# On Transport Methods for Peak Utilization of Dedicated Connections

Qishi Wu
Department of Computer Science
University of Memphis
Memphis, TN 38152, USA
Email: qishiwu@memphis.edu

Nageswara S. V. Rao
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA
Email: raons@ornl.edu

Xukang Lu
Department of Computer Science
University of Memphis
Memphis, TN 38152, USA
Email: xlv@memphis.edu

***Abstract*—Several research and production networks now provide multiple Gbps dedicated connections to meet the demands of large data transfers over wide-area networks. Application throughputs, however, were not able to match these rates because the traditional transport methods have not been optimized for such connections. We propose a transport method based on stochastic approximation methods that: (a) stabilizes the source rate for peak utilization of connection bandwidth, and (b) adapts the acknowledgment interval to maximize the goodput at the receiver. We show the asymptotic stability and convergence of this method in maximizing the throughput over dedicated connections under fairly general conditions. Extensive experimental results indicate the effectiveness of this transport method in achieving file transfer throughputs that closely match iperf bandwidth measurements on dedicated connections of several thousand miles over UltraScience Net and ESnet, and also illustrate its superior performance on a local dedicated connection in comparison with existing methods.**

**Keywords:** Transport methods, dedicated channels, stochastic approximation.

## I. Introduction

A number of large-scale computational applications in various fields of science require high-speed data transfers to support remote collaborations and distributed computing over wide-area networks. Efforts to support such applications on shared IP networks have met with a limited success due to the variability of available bandwidth in response to "other" network traffic. Dedicated bandwidth connections are considered as an alternative solution and several research projects are currently underway to provide such connections including User Controlled Light Paths (UCLP) [1], UltraScience Net (USN) [2], Dynamic Resource Allocation via GMPLS Optical Networks (DRAGON) [3], On-demand Secure Circuits and Advance Reservation System (OSCARS) [4] of ESnet, Hybrid Optical and Packet Infrastructure (HOPI) [5], and many others. Such dedicated connections are expected to proliferate widely as reflected by production networks, such as Internet2 and ESnet, now offering on-demand circuits, Multiple Protocol Label Switching (MPLS) tunnels, and dedicated Virtual Local Area Networks (VLAN) connections.

From a transport protocol perspective, the dedicated channels obviate the need for explicit congestion and fairness control. However, as indicated by measurements over dedicated channels [6], the packet loss at high sending rates is often non-zero and the delay variations contain non-trivial random components. Consequently, simply *a priori* fixing the source sending rate right at the connection capacity is unlikely to maximize the throughput at the destination since packet losses occur at rates that depend on technologies used to provision the connection. In fact, such rates have pushed the bottleneck from the network to the end system, whose dynamics the traditional transport methods are not optimized for handling. Since the data receiving process typically runs concurrently with other resource-demanding workloads in a shared computing environment, it may not always obtain sufficient system resources such as CPU, memory, buffer/cache, and file system to process and save packets arriving from high-speed dedicated links, resulting in significant packet drops at the end system. Therefore, the source rate must be dynamically adjusted to yield the highest goodput at the destination by taking into account both connection and host effects. Here, the goodput is defined as the throughput of user payload excluding duplicated datagrams and is equivalent in value to the throughput if the user-defined header size is negligible.

Currently there are two approaches to transport protocol design: TCP enhancements and UDP-based transport with non-Additive Increase Multiplicative Decrease (AIMD) control. In recent years, many changes to TCP have been introduced to improve its performance for high-speed networks [7], including Scalable TCP [8], High-Speed TCP Low Priority (HSTCP-LP) [9], Fast Active-Queue-Management Scalable TCP (FAST) [10], and Stream Control Transmission Protocol (SCTP) [11].

Several UDP-based high-performance transport protocols have been developed to overcome TCP's throughput limitations for high bandwidth connections, although not necessarily optimized for dedicated connections. Such research efforts include SABUL/UDT [12], Tsunami [13], RBUDP [14], Rate-Adaptive Protocol for Intelligent Data-transfer (RAPID) [15] and others (see [16] for overviews). We tested several of these protocols over 1 Gbps ORNL-Atlanta-ORNL (Oak Ridge National Laboratory) dedicated channel, and they required finer manual tuning of parameters to achieve throughputs on the order of 900 Mbps [6]. Furthermore, this tuning required an intricate knowledge of implementation details, and is typically very labor-intensive and somewhat unstructured.

*Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7232*
*http://dx.doi.org/10.4108/ICST.BROADNETS2009.7232*

In the Hurricane protocol [17], a suitable sending rate is first derived from the throughput profile of the connection, and then its control parameters are further manually tuned for rate optimization. Both steps involved significant time and efforts, wherein several hours of active measurements were needed for profile generation, followed by trial-and-error parameter tuning. With such efforts, Hurricane performed better than or comparable to other TCP- or UDP-based transport methods over ORNL-Atlanta-ORNL and ORNL-NCState (North Carolina State University) connections [6]. However, the underlying optimizations are ad hoc and must be repeated for each different connection, thereby incurring the efforts all over again. This paper is motivated by the need to automate and shorten such parameter selection and tuning process, which involves adapting various transport parameters such as data and acknowledgment rates. A main challenge is to compute and adapt various data rates and transport parameters automatically, whose estimates are subject to the variations due to connection and host effects as well as the finite window effects. In particular, these estimates contain seemingly stochastic components that are connection- and host-specific and must be explicitly accounted for to achieve high link utilization.

We propose *Peak Link Utilization Transport* (PLUT) that automates the rate stabilization for throughput maximization using stochastic approximation methods, as opposed to the manual parameter tuning in the Hurricane transport. PLUT incorporates two critical components: (a) the source rate at the sender is statistically stabilized to achieve the peak link utilization at a low retransmission rate, and (b) the acknowledgment (ACK) interval at the receiver is adapted to maximize the destination goodput. We utilize the existing stochastic approximation (SA) methods to show the asymptotic stability and convergence of this method in maximizing the throughput over dedicated connections under fairly mild conditions on the error process and underlying throughput, loss and retransmission profiles. These conditions are justified by the measured profiles of dedicated connections under different conditions and parameter values.

PLUT is implemented and tested on 1 Gbps local and wide-area dedicated connections over USN, ESnet and their hybrid concatenations of several thousand miles. USN is a wide-area experimental network that provides dedicated high-bandwidth channels for large data transfers. The data plane of USN consists of four thousand miles of dual OC192 connections spanning ORNL, Atlanta, Chicago (CHI), Seattle (SEA) and Sunnyvale (SUN). These connections use Ethernet over SONET (EoS) technology: they are switched in the core at SONET level using Ciena CD-CI switches and are provisioned at the edges using Force10 E300 switches at the Ethernet level. On ESnet, 1 Gbps VLAN-tagged MPLS tunnel is set up between Chicago and Sunnyvale via Cisco and Juniper routers, which is about 3600 miles long. USN peers with ESnet in Chicago, where 1 GigE USN and ESnet connections are cross-connected using Force10 Ethernet switch. The experimental results show that PLUT achieves file transfer rates

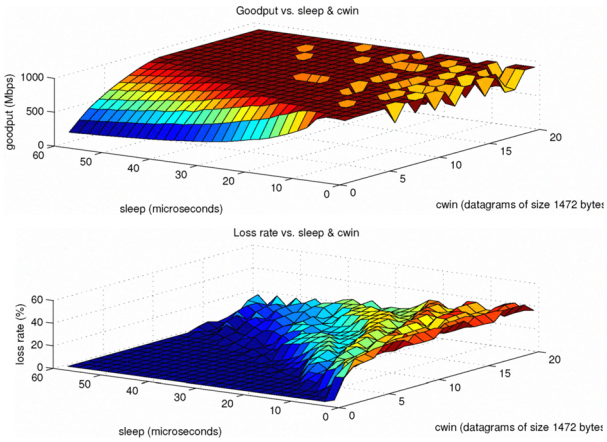


Fig. 1. Goodput and loss profiles of 9900 mile 1 Gbps USN-ESnet hybrid connection.

within a few Mbps of iperf peak bandwidth measurements on USN and ESnet, and consistently outperforms TCP and UDT for different file sizes on a local back-to-back dedicated connection.

The rest of the paper is organized as follows. In Section II, we present a UDP-based transport control structure for PLUT. In Section III, we describe the transport profiles based on measurements collected on dedicated channels. The strategies for source rate control and destination acknowledgment interval control are described in Section IV. The PLUT implementation details and experimental results are given in Section V.

## II. PLUT Control Structure

For a given dedicated connection, we generate the *goodput* and *loss profiles* using a UDP window-based scheme: we send a *window* $W(t)$ of datagrams in a batch and wait for *idle* or *sleep time* $T(t)$, and repeat the process. Thus the sending rate $r_S(t)$ at the source is regulated by the pair $(W(t),T(t))$, and at the destination we compute the *goodput* rate $g_D(t)$, i.e. the rate at which datagrams are received. Then the *loss rate* is given by $l(t) = r_S(t) - g_D(t)$, where the rates are computed within a time interval $T_R$. We show the goodput and loss profiles of 1 Gbps 9900 mile hybrid connection consisting of 6300 mile SONET connection on USN and 3600 mile MPLS tunnel of ESnet in Fig. 1, where each point in the horizontal plane corresponds to $(W(t),T(t))$. When the sending rate is low, the loss rate is zero but increases sharply as the sending rate approaches the connection bandwidth, and the goodput increases with the sending rate and flattens around a peak value once losses appear. It is critical to note that the peak throughput is achieved at non-zero, albeit very small, loss rates; such phenomenon is quite common over dedicated connections of different modalities [6]. Thus, lost datagrams need to be recovered, but this process must be controlled carefully so that the retransmitted datagrams do not constitute a significant portion of the sending rate.

PLUT employs a UDP-based transport control structure for disk-to-disk data transfer as shown in Fig. 2. The sender

*Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7232*
*http://dx.doi.org/10.4108/ICST.BROADNETS2009.7232*

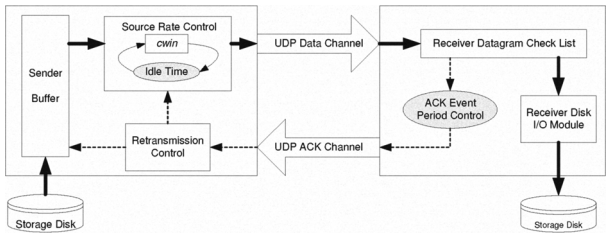


Fig. 2. Transport control structure for file transfer.

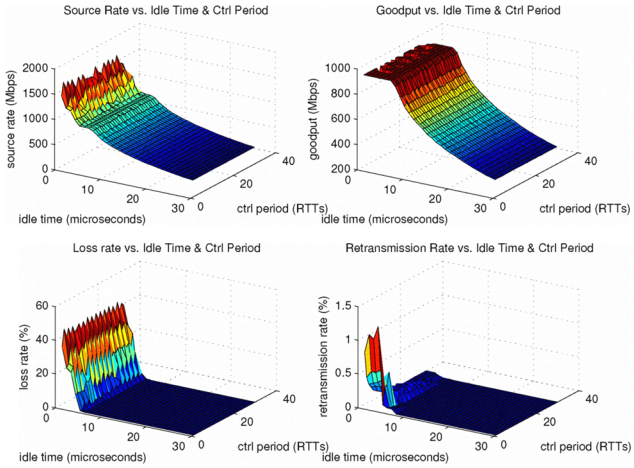


Fig. 3. Goodput, loss and re-transmission profiles of PLUT over 9900 mile 1 Gbps USN-ESnet hybrid connection with default MDS of 1472 bytes.

(source) reads data sequentially from its local storage device as a set of UDP datagrams of *Maximum Datagram Size* (MDS), each of which is assigned a unique continuous sequence number and loaded into the sender buffer. Note that the default MDS is of 1472 bytes (MTU of 1500 bytes − IP header of 20 bytes − UDP header of 8 bytes) in the Internet. The receiver (destination) accepts the incoming datagrams in the order of their arrival and keeps track of the datagram sequence numbers in a check list. The received datagrams are immediately forwarded to a disk I/O module that handles datagram reordering if necessary and writes them to the disk in order in the background. Based on the status of the datagram checklist, a list of positive or negative acknowledgments (ACK) of lost datagrams for the interval $I(t)$ are generated and sent periodically to the sender for retransmission.

As shown in Fig. 2, the data flow moves from source to destination along the solid lines and the acknowledgment feedback follows the dotted lines from destination to source. In this transport structure, there are two control operations represented by two shaded elliptic boxes: (a) source rate control through idle time and (b) ACK event interval control. The transport performance over high-speed dedicated channels critically depends on the strategies used in these control operations. In several transport control protocols, a positive acknowledgment is sent for received data packets, which is necessary for shared lossy links in Internet environments. However, dedicated channels usually provide much more reliable connections, where packet loss is much smaller than connection capacities. At high data rates, generating and sending acknowledgments at the receiver consumes CPU time and may interfere with the host receiving process. Similarly, accepting and processing acknowledgments at the sender may also affect the host sending process. To achieve peak performance over dedicated channels, we employ a mixed acknowledgment mechanism that sends an either positive or negative acknowledgment after a carefully selected period of time. We adaptively determine appropriate delay times of mixed acknowledgments for network connections based on link and host properties.

## III. Transport Profiles

We collected source rates, throughputs, loss rates and retransmission rates of PLUT over the USN-ESnet hybrid channel as shown in Fig. 3, where each point in the horizontal plane corresponds to $(T(t),I(t))$. These profiles illustrate how the destination acknowledgment interval together with the source rate affects the transport performances over dedicated channels. In each transport experiment, we employ a fixed pair of $(T(t),I(t))$ such that $T(t)$ controls the source rate and $I(t)$ determines the ACK control period. By varying these two parameters, we measure the corresponding source rate, destination goodput, loss rate and retransmission rate over a 1 Gbps wide-area dedicated connection as plotted in Fig. 3. The ACK interval is measured in the unit of Round Trip Time (RTT) of the connection. Due to in-memory datagram retransmission, the source rate may temporarily exceed the link speed, which caps the goodput measured at the destination.

The peak throughput is achieved with low loss and low retransmission rate when $T(.)$ is slightly below 10 microseconds and $I(.)$ is above 20 times the round trip time; if parameters are manually set to values within these ranges, PLUT will achieve the peak throughput of approximately 950 Mbps. These ranges are dependent on the connection as well as host parameters, and can be manually identified if various profiles are *a priori* generated. However, profile generation typically is a very time consuming process (about 8 hours for this case), and the subsequent parameter selection requires active human involvement. In Section IV, we present SA methods to automate the process for parameter selection that bypasses both profile generation and manual parameter tuning.

The throughput in Fig. 3 is limited by the connection bandwidth. To illustrate a complementary case where throughput is limited by end hosts, we conducted experiments using 2GB file transfers over OC-192 links between two hosts located at ORNL and Chicago over USN. The measurements of source rate, destination goodput, and retransmission rate are plotted in Fig. 4, Fig. 5, and Fig. 6, respectively. Here, the peak throughput is limited by the file systems on end hosts to around 2 Gbps much below the connection bandwidth of 10 Gbps.

We have the following observations on these transport performance profiles: (i) at slow sending rates, i.e. large idle

*Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7232*
*http://dx.doi.org/10.4108/ICST.BROADNETS2009.7232*

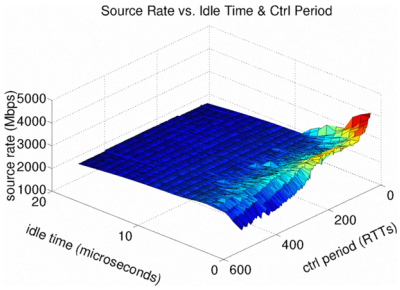


Fig. 4. Actual sending rate measured during file transfer in response to idle time and ACK control period.

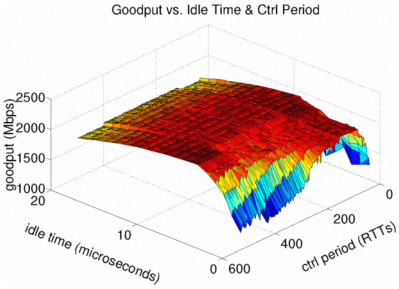


Fig. 5. Destination goodput measurement in response to idle time and ACK control period.

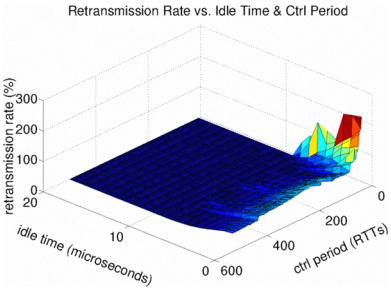


Fig. 6. Datagram retransmission rate in response to idle time and ACK control period.

times, the source rate and goodput almost linearly increase in response to the decreasing idle time with near-zero retransmission rate; (ii) at fast sending rates, i.e. small idle times, the transport performance critically depends on the ACK control period. Small ACK control periods cause frequent acknowledgment feedback and a large number of in-buffer datagram retransmissions[1], resulting in low goodput. On the other hand, large ACK control periods cause excessive delays and long lost datagram lists[2] and therefore sequential in-buffer datagram retransmissions, resulting also in low goodput. The main goal of transport protocol design for dedicated networks is to minimize datagram duplications, reduce retransmission rate, and increase destination goodput.

If the window size $W(t)$ is fixed to be one datagram, we have the rate-based control such that

$$r_S(t) = \frac{B \times MDS}{B \times T_i(t) + MDS}, \tag{1}$$

where the rate $B$ is determined by the speed of the host NIC in "writing" to connection. Besides the value set for idle time, other host factors such as disk I/O speed, buffer management process, and CPU scheduling policy may all affect the actual sending rate significantly. This is particularly true when high sending rates require peak processor utilization or the processor is shared with other CPU-bound applications, where the idle time itself may not be precisely enforced. Therefore, the destination goodput back-calculated from a user-specified target rate using Eq. 1 does not automatically match the target rate. Such host effects combined with variations in connection delays and finite computation periods lead to random components in the measured rates; these randomness components must be accounted for in automatic tuning of the parameters.

## IV. PLUT Transport Control

### A. Sender-receiver flow equations

We consider a steady state flow of packets from a sender to a receiver over a dedicated connection as shown in Fig. 7, wherein the time window over which various rates are computed are assumed to be sufficiently large to ignore the small window effects. In particular, rate variations due to jitter in packet delays caused by connection and host dynamics are assumed to be negligible, which is verified by our previous transport performance measurements over dedicated connections [18]. Let $r_S(t)$ be the rate at which packets are sent and let $l(t)$ be the fraction of them that are lost before being read by the receiver, and hence have to be retransmitted. Let $x(t)$ be the fraction of $r_S(t)$ that corresponds to retransmitted packets. Thus the flow $r_S(t)$ is composed of two streams of rates $g_S(t)$ and $x(t)r_S(t)$ corresponding to packets sent for the first time and retransmissions, respectively. In general the goodput $g_R(t)$ at the receiver depends on $r_S(t)$, $l(t)$ and $x(t)$, and there are three different regions:

(a) *No loss region:* Under very low sending rate, there are no losses and retransmissions as shown in Fig. 7(a) such that $g_R(t) = r_S(t)$, which results in low utilization.

(b) *Low loss region:* Under the peak utilization and low loss rate, the retransmitted packets are not lost as shown in Fig. 7(b). Thus we have

$$\begin{aligned} g_R(t) &= g_S(t)[1-l(t)] + r_S(t)x(t) \\ &= r_S(t)[1-x(t)][1-l(t)] + r_S(t)x(t) \\ &= r_S(t)[1-l(t)+x(t)l(t)]. \end{aligned}$$

When all lost packets are replenished in each window, we have $g_S(t)l(t) = r_S(t)x(t)$ and

$$\begin{aligned} g_R(t) &= r_S(t)\left[1-[1-x(t)]\frac{r_S(t)x(t)}{g_S(t)}\right] \\ &= r_S(t)\left[1-[1-x(t)]\frac{r_S(t)x(t)}{r_S(t)[1-x(t)]}\right] \\ &= r_S(t)[1-x(t)]. \end{aligned}$$

This is an optimal region to stabilize transport since the peak utilization is achieved with a low "wasted" bandwidth due to retransmissions.

(c) *High loss region:* Under the peak utilization and high loss rate, both original and retransmitted packets can be lost as shown in Fig. 7(c), and we have

$$\begin{aligned} g_R(t) &= g_S(t)[1-l(t)] + r_S(t)x(t)[1-l(t)] \\ &= r_S(t)[1-x(t)][1-l(t)] + r_S(t)x(t)[1-l(t)] \\ &= r_S(t)[1-l(t)]. \end{aligned}$$

[1]The high sending rates are due to in-memory datagram retransmissions.

[2]Buffer may remain full if the ACK is excessively delayed, hence temporarily halting the sending process, resulting in low sending rate.

*Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7232*
*http://dx.doi.org/10.4108/ICST.BROADNETS2009.7232*

In practice, however, the above rates are computed over finite window sizes, and the packets experience non-constant delays at the destination. Typically, jitter levels are more prominent over MPLS tunnels compared to SONET circuits, and heavily-loaded, shared end hosts lead to higher delay variations compared to dedicated hosts. Consequently, the estimators $\hat{r}_S(.)$, $\hat{g}_R(.)$ and $\hat{x}(.)$ computed at discrete time points are random variables with typically unknown distributions. Their values deviate from their long term averages, and in particular, they do not exactly satisfy the equalities such as $g_R(t) = r_S(t)[1 - x(t)]$ due to randomness. The effect of such randomness necessitates the utilization of stochastic approximation methods, which has a non-trivial effect on the underlying transport method: the step sizes used in parameter adaptation must be appropriately varied as per conditions such as in classical Robbins-Monro case[19]. In particular, methods that utilize fixed step sizes are not sufficient to guarantee optimal results in all but very simple cases. To take into account such random effects, we define *goodput-rate regression* as

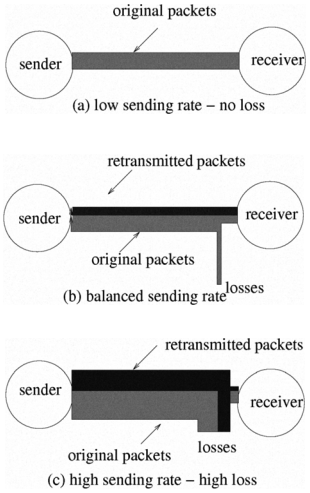


Fig. 7. Steady-state packet flows over a dedicated connection.

$$G_R(r) = E\left[\hat{g}_R(t) | r_S(t) = r\right],$$

and *goodput-ACK regression* as

$$G_I(a) = E\left[\hat{g}_R(t) | I(t) = a\right].$$

Similarly, we have *loss-fraction* and *retransmission-fraction regressions* defined as

$$L(r) = E\left[\hat{l}(t) | r_S(t) = r\right] \quad \text{and} \quad X(r) = E\left[\hat{x}(t) | r_S(t) = r\right].$$

Let $g^*$ be the maximum attainable goodput at the receiver over a given dedicated connection. The objective of PLUT control is to stabilize both $r(.)$ and $I(.)$ at suitable rates $r^*$ and $I^*$, respectively, such that:

(a) $G_R(r^*) = g^* = r^*[1 - X(r^*)]$, which ensures that peak throughput is attained at low loss rate, and
(b) $G_I(I^*) = \max_I G_I(I)$, which ensures that $I$ is optimally tuned.

We make the following assumptions about the regression functions and the underlying random process based on the measurements from Section III:

(A.1) The goodput-rate regression $G_R(r)$ is continuous and non-decreasing in $r$ in both zero and low loss regions, and both $L(r)$ and $X(r)$ are continuous and non-decreasing in $r$.
(A.2) The goodput-ACK regression $G_I(a)$ is a unimodal and differentiable function of $a$.
(A.3) The error magnitudes are bounded for $G_R(.)$, $G_I(.)$, $L(.)$ and $X(.)$. For example, $|\hat{g}_R(r) - G(r)| < \tau_G$ for all $r$ and some $\tau_G$.
(A.4) Error terms for $G_R(.)$, $G_I(.)$, $L(.)$ and $X(.)$ are not temporally correlated in the sense described in the next section using the martingale property.

We define two functions $\alpha(k)$ and $a(k)$ such that $\hat{g}_R(k) = \alpha(k)g^*$ and $\hat{r}_S(k) = a(k)g^*$.

The source rate control of PLUT at the sender is a two-step process corresponding to the above two objectives:

(a) In step one, the maximum goodput is estimated by $\hat{g}^*(k)$ at time step $k$, and $\hat{r}_S(k)$ is stabilized to achieve this goodput at a low loss rate. This step involves the adaptation based on monotone $G_R(.)$ and $X(.)$, which makes it suitable for Robbins-Monro [19] type stochastic approximation.
(b) In step two, the source rate is adjusted so that the measured goodput $\hat{g}_R(k)$ at the receiver stabilizes at the maximum achievable level for the given connection. This step involves the adaptation of $\hat{I}(k)$ based on unimodal $G_I(.)$, which makes it suitable for Keifer-Wolfowitz [19] type gradient descent method.

### *B. Source rate control for peak link utilization*

At time step $k$, for the measured source rate $\hat{r}_S(k)$, measured goodput $\hat{g}_R(k)$, and measured retransmission rate $\hat{x}(k)$, the equation $\hat{r}(k) = \hat{g}(k)/[1 - \hat{x}(k)]$ is only approximately satisfied. For $\hat{r}_S(k) = a(k) \cdot g^*(k)$ and $\hat{g}_R(k) = \alpha \cdot g^*(k)$, the coefficient function are typically $a(k) >= 1$ and $\alpha(k) <= 1$. Thus there are two possible estimates of $g^*(k)$ based on $\hat{r}_S(k)$ and $\hat{g}_R(k)$, which yield two different values. We consider the following general form that combines these two estimates:

$$\hat{g}^*(k) = [\hat{r}_S(k)(1 - \hat{x}(k))]^{\beta} \hat{g}_R(k)^{1-\beta}, \quad 0 \leq \beta \leq 1, \qquad (2)$$

where $\beta$ is determined by host and link properties. Typically, $\hat{r}_S(k)$ and $\hat{x}(k)$ are more stable compared to $\hat{g}_R(k)$ since the the former are not subject to connection-level variations. For the specific case where $\alpha(k) = 1/a(k)$, we have $\hat{g}^*(k) = \sqrt{\hat{r}_S(k)\hat{g}_R(k)}$. To account for randomness in measurements and the effects of delay and its variation of sending rate $\hat{r}_S(k)$ on goodput measurement $\hat{g}_R(k)$, we apply a dynamic version of Robbins-Monro method [19], [20] to adjust the source rate to achieve the target goodput $g^*(k)$ at the receiver:

$$\hat{r}_S(k+1) = \hat{r}_S(k) - \rho_k[\hat{g}_R(k) - \hat{g}^*(k)], \qquad (3)$$

where the time step adjustment coefficient is given by $\rho_k = b/k^{\gamma}$ for $0.5 < \gamma < 1.0$ and $b > 0$, a suitably chosen constant. The sending rate will increase if the measured goodput $\hat{g}_R(k)$ is less than the estimated maximum attainable goodput $\hat{g}^*(k)$ at low sending rates; while in the source rate control zone approaching the peak goodput, the goodput measurement may exceed the maximum goodput estimate due to increased retransmission rate, causing the sender to back off.

The step sizes satisfy the Robbins-Monro property namely, $\sum_{k=1}^{\infty} \rho_k = \infty$ and $\sum_{k=1}^{\infty} \rho_k^2 < \infty$. We assume that the errors satisfy the following martingale property for $\hat{r}_S(k) = r$:

$$E\left[\hat{g}(k) - \hat{g}^*(k) | \hat{r}_S(k) = r\right] = G_R(r) - [r(1 - X(r)]^{\beta} G_R(r)^{1-\beta},$$

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7232
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7232

which essentially assumes that the errors are not correlated across the time steps other than through $\hat{r}(.)$. Then the limit behavior of Eq. 3 is specified by the Ordinary Differential Equation (ODE) (Chapter 5, [19]):

$$\frac{d\hat{r}}{dt} = E\left[\hat{g}^*(k) - \hat{g}_R(k)\right] = E\left[\hat{g}^*\right] - G_R(\hat{r}).$$

Under low loss condition, we approximate

$$E[\hat{g}^*] = [\hat{r}(1 - X(\hat{r})]^{\beta} G_R(\hat{r})^{1-\beta}.$$

Then under the conditions (A.1), (A.3-4), the solution to ODE is given by the stationary point corresponding to

$$G_R(\hat{r})\left[1 - \left(\frac{\hat{r}[1 - X(\hat{r})]}{G_R(\hat{r})}\right)^{\beta}\right] = 0,$$

which in turn corresponds to $G_R(\hat{r}) = \hat{r}[1 - X(\hat{x})] = g^*$. Thus the limit behavior of this algorithm is to stabilize at sending rate $\hat{R}_S(k) \to \hat{r}$ such that $\hat{g}_R(k) \to g^*$ as $k \to \infty$. Alternatively, the required stability property can be derived for this algorithm using the monotonic property of $G_R(.)$ and $X(.)$ to show this convergence result as in [21]. Thus, this step ensures that PLUT probabilistically stabilizes at a high utilization rate $g^*$ of the connection while ensuring the low loss rate.

### C. Destination ACK interval control for goodput maximization

At the destination, we adaptively adjust the ACK interval $I(k) = I^*$ such that the goodput is maximized, i.e. $G_I(I^*) = \max_I G_I(I)$. The Kiefer-Wolfowitz Stochastic Approximation (KWSA) and Simultaneous Perturbation Stochastic Approximation (SPSA) have been shown to be very effective in solving such stochastic maximization problems whose gradient information cannot be directly obtained [22].

These two methods require collecting at least two measurements before making an adjustment on control parameters for the next time step. In transport control, however, it might be difficult to do so if the process dynamics change in the course of collecting two measurements for the gradient approximation. We apply a one-measurement form of SPSA to $I(k)$ at the receiver for goodput maximization as follows:

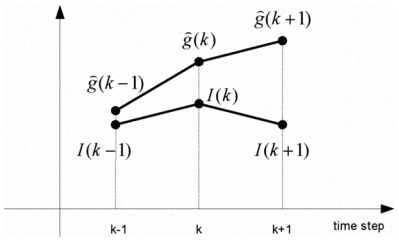


Fig. 8. Approximation of goodput gradient in the one-measurement SPSA.

$$\hat{I}(k+1) = \hat{I}(k) - c_k \hat{\mathcal{G}}(I(k)), \tag{4}$$

where $\hat{I}(k)$ denotes the ACK interval at time step $k$, $\hat{\mathcal{G}}(I(k))$ denotes the SP approximation to the gradient of goodput-ACK regression $G_I(.)$ and $c_k$ is a scalar gain coefficient such that $c_k \to 0$ as $k \to \infty$, $\sum_{k=1}^{\infty} c_k = \infty$ and $\sum_{k=1}^{\infty} c_k^2 < \infty$. As shown in Fig. 8, the goodput gradient in the one-measurement form of SPSA is approximated as:

$$\hat{\mathcal{G}}(I(k)) = \frac{\hat{g}_R(k) - \hat{g}_R(k-1)}{d_k \cdot (\hat{I}(k) - \hat{I}(k-1))}, \tag{5}$$

where $d_k$ is a positive scalar.

We assume that the error process satisfies the following martingale property:

$$E[\hat{\mathcal{G}}(I)|\hat{I}(k) = I] = \hat{\mathcal{G}}(I),$$

which is similar to the above case in that errors are not correlated in time domain except through $\hat{I}(.)$. Then under the conditions (A.2-4), the ODE specifying the limit behavior of Eq. 4 is given by

$$\frac{d\hat{I}}{dt} = E[\hat{\mathcal{G}}(\hat{I})],$$

which corresponds to the maximization of $G_I(.)$ [19]. Thus $\hat{I}(k) \to I^*$ as $k \to \infty$ such that $G_I(I^*) = \max_I G_I(I)$. The convergence to optimal $I^*$ is asymptotically and probabilistically guaranteed and does not depend on the knowledge of the underlying probability distributions.

## V. Implementation and Experimental Results

### A. Implementation

We now present the details of Sender Datagram Buffer Management (SDBM) and receiver acknowledgment scheme.

*1) Sender buffer management:* The initial PLUT implementation employs a simple static circular buffer management that allocates a buffer of datagrams and iterates that buffer for transmission. This has its merits: indexing with random access, fairly simple implementation, and no expensive allocation/deallocation of heap-dynamic memory. There is, however, one critical flaw in that the static buffer has a chance of "overflowing" in the case of a persistent datagram loss or delayed receiver retransmission request. When the flow window in the circular buffer cannot advance, the sender will not load datagrams until the first outstanding datagram is acknowledged, hence drastically reducing the overall transport throughput.

To solve this SDBM problem, we designed a new data structure — Three Tier Dynamic Queuing Buffer (3TDQB) and a dynamic buffer management scheme to not only fix but provide a failsafe measure for buffer overflow as well as low retransmission frequency. This 3TDQB is composed of: 1st tier – Linked Queue, 2nd tier – Linked List, and 3rd tier – Linked List. These three tiers implement three buffers: Datagram Queuing Buffer (DQ), Outstanding Pointer Buffer (OP), and Reload Pointer Buffer (RP), respectively. Note that the DQ is dynamically allocated at the start of the protocol to alleviate allocating memory from the heap during protocol runtime, while the OP and RP only contain the pointers to the datagram buffer space allocated in DQ. This process is shown in Fig. 9 via the DQ's four datagrams ($DG\_[1-4]$). Once allocated, the total number of nodes the DQ initially created will not change except under extreme circumstances.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7232
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7232

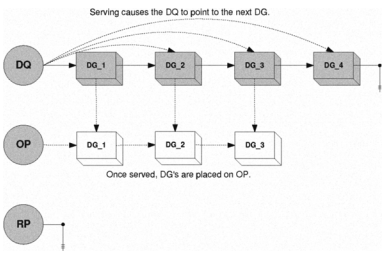


Fig. 9. Initial buffer states.

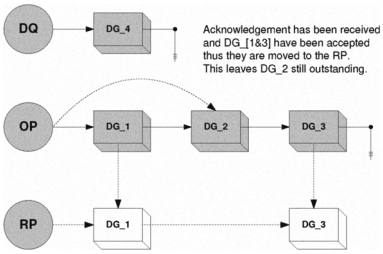


Fig. 10. Buffer states after receiving acknowledgements.

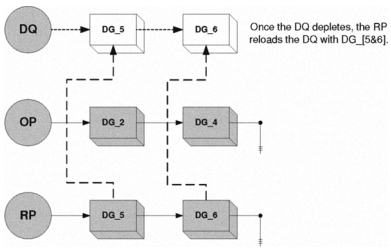


Fig. 11. Buffer states after reloading.

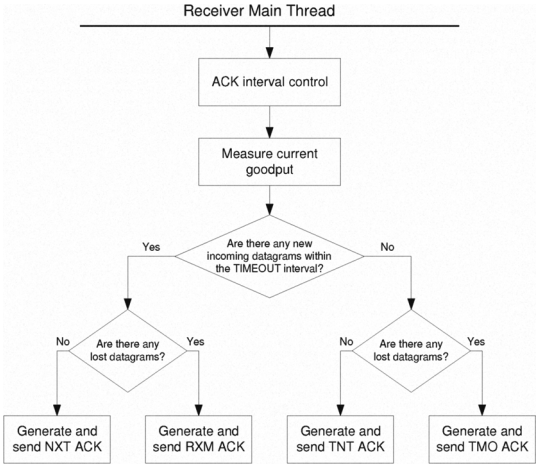


Fig. 12. Control flow diagram at the PLUT receiver.

The 3TDQB works in a descending manner. Each datagram will travel through the tiers consecutively before being flushed, reloaded and returned to the DQ. The DQ is a queue in which datagrams reside awaiting to be sent. As shown in Fig. 9, once served, the datagram will be placed (linked through a pointer) in the OP buffer where it waits for acknowledgment from the receiver. When acknowledged, the datagram is then placed in the RP buffer for flushing of the accepted data and reloading of the new data, as shown in Fig. 10. As the DQ depletes of all datagrams, the RP then reloads the DQ with an address change as shown in Fig. 11 so that the DQ now points to the flushed/reloaded datagrams on the RP, thus creating a new queue without new memory allocation.

The 3TDQB provides a failsafe feature to overcome the buffer overflow problem in the static circular buffer management scheme. Consider the following case: The DQ has served all remaining datagrams, the RP lies empty, and for some reason the OP has not yet received an acknowledgment. This is the 3TDQB's worst case scenario in that all datagrams reside in the OP buffer. This is similar to the circular buffer implementation, resembling a buffer overflow. To counter the overflow, the 3TDQB allocates datagrams from the heap to continue sending while waiting for acknowledgment from the receiver, thus never completely stopping the stream of data, just reducing the datagram sending rate.

*2) Acknowledgment types:* As shown in Fig. 12, we implement four different types of acknowledgment at the receiver: NXT (Next), RXM (Retransmission), TNT (Timeout Next), and TMO (Timeout Retransmission). For every normal ACK control period, if all datagrams received so far are in continuity, an "NXT" ACK is generated and sent to the sender; otherwise if there are lost datagrams (i.e. "holes" in the datagram checklist), the receiver compiles a list of lost datagram sequence numbers and sends them with a "RXM" ACK. If no datagram is received within a certain period of time, a timeout event is triggered where the receiver sends either a "TNT" ACK if all datagrams received so far are in continuity, or a "TMO" ACK enclosing the lost datagram sequence number list if there are "holes" in the datagram checklist. For all ACK types, the receiver measures the current instant goodput and sends it to the sender as part of the acknowledgment. On the sender side, for each incoming acknowledgment, we apply rate control as described in Section IV-B using the goodput measurements enclosed in the acknowledgment.

### B. Experimental results

*1) Wide-area dedicated connection:* We collect goodput measurements using iperf and PLUT over USN and ESnet. For iperf TCP, the number of streams $n$ is varied between 1 and 10, and for iperf UDP, the target rate is varied as 100, 200, ..., 1000, 1100 Mbps; each set of measurements is repeated 100 times. First, we compare USN and ESnet connections of lengths 3500 and 3600 miles, respectively, and their concatenation. TCP throughput is maximized when $n$ is around 7 or 8 and remained constant around 900, 840 and 840 Mbps for SONET, MPLS and hybrid connections, respectively. For UDP, the peak throughput is 957, 953 and 953 Mbps for SONET, MPLS and hybrid connections, respectively. Thus there is a difference of 60 Mbps and 4 Mbps between the TCP and UDP peak throughputs, respectively, over SONET and MPLS connections. There is a difference in peak throughput achieved by TCP and UDP in all cases, in particular, 57 and 93 Mbps for SONET and MPLS connections, respectively. This difference is in part due to the congestion control of TCP, and the high UDP bandwidth makes it a viable candidate for transport since there is no "congestion" on dedicated channels. We measured file transfer rates over these connections using PLUT, which achieved 954, 951 and 951 over SONET, MPLS and hybrid connections, respectively. The hosts used in these

*Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7232*
*http://dx.doi.org/10.4108/ICST.BROADNETS2009.7232*

| Dedicated channels | Goodput #1 (Gbps) | Goodput #2 (Gbps) | Goodput #3 (Gbps) | Goodput #4 (Gbps) | Goodput #5 (Gbps) | Goodput #6 (Gbps) | Goodput #7 (Gbps) | Goodput #8 (Gbps) | Goodput #9 (Gbps) | Goodput #10(Gbps) | Mean (Gbps) | Standard deviation |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ORNL-CHI USN EoS | 954.22 | 954.57 | 954.78 | 954.58 | 954.03 | 951.70 | 953.94 | 954.57 | 952.64 | 954.42 | 953.95 | 0.996 |
| CHI-SUN ESnet MPLS | 952.11 | 952.13 | 952.09 | 948.92 | 952.41 | 951.39 | 952.16 | 947.27 | 950.41 | 951.69 | 951.06 | 1.707 |
| ORNL-SUN EoS/MPLS Mixed | 952.14 | 952.03 | 952.11 | 951.67 | 949.29 | 952.02 | 950.32 | 952.85 | 951.06 | 946.73 | 951.02 | 1.829 |

Fig. 13. PLUT performance comparison with iperf.

TABLE I
PERFORMANCE COMPARISON OF PLUT, TCP, AND UDT.

| | Average Goodput / Std. Dev. (Mbps) | | |
|---|---|---|---|
| File Size (MBytes) | PLUT | TCP | UDT |
| 100 | 944.73 / 3.74 | 901.76 / 2.86 | 612.93 / 0.43 |
| 300 | 946.27 / 1.09 | 873.13 / 76.33 | 828.04 / 28.39 |
| 500 | 952.44 / 1.39 | 869.21 / 71.88 | 838.91 / 19.06 |
| 800 | 950.54 / 0.85 | 658.23 / 66.14 | 842.67 / 11.45 |
| 1000 | 947.27 / 1.30 | 689.86 / 2.42 | 840.91 / 9.35 |
| 1500 | 948.37 / 1.16 | 820.77 / 92.67 | 841.99 / 6.45 |
| 2000 | 946.08 / 13.14 | 632.57 / 26.58 | 840.17 / 5.78 |

experiments are Intel Xeon Linux workstations, each of which is equipped with 4 GB memory, four 3.2 GHz CPUs, one 1 GigE NIC, and one 10 GigE NIC. The results are summarized in Fig. 13. Thus PLUT is able to achieve actual file transfer rate within 3 Mbps of iperf UDP bandwidth estimate in all three types of dedicated connections.

*2) Local dedicated connection:* For performance comparison, we run PLUT, TCP, and UDT (version 4.4) on a local dedicated connection, which is provisioned by a back-to-back link between two Dell Precision 490 Linux boxes with kernel 2.6.20, each equipped with a 1 Gigabit NIC, dual Pentium 4 processors, 3 GBytes of RAM, and 1 TBytes of SCSI hard drive. For each file size, we run 10 tests using each transport method and collect corresponding goodput measurements. The average goodput measurements for 7 different file sizes are tabulated in Table I. We observe that PLUT consistently achieves higher goodput than TCP and UDT.

## VI. CONCLUSIONS

We described PLUT to support high-speed data transfers over dedicated channels. We conducted transport experiments over dedicated channels and generated detailed transport profiles to investigate the link, host, and protocol issues in the design of transport methods. To account for the random components in transport performance measurements, we designed control strategies based on SA at the source for rate control and at the destination for acknowledgment interval control to achieve sustained high goodput. We implemented and tested PLUT on dedicated channels provisioned as SONET, MPLS and hybrid connections, and in all cases PLUT achieved file transfer rates closely matching UDP iperf bandwidth measurements. These experimental results provided us valuable qualitative insights into both channel and host aspects of supporting data transfers over dedicated channels.

The area of data transfer protocols for dedicated channels is still in its infancy, because: (a) deployments of networks capable of providing dedicated channels are very limited, and (b) most efforts in protocol design target shared IP networks. An optimal utilization of dedicated channels requires an understanding of both channel and host properties together with a judicious tuning of protocol parameters. Unlike Hurricane transport protocol that employs manual parameter tuning, PLUT implements automatic rate and acknowledgment control. Rigorous analytical methods for the design and analysis of transport protocols specifically for dedicated channels would be of our future interest.

## REFERENCES


[1] User Controlled LightPath Provisioning. http://phi.badlab.crc.ca/uclp.

[2] N. S. V. Rao, W. R. Wing, , S. M. Carter, and Q. Wu, "Ultrascience net: Network testbed for large-scale science applications," *IEEE Comm. Mag.*, vol. 43, no. 11, pp. s12–s17, 2005.

[3] Dynamic Resource Allocation via GMPLS Optical Networks. http://dragon.maxgigapop.net.

[4] On-demand Secure Circuits and Advance Reservation System. http://www.es.net/oscars.

[5] Hybrid Optical and Packet Infrastructure, http://networks.internet2.edu/hopi.

[6] N. Rao, Q. Wu, S. Carter, and W. Wing, "High-speed dedicated channels and experimental results with hurricane protocol," *Annals of Telecommunications*, vol. 61, no. 1-2, pp. 21–45, 2006.

[7] S. Floyd, "Highspeed tcp for large congestion windows," Request for Comments: 3649, Dec. 2003.

[8] T. Kelly, "Scalable tcp: Improving performance in highspeed wide area networks," in *PFLDnets*, Feb. 2003.

[9] A. Kuzmanovic, E. Knightly, and R. L. Cottrell, "Hstcp-lp: A protocol for low-priority bulk data transfer in high-speed high-rtt networks," in *PFLDnets*, Feb. 2004.

[10] S. Low, L. Peterson, and L. Wang, "Understanding vegas: a duality model," *Journal of the ACM*, vol. 49, no. 2, pp. 207–235, March 2002.

[11] R. Stewart and Q. Xie, "Stream control transmission protocol," www.ietf.org/rfc/rfc2960.txt, Oct. 2000, iETF RFC 2960.

[12] SABUL. http://www.dataspaceweb.net/sabul.htm.

[13] Tsunami. http://newsinfo.iu.edu/news/page/normal/588.html.

[14] E. He, J. Leigh, O. Yu, and T. DeFanti, "Reliable blast udp: predictable high performance bulk data transfer," in *IEEE International Conference on Cluster Computing*, Chicago, Illinois, Sept. 23-26 2002.

[15] A. Banerjee, W. Feng, B. Mukherjee, and D. Ghosal, "Rapid: An end-system aware protocol for intelligent data-transfer over lambda-grids," in *Proceedings of the IEEE/ACM IPDPS*, 2006.

[16] E. He, P. V. Primet, and M. Welzl, "A survey of tranport protocols other than "standard" tcp," Global Grid Form Report, Tech. Rep., 2005.

[17] Q. Wu and N. S. V. Rao, "Protocol for high-speed data transport over dedicated channels," in *PFLDnets*, Feb. 2005, pp. 155–162.

[18] N. Rao, W. R. Wing, Q. Wu, N. Ghani, T. Lehman, and E. Dart, "Measurements on hybrid dedicated bandwidth connections," in *INFOCOM2007 Workshop on High Speed Networks*, 2007.

[19] H. J. Kushner and C. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*. Springer-Verlag, 2003, second Edition.

[20] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley Pub, 2003.

[21] A. Benveniste, M. Metivier, and P. Priouret, *Adaptive Algorithms and Stochastic Approximation*. Springer-Verlag, New York, 1990.

[22] J. C. Spall, "A one-measurement form of simultaneous perturbation stochastic approximation," *Automatica*, vol. 33, no. 1, pp. 109–112, 1997.



*Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7232*
*http://dx.doi.org/10.4108/ICST.BROADNETS2009.7232*