth and minimum delay. Section 4 presents the algorithms for computing multiple quickest paths. Section 5 is devoted to the minimization of measurements. The implementation details and experimental results are provided in Section 6.

2. Network daemon of measurement-based transport method

A framework of network transport daemons for providing end-to-end delay performance using measurements is illustrated in Fig. 1 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 estimation of link properties, and routing table construction. The transport module establishes a *virtual link* by propagating the source routing control information along a pre-computed routing path, and also provides data transfer service to user applications through API functions. Here, the virtual link refers to the communication channel established between two daemons, which in turn may consist of multiple routers/switches connected together by physical links.

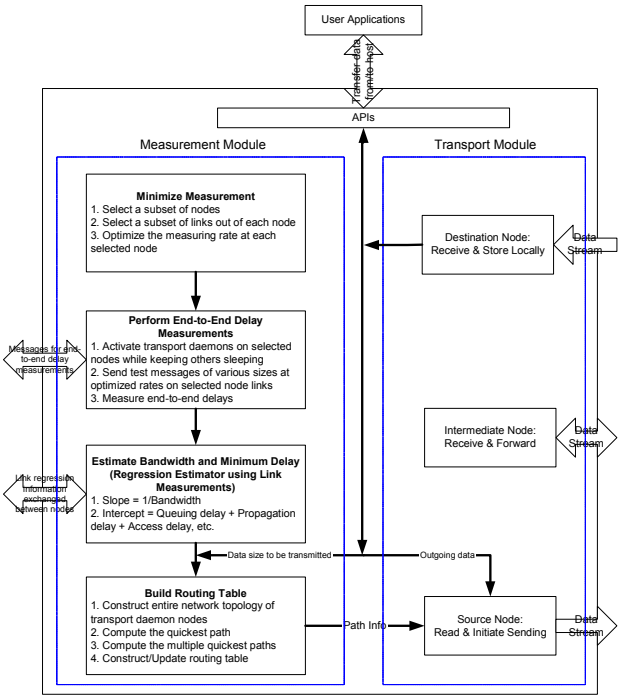

Fig. 1. Framework of measurement-based transport daemon.

During message transmission, the daemons operate in one of three modes: sending, routing, and receiving. A sending node reads data from its local applications and prepares the source-routing header information. A receiving node forwards incoming data to its local applications and handles acknowledgements if necessary. An intermediate routing node acts as a virtual router at the application level, forwarding packets to the next-hop daemon node, which is extracted from the path information stored in the source-routing header.

Compared to the earlier works in [9-11], our work refines various components, and additionally incorporates a measurement optimization component. To save the computational and networking resources, not every deployed transport daemon needs to collect measurements on all links connected to it. Instead, we strategically select a subset of them to collect link measurements at optimal rates that are determined statistically. The measurement results are broadcast to the entire network so that each

transport daemon can build the network topology needed for source routing via multiple quickest paths.

3. Bandwidth and delay measurement

The link bandwidth is generally regarded as the fastest rate at which data can be sent along the link, while the available bandwidth is the spare bandwidth "left over" after subtracting the cross traffic level from the link bandwidth [2,3]. Due to the complex traffic distributions over wide-area networks and the non-linear nature of transport protocol dynamics (in particular TCP), the throughput achieved in actual message transfers is often different from both the link and available bandwidths, and typically contains a random component. We consider the *effective path bandwidth* to be the maximum throughput achieved on a path by a flow given the cross traffic load and the transport protocol. Thus, the notion of effective bandwidth is specific to the transport protocol employed by the transport daemon, and is related to both link and available bandwidths, perhaps in a complicated way based on the mechanism used by the protocol. We employ an active measurement technique to estimate the effective path bandwidth and minimum delay for each virtual link that connects two daemons.

There are three main types of delays involved in the message transmission over virtual links, namely, link propagation delay d_p imposed at the physical link level, equipment-associated delay d_q incurred by the processing and buffering at hosts and routers/switches, and bandwidth-constrained delay d_{BW}, which is determined by the currently available path bandwidth and message size. Due to the time-varying 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, d_{BW} may also exhibit randomness particularly over congested wide-area links. 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:

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

Note that $d(P,r)$ is a random variable for fixed P and r. If the message size r is smaller than the smallest Maximum Transmission Unit (MTU) on the path, and the network is lightly loaded, the bandwidth-constrained delay in Eq. (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 $d_{\min}(P) = \sum_{i=1}^{l}(d_{p,i}(P) + d_{q,i}(P,r)), \quad r < MTU$. 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 path P. Then, the message delay $d(P,r)$ for transmitting a message of size r can be approximated by the following linear model:

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

In a circuit-switched 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 P:

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

In a packet-switching network wherein data packets are stored and forwarded at intermediate nodes, the bandwidth-constrained delay accumulates at every link. For $r<MTU$, the EBW is approximated in [11] as:

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

However, for large message sizes, the pipelining of data packets along component links actually makes the effective path bandwidth practically close to Eq. (3).

We use an active measurement technique to estimate the effective bandwidth of a virtual link. The measurement module generates a set of test messages of various sizes, sends them over an outgoing link, and measures the message delays. Then, we apply a linear regression to fit the measured message size r and end-to-end delay d pairs. A first order approximation of the effective path bandwidth EBW and the minimum message delay $d_{\min}$ are then given by the slope and intercept of the regression line $d = r/EBW + d_{\min}$, respectively. The intercept of the regression line is sensitive to the delay measurements, and sometimes yields a negative value, in which case we simply replace the estimate of $d_{\min}$ with a UDP-based minimum message delay measurement.

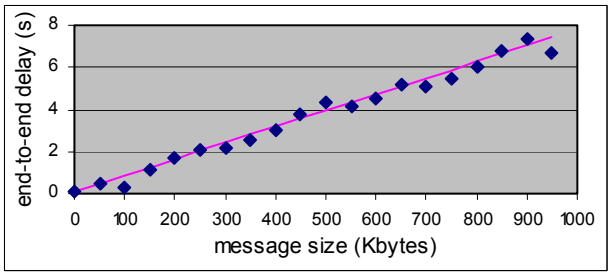

Fig. 2. End-to-end delay between LSU and ORNL.

[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.

The message delay measurements between two transport daemons deployed at Louisiana State University (LSU) and Oak Ridge National Laboratory (ORNL), and the corresponding linear regression estimate are shown in Fig. 2. The measurements are collected using TCP-based transport method. The message transmission is carried out three times for each message size and the average delay is calculated. From this figure, the effective path bandwidth *EBW* of this virtual link is estimated to be about 1.0 Mbps and the minimum end-to-end delay to be about 35 ms at the time of experiment. These estimates are generally consistent with the results obtained from other network utilities such as iperf.

Given a set of test messages with 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 from [11] gives the coefficient vector of a polynomial regression estimate in the least squares sense:

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

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:

$$X = \begin{bmatrix} r_1^{n-1} & r_1^{n-2} & ... & r_1 & 1 \\ r_2^{n-1} & r_2^{n-2} & ... & r_2 & 1 \\ ..... & & & & \\ r_k^{n-1} & r_k^{n-2} & ... & r_k & 1 \end{bmatrix}. \quad (6)$$

Here we employ the special case n=2. Since the message delays have a random component, the performance of this method is unclear and has not been discussed in [11]. We now analyze the effectiveness of this method here under fairly general conditions. For a given message of size *r*, let the corresponding delay *d* be distributed according to the joint probability distribution $P_{d,r}$. Recall that the underlying 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 NIC buffers. In general such variations are not a major contributor to the delays, particularly for large messages sent at low rates but can become significant at high transport rates such as 1Gbps. Nevertheless, it is important to assess the effectiveness of the above method compared to the best case when one has a complete knowledge of the delay distributions. Due to the myriad and complexity of the factors that contribute to the randomness, it is impractical to estimate the distribution $P_{d,r}$; so here we assume that it is completely unknown. Given a linear estimator $a_1 r + a_0$ for the delay *d*, its expected square error is given by:

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

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, however, the coefficients of the best estimator cannot be computed even in principle since it requires a complete knowledge of $P_{d,r}$; note that even if the distribution is completely known, the minimization problem may not be computationally tractable if the distribution is complex. Typically $P_{d,r}$ is an infinite dimensional quantity and cannot be accurately estimated using *k* delay measurements, and in fact it is easier to estimate the coefficient pair instead in the following sense. Consider that $a_1^1 r + a_0^1$ is an estimator for the delay *d* computed using the above method. We will show that for any distribution $P_{d,r}$, we have:

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

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. Informally speaking, Eq (8) guarantees that the error of the estimator is within ε of the optimal with the probability that approaches 1 as sample size becomes large, and it is valid independent of the distribution. This is best type of result possible when the distribution $P_{d,r}$ is completely unknown. Similar results are shown in [10] 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 specific distributions if known. On the negative side, it only ensures the closeness of the estimator error to the best possible linear approximation, but it is quite possible that the latter itself is unsatisfactory if the underlying relationship is non-linear. However, the linear approximation model has been supported by the domain-specific considerations as shown in the measurements of Fig. 2. We will now prove the above performance result. Consider the empirical error of the delay estimator $a_1 r + a_0$ given by:

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

Now note that the above delay estimator $a_1^1 r + a_0^1$ minimizes this empirical error, which is given by $I^1_{(a_1^1, a_2^1)} = \min_{(a_1, a_0)} I^1_{(a_1, a_0)}$. The estimator 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:

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

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

4. Multiple quickest path computation

The 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.

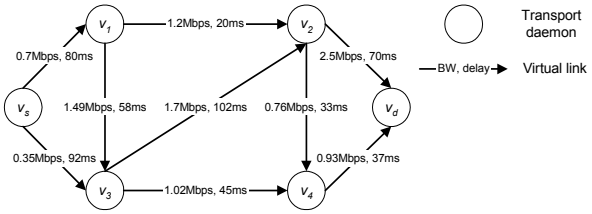

Fig. 3. Overlay network of transport daemons with bandwidths and end-to-end delays.

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

For a path P_i from v_s to v_i, the message delay is given by $r / EBW[v_i] + d_{\min}[v_i]$, where $EBW[v_i]$ is the effective path bandwidth of P_i and $d_{\min}[v_i]$ is the sum of the minimum virtual link delays along P_i. We use the minimum of the bandwidths of virtual links along path P_i to approximate the effective path bandwidth $EBW[v_i]$. Under these conditions, this problem is the classical quickest path problem, which is solved by Chen and Chin [17] using multiple invocations of Dijkstra's shortest path algorithm on suitably constructed sub-graphs of G. We start with the original graph G and compute multiple quickest paths by repeatedly finding the quickest path on the residual graph G_1 using the $QP(G_1, r)$ algorithm of [17]. Every time a quickest path is found, its effective path bandwidth is subtracted from the bandwidths of its component virtual links in the residual graph before we compute the next quickest path. This approach is similar to [9,10] but differs in the utilization of the available bandwidth estimates described in Section 3. The multiple quickest path algorithm $MQP(G, r, v_s, v_d)$ is illustrated in Fig. 4.

Algorithm $MQP(G, r, v_s, v_d)$
begin
  Initialize the residual graph G_1 with G;
  Initialize the quickest path index $i = 1$;
  while v_s and v_d are connected in G_1 **do**
    Compute the quickest path P_i from v_s to v_d in G_1 using Chen and Chin $QP(G_1, r, v_s, v_d)$ algorithm;
    For each edge e in path P_i, reduce the link bandwidth of edge e in G_1 by $EBW(P_i)$;
    $i = i + 1$;
end

Algorithm $QP(G_1, r, v_s, v_d)$
begin
  Find a set of distinct bandwidths $\{BW(e) \mid e \in G_1\} = \{b_1, b_2, \ldots, b_q\}$, q: no of distinct BWs in G_1;
  for each bandwidth $b_j, j = 1, 2, \ldots, q$ in the bandwidth set **do**
    Construct a sub-graph of G_1, each link with bandwidths larger than or equal to b_j;
    Apply Dijkstra's algorithm with key value $r / b_j + d_{\min,j}$ to compute shortest path from v_s to v_d;
  Compare the above q shortest paths and return the one with the minimum $r / b_j + d_{\min,j}$;
end

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

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 corresponding next nodes concurrently. Suppose that m quickest paths have been found, denoted

by $P_1, P_2, \ldots, P_m$. The partitioning of user data of size r into m parts, given by $r_1, r_2, \ldots, r_m$, requires that the transmission times along all routing paths are equal:

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

and

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

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

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

and the equations for the partition in the general case are described in [10].

A source daemon partitions the dataset based on the measurements results and sends the components via multiple quickest paths. As an overlay router, an intermediate transport daemon receives incoming data components, extracts routing path information, and forwards them to the 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 daemon to the source to notify the completion of transmission.

5. 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 necessarily contribute useful information. In this section, we adapt the general method of [7] based on the statistical design of experiments [5] to the *EBW* estimation problem. We allocate suitable rates for various links and re-compute the estimates using the measurements collected so far.

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:

$$Y_t = U_t(x_i) + \varepsilon_t(x_i), \quad (14)$$

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 virtual 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 $E_u[U] = U_0$ and

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

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:

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

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

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

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

Let φ be a criterion of optimality. One of the simplest problems is to select a design ξ^* that minimizes $\varphi[D(\xi)]$, 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 the total measurement rate 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.

6. Implementation and experimental results

The transport daemons described in Section 2 are implemented in C++ on Linux operating system. TCP is used for both message delay measurements and user data transmissions. We construct a simple overlay network solely for performance evaluation purposes by deploying transport daemons at 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 an IP path connecting any two hosts on the Internet, the overlay network is essentially a complete graph.

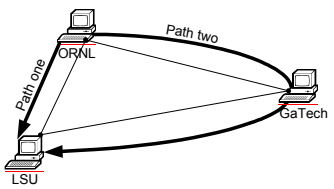

Fig. 5. Overlay network of transport daemons over Internet.

We conducted two sets of transport experiments between ORNL and LSU: one using a single default TCP stream, and the other using two quickest paths computed by the transport daemons. The first quickest path is the direct IP connection to destination while the second one is via the intermediate 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 sequential routing nodes, is propagated from the source node to the destination node to set up a data channel via the transport daemons.

In each experiment, we transferred datasets of different sizes from ORNL to LSU through the transport daemons we deployed at these sites. The data is partitioned into two parts for two quickest paths according to Eq. (13) based on the data size as well as the effective bandwidth and minimum end-to-end delay measurements. The throughput performances of these experiments using different transport methods are tabulated in Table 1 for comparison.

Table 1. Throughput performances of two transport methods from ORNL to LSU.

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

In consideration of recent interest in parallel-TCP method that employs a number of concurrent TCP streams, we briefly compared 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 contention 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 method 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 measured in Mbps.

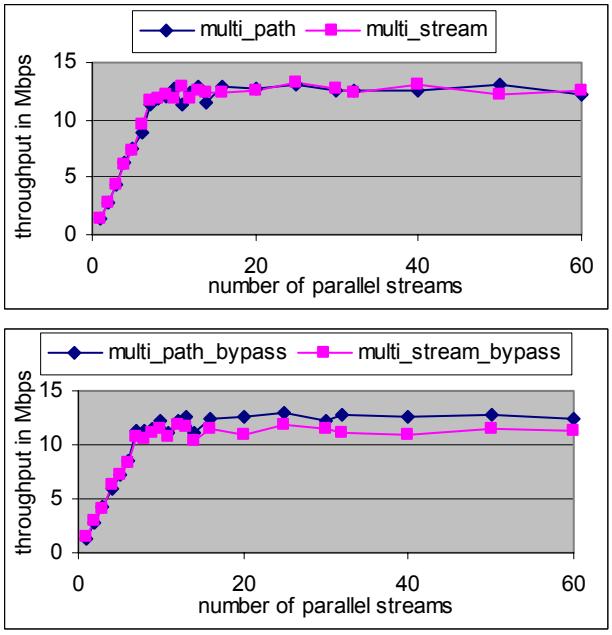

Fig. 6 Comparison of throughput performance between multi-stream and multi-path transports: (a) upper plot: no congestion is bypassed; (b) lower plot: congestion is bypassed.

7. Conclusion

In this paper, we presented a measurement-based approach to minimize the message delays between the nodes of small collaborative applications distributed over the Internet using multiple quickest paths. These disparate overlay paths are realized by specially designed transport daemons at the collaborating sites that enable the source routing. We developed an active measurement method using a linear approximation technique to estimate the effective bandwidth and minimum end-to-end delay of a virtual link, and showed it to be computationally efficient and analytically tractable under fairly general conditions. We presented an algorithm to compute multiple quickest paths between the collaborating nodes using the link estimates. We also used a measurement minimization technique to reduce the computing and communication overhead incurred by the active measurement method. All these functional components are integrated into a single transport daemon implemented at the application level for easy deployment. The test results collected from the experiments conducted in Internet environments demonstrated its superior transport performance over the default TCP/IP method and also parallel TCP. This approach is primarily intended for collaborative applications that are distributed over a small number of

sites connected over the Internet, and requires no infrastructure changes. It is, however, not intended for wide Internet deployment or large collaborative teams.

There are a number of possible directions for future investigations. First, the current TCP-based method used in the measurement and transport modules could be replaced with other UDP-based transport protocols to further improve bandwidth utilization or to match the underlying connections such as dedicated channels. Second, the measurement optimization method may be extended to identify a subset of nodes to collect measurements while still assuring the desired performance. Also, the practical advantages of the measurement optimization method may be explored in further detail. Finally, our goal is to deploy these transport daemons in real applications and provide improved transport performance to collaborative computations over the Internet.

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*, University of Canterbury, New Zealand, 19-20th April 2001.

[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, pp. 499-504, 1988.

[8] N. S.V. Rao, On design of measurements for end-to-end delay minimization in wide-area networks, *the 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, pp. 834-844, 2003.

[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.

[16] Terascale Supernova Initiative, http://tsi.phys.ornl.gov

[17] Y. L. Chen and Y. H. Chin, The quickest path problem, *Computers and Operations Research,* vol. 17, no. 2, pp. 153-161, 1990.