

Experimental Results On Data Transfers Over Dedicated Channels

Nageswara S. V. Rao, Qishi Wu, Steven M. Carter, William R. Wing
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6364
{raons,wuqn,scarter,wrw}@ornl.gov

Abstract

We describe experimental results on large data transfers and stable control streams over a dedicated 1Gbps channel of several hundred miles length implemented over ORNL-Atlanta production OC192 link. To support file transfers at 1Gbps rates, the end hosts are equipped with SCSI hard drives in a RAID 0 configuration and dedicated NICs, which are directly connected to the router. The 1Gbps channel is provisioned by configuring routers using the filter-based forwarding mechanism to implement an IP loop-back over the OC192 link. The performance profile generated from traffic measurements on the channel indicates non-zero packet losses and non-trivial jitter levels, both of which must be accounted for by the transport protocols to ensure high throughput and robust performance. We developed a UDP-based transport protocol by leveraging existing methods to achieve close to 100 percent channel utilization for file and data transfers. We also tested an existing protocol for implementing stable control streams over this channel. These results provide valuable insights into both the channel and host aspects of supporting data transfers over dedicated links.

1. Introduction

A number of large-scale scientific applications require throughputs within the range of 1-10Gbps and also stable control streams at much lower bandwidths, both over wide-area networks [2]. It is often believed that such capabilities are easier to realize over dedicated channels than shared IP networks. For example, compared to the Internet the challenge of achieving 10 Gbps application-level throughput over a dedicated OC192 link might be mitigated by the absence of competing traffic and fairness issues. In general, the promise of dedicated channels has resulted in a number of recent projects that provision dedicated channels of various characteristics; such projects includes UCLP

[8], CHEETAH [1], DRAGON [5], UltraScienceNet [9], and others. On the other hand, first hand experiences with exploiting the capacities of dedicated channels at the application level are very limited, particularly, for 1-10Gbps channels spanning several hundred miles. To make best use of these capabilities at the application level, it is very important to understand the end-to-end characteristics of these channels and their impact on the application performance. In particular, the packet loss/drop properties of the channels and of the end hosts are very important in designing and/or adapting transport protocols to achieve throughputs close to the channel capacities. Also, the jitter level experienced by applications is a very important factor that must be considered in designing transport protocols to stabilize data flows for control operations.

It is expected that there will be a proliferation of networks that provide on-demand dedicated channels for general users within next few years. In response, it is extremely important to design protocols that endow the applications the full power of dedicated channels in terms of high throughputs and/or stable streams. In this sense, the protocols play a much closer role to the applications compared to the more separated functionalities common in shared IP networks. To optimally utilize dedicated channels, however, it is important to understand the channel properties and their interactions with the hosts, including network interface card (NIC), kernel and application aspects. Our current experience of network protocols is mostly limited to the Internet environments. For dedicated channels in particular, the application-level experimental results are limited to testbeds with limited capacities and/or distances. As a preparatory phase for the upcoming Department of Energy (DOE) UltraScienceNet [9], we set up a testbed with a dedicated 1Gbps channel between two hosts located at Oak Ridge National Laboratory (ORNL) via an IP channel that loops back over ORNL-Atlanta OC192 link. Our objective is to perform experiments to understand the properties of dedicated channels as well as hosts for supporting data transfers at the rate of 1Gbps and stable control streams

at significantly smaller bandwidths. DOE UltraScienceNet plans to provide dedicated channels of 10 Gbps capacity at a resolution of 150 Mbps between ORNL, Chicago, Seattle and Sunnyvale. These channels will have much larger bandwidth and significantly longer footprint (3000 miles) compared to the ORNL-Atlanta-ORNL channel. But due to the scarcity of experimental results over realistic dedicated channels, our results provide a stepping stone for developing the technologies for the former.

While the dedicated channels obviate the need for congestion control, there are a number of important issues that critically affect the network performance observed at the application level:

- (a) **Capacity and Throughputs:** In general, the application level throughput are smaller than the channel capacities due to channel and host losses, which in turn are a function of sending rates at the source. As indicated by our measurements the loss rates are non-zero and often random ¹ with no apparent pattern, and they occur at various sending rates. Consequently, it is sub-optimal a priori fix the source sending rate right at the channel capacity. Instead, the source rate must be chosen and maintained at a level that results in the highest goodput at the destination. Such sending rates can be decided by experimentally building the so-called *throughput profile* [7] that displays the destination goodput as a function of source sending rate.
- (b) **Host Issues:** In addition to the link properties, a number of host components play a critical role in deciding the achieved throughputs or jitter levels, and their effects become particularly important at 1-10Gbps data rates.
 - (i) Because IP packets from the source application are copied into kernel buffers and then onto NIC, various buffer sizes together with the speeds and policies for clearing them can have an impact on the source rates and dynamics. At the receiver, the packets percolate from the NIC buffer to kernel buffer to application buffer. The application modules typically share the processor with other concurrently running applications and kernel processes. As a result, some of the newly arrived packets may be dropped at the NIC when the host processor is heavily loaded. Such unread packets are concluded as losses by the application.

¹In this paper, we refer to randomness in a broad sense that repeated measurements will result in different observed values with no simple way of predicting them. Statistically, the measurements could have been produced by an unknown distribution of arbitrary complexity, and no parametrization of the distribution is implied.

- (ii) The differences in the rates of NIC and provisioned channels can result in losses since most Ethernet cards do not support explicit rate controls. For example, a GigE card connected to a 100Mbps channel might send application packets at 1Gbps rates thereby exceeding the provisioned capacity. The excess packets will simply be dropped at the ingress router. On the other hand, even if streams require bandwidths much smaller than the channel and NIC capacities, the sending rate specified at the source might not be automatically guaranteed at the NICs, end routers/switches, and within the channel. This happens because the source packets might be clumped together along the channel, and as a result the actual flow rate might exceed the expected sending rate at least temporarily. Consequently, the packets may experience losses or jitter, both of which could appear random to the sender or receiver.
- (iii) Traditional storage devices and file systems on PCs used by average science users are not capable of supporting 1-10Gbps rates. For example, typical IDE disks provide peak I/O rates of 300 Mbps. However, higher data rates for file transfers can be achieved through striping data streams using clusters or RAID disks.
- (c) **Jitter and Stabilization:** When control operations are to be performed over network connections, it is very important that the packets flows be stable. Variations in delays, often termed as *jitter*, can destabilize transport flows and cause the loss of control. In particular, the packet losses can have a profound impact on the jitter levels. The lost packets have to be re-sent thereby increasing their net delays; thus the randomness in the loss process will be reflected in the jitter, which can also result in complicated delay dynamics [6]. While the losses over dedicated links are much less pronounced than over Internet, they still need to be explicitly accounted for in designing protocols for stable streams. In addition, the host losses must also be accounted for to ensure flow stabilization at the application level.

Effects of the above factors on applications and protocols can be assessed by conducting experiments over dedicated channels, which is a main focus of this paper. We describe an experimental 1Gbps dedicated IP channel implemented over ORNL-Atlanta production OC192 link to gain an initial understanding of the above issues. The channel is provisioned between two ORNL hosts that are directly connected to ORNL gateway router. This router is configured using the filter-based forwarding mechanism to implement an IP

loop-back channel over the OC192 link which is underutilized. The throughput profile of the channel is generated using measurements at the application level. This profile indicates packet loss which is a (random) function of the sending rate, and is also non-zero at various rates. The measurements also indicate non-trivial jitter levels even at low sending rates, which are to be expected due to non-zero and visibly random loss rates.

To support file transfers at 1Gbps rates, we equipped the end hosts with RAID 0 disk file system and dedicated NICs. We tested a number of existing protocols for high-throughput data transfers, including SABUL, tsunami, and UDT (see Falk et al [3] and references therein for a discussion on these and other high performance protocols). Among them the highest file transfer rate we achieved was 919Mbps. But both the Iperf measurements and performance profile indicated that the destination goodputs of the level 990Mbps are possible. Iperf is a tool to measure maximum TCP or/and UDP bandwidth, allowing the tuning of various control parameters and transport characteristics. Based on the principles of the existing UDP-methods, we developed a file transfer protocol with adjustable rate control parameters. By manually tuning these parameters we are able achieve 990Mbps file transfer rates, which correspond to close to 100% channel utilization. We also tested an existing protocol based on stochastic approximation for implementing stable control flows [7].

While our 1Gbps channel is limited in its capacity, span and capabilities, these experimental results provide us with valuable insights into both channel and host aspects of supporting data transfers over dedicated channels. For dedicated channels of much larger capacity and longer distance, we expect our qualitative results to hold although the actual loss and jitter levels might be quite different:

- (a) The throughput profile will be qualitatively similar in that losses will be non-zero and random at various sending rates, and jitter levels could be significant for control streams.
- (b) Host components play a significant role in the performance seen at the applications level.
- (c) Achieving data transfer rates close to the channel capacities would require a careful selection of control parameters and appropriate implementation of protocols.

These conclusions are somewhat straight-forward but indicate that careful protocol design and configuration are essential to achieving optimal throughputs over dedicated channels.

The organization of this paper is as follows. In Section 2, we describe the channel provisioning including the host and router configurations, and also present our channel measurements. In Section 3, we describe our protocol for

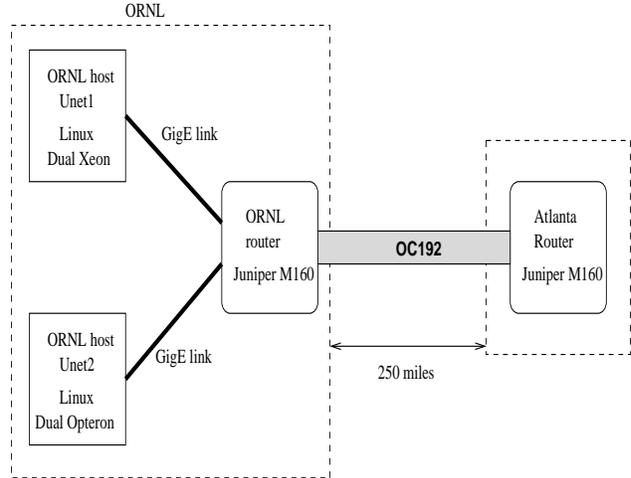


Figure 1. ORNL-Atlanta-ORNL dedicated 1 Gbps IP connection.

data transfers, and also discuss experimental results for both data transfers and stable streams.

2 Channel Provisioning

There are currently different types of dedicated channels and different mechanisms to provision them. The channels provisioned at SONET level are generally referred to as lightpaths [8]. These channels may be utilized by applications running on hosts which are connected to the channel via Ethernet links through a router, or a Multi Service Provisioning Platform (MSPP), or a similar device. Multi Protocol Label Switching (MPLS) and Virtual Local Area Networks (VLAN) tunnels provide another mechanism for providing dedicated channels at the IP level. Hosts with IP interfaces can be directly connected to such channels, and most common connections utilize the Ethernet. The capabilities of dedicated channels are currently being investigated for integration into production networks by a number of groups by utilizing Generalized MPLS (GMPLS) control plane [1, 5] or TL1-based control plane [9] with java or grid services interfaces [8]. Our ORNL OC192 connection, which carries the production traffic, is not endowed with such general-purpose control planes. Instead, we implement a single 1 Gbps channel by “hard” configuring the routers for the sole purpose of measurement and testing.

2.1 ORNL-Atlanta-ORNL Channel

Our testbed consists of two hosts, called unet1 and unet2, both located at ORNL. Each of them is equipped with a dedicated NIC which is connected to a GigE slot on a linecard

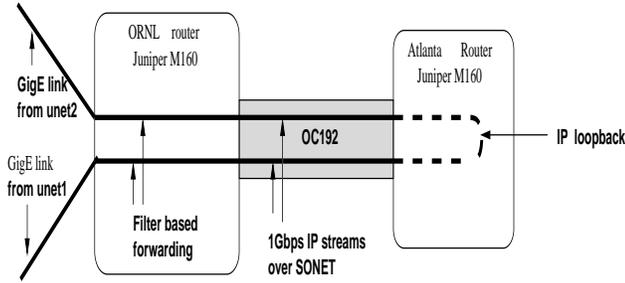


Figure 2. Router configuration for implementing dedicated channel

of Juniper M160 router located at ORNL as shown in Figure 1. There is an OC192 link from this ORNL router to another Juniper M160 router located in Atlanta, which is approximately 250 miles away. Only 1 Gbps of ORNL production traffic is currently carried on this OC192 link, and thus there is a spare bandwidth of 9 Gbps on this link. We utilize 2 Gbps of this spare bandwidth to implement a loop-back connection from ORNL to Atlanta back to ORNL. The traffic at each of the hosts is limited to 1Gbps due to the Ethernet connection. And the traffic flows from the hosts will flow unimpeded between the routers at ORNL and in Atlanta over the OC192 link. This arrangement effectively realizes a dedicated 1Gbps IP connection between unet1 and unet2 of approximately 500 miles in length.

Both unet1 and unet2 NICs have IP addresses belonging to a local subnet, and thus by default the IP packets between them are forwarded within the GigE linecard of the router itself. We changed this default routing so that the IP packets from each of these ports are statically forwarded to the output port of the OC192 linecard by utilizing the Filter Based Forwarding (FBF) capability of ORNL router. This is achieved by applying a firewall filter to each GigE port to incorporate a routing-instance that specifies the static route for all arriving packets to depart via the OC192 linecard.

The IP packets arriving at Atlanta router are handled by the default routes, namely, the packets from a ORNL host destined to other ORNL hosts are simply routed back at the OC192 linecard. Under this configuration, packets between unet1 and unet2 are routed along the loop-back path implemented over OC192 link as shown in Figure 2. While this configuration provides a dedicated 1Gbps channel between unet1 and unet2, it is not a lightpath or an MPLS tunnel in a strict sense. The underlying mechanism of this channel provisioning makes it more similar to an MPLS tunnel than a lightpath. On the other hand, when viewed from an end-host viewpoint, this configuration is quite similar to how typical PC hosts might be connected to utilize a dedicated SONET channel (lightpath), namely through an Ethernet interface.

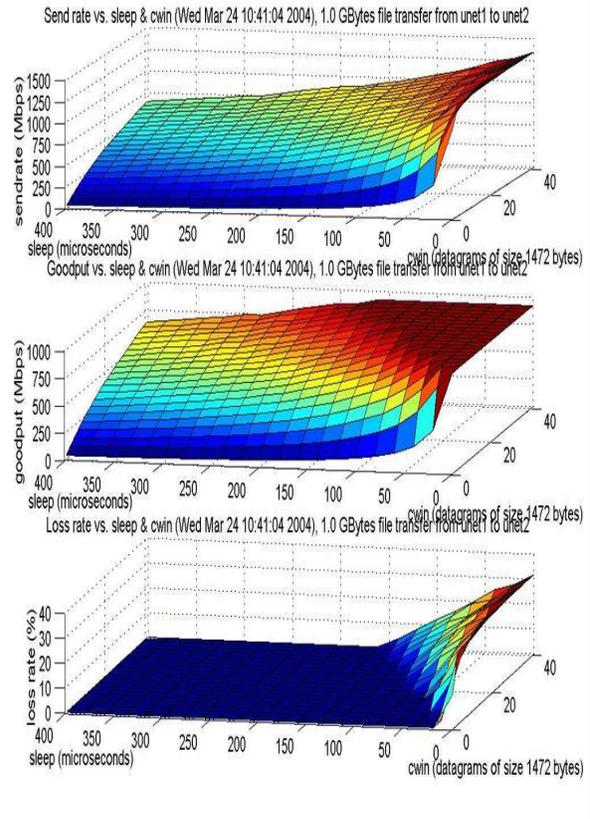


Figure 3. Measurements for ORNL-Atlanta-ORNL dedicated 1Gbps IP channel. Each point in horizontal plane represents sending rate given by window size and idle time pair. Top plot corresponds to sending rate, middle plot corresponds to the goodput at the destination and the bottom plot corresponds to the loss rate.

2.2 Channel Characteristics

We collected measurements to understand the channel and host properties that could be relevant for data transfers and stable streams particularly from an application perspective. Using a UDP stream with varying sending rates we measured the effective throughput, called the *goodput*, at the destination, and also the loss rate. The sending rate is controlled by transmitting a number of datagrams, denoted by the *window size* $W_c(t)$, in a single burst² and then waiting for a time period called the *sleep time* $T_s(t)$. Thus the sending rate is specified by a point in the horizontal plane, given by $(W_c(t), T_s(t))$, and its corresponding sending rate is shown in the top plot of Figure 3; note that sending rate is proportional to $W_c(t)$ and $1/T_s(t)$ as indicated by the monotonicity of this plot. The goodput measurements at the destination corresponding to various window size and sleep (idle) time pairs are shown in the middle plot, which is commonly known as the *throughput profile*. When the sending rate is small, the destination goodput increases with the sending rate, and reaches a plateau within the vicinity of 1 Gbps as shown in the right hand side of the throughput profile. In the bottom plot, the loss rate is shown as a function of the window size and idle time. The loss rates are near zero when the sending rate is low, but they becomes significant when the sending rate reaches the vicinity of 1 Gbps, where they monotonically increase with the sending rate. We also observed that the loss rates from multiple runs of an experiment with the same sending rate vary within a certain range even though the average trend was monotonic as shown in Figure 3.

Based on the measurements, one can draw two important observations:

- (a) For throughputs around the vicinity of 1 Gbps, suitable sending rate must be computed to achieve goodput plateau with minimal loss rate. From a transport perspective, the lost packets have to be identified and re-sent, and this is a process which consumes CPU resources, particularly so at high throughput rates. It is important to minimize this overhead activity to optimize the throughput, and this in turn involves utilizing a sending rate at a minimal loss rate. On the other hand, extremely low loss rates can only be achieved when the goodputs are significantly below 1 Gbps.
- (b) At all high sending rates, the losses are non-zero and random. Hence flow stabilization at these fixed target bandwidths requires explicit step size adaptation to

²It is possible to “space out” the packets evenly during the sleep time instead of sending them in a burst. In our experiments we did not observe significant differences between these two strategies but it is conceivable that they can result in different performances depending on how the waiting process is implemented and how heavily the host processor is loaded.

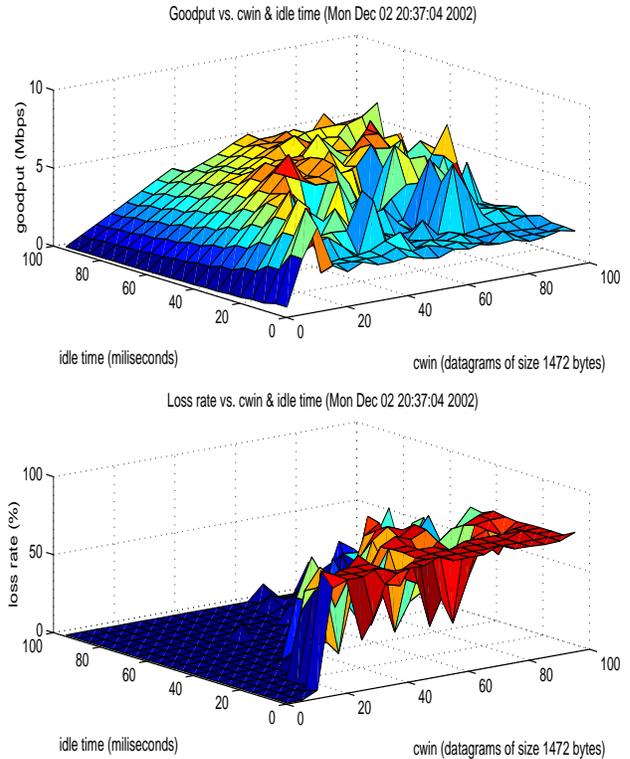


Figure 4. Measurements for ORNL-LSU Internet Connection. Each point in horizontal plane represents a sending rate given by window size and idle time pair. Top plot corresponds to the goodput at the destination and the bottom plot corresponds to the loss rate.

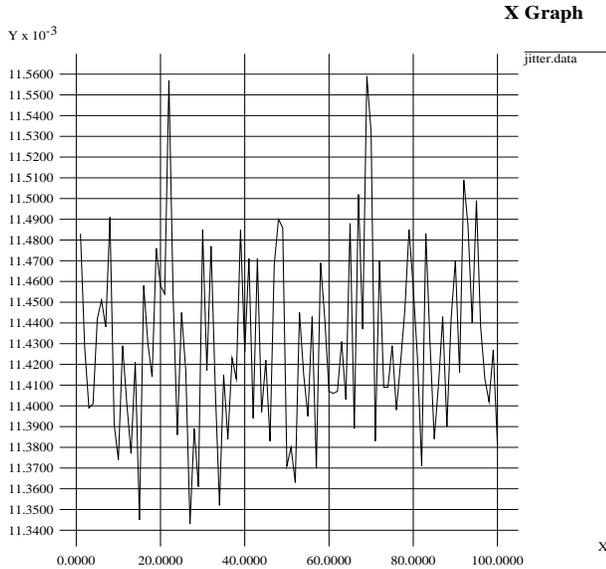


Figure 5. Jitter levels over ORNL-Atlanta-ORNL dedicated channel

achieve overall flow stability [7]. This stability is not particularly vital to data transfers but is extremely important in control streams.

It is instructive to compare this throughput profile with that observed for Internet connections [10]. The measurements collected over the Internet are shown in Figure 4 between ORNL and Louisiana State University (LSU). This connection runs over the OC192 link from ORNL to Atlanta, on Internet from Atlanta to Houston, and on a local network from Houston to LSU. There are two important features:

- (i) There is an overall trend of increase followed by decrease in the goodput as sending rate is increased. This overall behavior is quite stable although the transition points vary over time. It is to be noted that goodput for the dedicated channel reached a plateau and remained constant afterwards. For Internet connections, the goodput actually decreased when the sending rate is increased beyond a certain level.
- (ii) The plot is quite non-smooth mostly because of the randomness involved in packet delays and losses. The variation in the goodput is particularly high at high sending rates.

For applications involving interactive visualization and computational steering, the variation in the latency, namely jitter, plays an important role. High jitter levels can destabilize control loops. We sent packets of fixed sizes (10K) be-

tween the hosts and measured the application level delays. The variations are shown in Figure 5. The average delay is approximately 11 millisecc with jitter level of about 2%. While this jitter level is extremely low compared to Internet connections where the jitter levels could be as much as 30%, control streams for highly sensitive end devices could require an explicit handling of the jitter.

2.3 Host Configurations

The storage devices and file systems on unet1 and unet2 are carefully configured to achieve the file access speed of 1Gbps. Specifically, we implemented RAID 0 disk system on both hosts using dual SCSI hard drives and implemented xfs file system that achieved disk I/O rates in excess of 1 Gbps.

Note that the measurements in the previous section are collected at the application level, and hence they are subject to processor scheduling between the application processes and also between application and kernel processes. The measurements could be significantly affected if other applications are concurrently running on the hosts since the processor is shared between them. The plots in Figure 3 were collected when no other user programs are executed at the hosts, and in that sense represent the best case performance experienced by the applications. Our motivation is to utilize unet1 and unet2 as dedicated hosts for data transfers. If hosts were to be used as user workstations as well, the throughput profile must be generated under the normal host conditions. In general, additional applications running on the host will result in higher application-level losses and lower goodputs. Also, jitter levels shown in Figure 5 are observed when no other user level processes are running.

3 Protocols

We consider protocols for data transfers, both memory and file transfers, and stable control streams. Our original intention was to test a number of existing protocols and collect performance metrics. The default TCP throughputs were below 100 Mbps and could be improved by a factor of 2-3 with parameter tuning. Since dedicated channels do not have competing traffic, UDP-based protocols are more suited for these channels, although a careful parameter tuning was necessary to achieve goodput rates in the vicinity of 1Gbps. All UDP protocols we tested for file transfers required some manual parameter tuning to achieve throughputs close to 900 Mbps; this process required some understandings of the protocols and their implementations as well. The details were different among the protocols and it required significant efforts to gain even a partial understanding of the relationship between the parameters and the achieved throughput. On the other hand, the basic structure

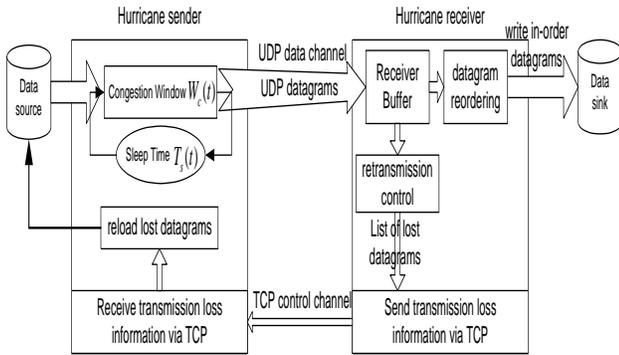


Figure 6. Hurricane transport control structure.

of the UDP-based protocols is quite similar and straight forward. We implemented our own version of a UDP-based protocol by significantly leveraging principles from the existing methods. This implementation provided us tunable parameters that were easier for us to tune.

For implementing stable flows, TCP is inherently ill-suited because by default it attempts to infer and occupy the available bandwidth, which is the entire channel capacity in case of a dedicated channel. The sending rate of TCP can be clipped to a desired level by suitably restricting the flow window sizes. If there are no losses, then TCP would indeed maintain the same sending rate. But as indicated by the throughput profile in previous section, the non-zero loss rates at various sending rates result in TCP underflow, since it interprets the loss as a congestion indication. Also, the randomness of the losses makes the TCP flow stabilization a difficult task. We tested the recently developed flow stabilization method [7] based on stochastic approximation which provided quite robust results as will be described in this section.

protocol	throughput
Tsunami	919 Mbps
UDT	890 Mbps
FOBS	708 Mbps

Table 1. Throughputs achieved by various UDP-based protocols for file transfers.

3.1 High Throughput Data and File Transfer

Recently researchers have been seeking solutions to develop UDP-based high-performance transport protocols that

overcome TCP’s throughput limitation. Such research efforts include SABUL, Tsunami, RBUDP, UDT and others (see [3] for an overview). These transport methods generally employ a UDP-based data channel to send application data (typically in a non-TCP-friendly manner) and a TCP-based control channel to exchange status information for source rate adjustment or lost packet retransmission. UDT employs UDP-mechanism for data and control information. We tested several of these protocols³ for file transfers and their peak throughput results are shown in Table 3. The best performance we achieved for file transfers is slightly above 900Mbps. But Iperf tool estimated the UDP bandwidth over this channel to be 992 Mbps, which represents the best performance achievable for data transfers. Also it was clear from the throughput profile in Section 2.2 that goodput rates of 990 Mbps are possible if the source rate is suitably maintained. We implemented a protocol, called *Hurricane* which is based on the basic ideas of existing protocols but provided us a more convenient parameter tuning mechanism. By an explicit manual tuning of the parameters, we were able to achieve goodput rates close to 990 Mbps (similar optimizations might be possible in other protocols as well). We believe that other protocols such as UDT can also be similarly tuned to achieve goodputs of 900 Mbps with an in-depth understating of various control parameters. Our decision to implement yet another UDP protocol is entirely pragmatic in that it took us less effort to write this code than understating and tuning the others to the extent we could.

Hurricane is developed exclusively for high-speed file transfer on dedicated links. It does not attempt to be TCP-friendly unlike some UDP-based protocols (UDT, for example) that are meant to be used over shared networks. Like other protocols in the same category, Hurricane employs a loose flow control mechanism and various (high) levels of persistent source rates. The design goal of Hurricane is to maximize link utilization without any expectation of sharing the channel. The architecture of Hurricane transport is illustrated in Figure 6.

The source rate $r_S(t)$ of a Hurricane sender is controlled by two parameters, congestion window size $W_c(t)$ and sleep time $T_s(t)$:

$$r_S(t) = \frac{W_c(t)}{T_s(t) + T_c(t)} = \frac{W_c(t)}{T_s(t) + \frac{W_c(t)}{BW}} = \frac{1.0}{\frac{T_s(t)}{W_c(t)} + \frac{1.0}{BW}} \quad (1)$$

where $T_c(t) = \frac{W_c(t)}{BW}$ is the time spent on continuously sending a full congestion window of UDP datagrams, which is determined by the congestion window size and link capacity BW , i.e. the maximum speed at which the NIC can generate the bit signal and put it on wire. According to Eq

³We were not able to obtain consistent results from some other UDP-based protocols due to incomplete executions.

Test link	MTU (bytes)	Target rate (Mbps)	Exp. #	Source rate (Mbps)	Goodput (Mbps)	Retxmt rate (%)
from unet1 to unet2	1500	50.0	1	49.69	49.56	0.016
			2	50.66	50.52	0.010
			3	50.75	50.60	0.020
		200.0	1	202.24	201.76	0.004
			2	202.86	202.38	0.005
			3	202.54	202.06	0.004
		500.0	1	504.32	501.84	0.008
			2	505.48	502.12	0.001
			3	506.73	505.40	0.001
		900.0	1	901.46	896.87	0.004
			2	895.13	890.57	0.003
			3	898.61	891.75	0.007
		950.0	1	945.61	934.15	0.007
			2	947.82	939.51	0.005
			3	950.56	945.17	0.004
from unet2 to unet1	9000	1000.0	1	991.24	990.48	0.032
			2	991.25	990.46	0.036
			3	991.27	989.41	0.083

Figure 7. Hurricane transport test results on unet1 and unet2.

(1), we may control the source rate $r_S(t)$ by adjusting either the congestion window $W_c(t)$ or sleep time $T_s(t)$ individually, or both simultaneously.

It is worth pointing out that in the design of Hurricane, we fix both window size and sleep time to achieve a flat source rate for a transport experiment with a specific bandwidth request. This is also true for those performance measurements collected in Figure 3. However, for transport protocols with more sophisticated rate control strategies such as the one we use later for stable streams, these control parameters are dynamically adjusted during one transport session. Hence, we associate these parameters here with time just for generalization purposes. Note that even with fixed window size and sleep time, the measured source rate is not likely to remain exactly the same over time due to the interactions between the kernel and processes.

A Hurricane receiver accepts incoming UDP datagrams, which are either written immediately to the local storage device if they arrived in order, or placed temporarily in a buffer for reordering otherwise. Whenever a control event is triggered, a sequential scanning is performed on the receiving buffer to check for a list of missing datagrams. The datagram ID numbers on this list are grouped together and sent over a separate TCP channel to the Hurricane sender. The sender then reloads the missing datagrams into the sender congestion window for retransmission upon the receipt of

such control strings.

It is worth pointing out that some concepts in the design of transport protocols are radically different for high-speed dedicated links. In the Internet environment, the bottleneck of data transfer is almost always the network (especially, access links and routers), and typical per-flow throughputs range from hundreds of Kbps to several tens of Mbps. Therefore, the designers have plenty of freedom to implement sophisticated rate control and buffering mechanisms without compromising the overall system performance. However, on a dedicated link whose speed is of the same order of magnitude (Gbps) as the clock rate of host processors (GHz), the system hardware and software configurations on end hosts (both sender and receiver) may affect the performance as much and sometimes more than the channel itself.

Specifically, we have observed that when the source rate approaches 1.0 Gbps, an inappropriately configured receiver equipped with 3.0 GHz dual CPUs was not fast enough to keep up with the data arrival speed, resulting in a large portion of packets being dropped by the kernel when moving them from the kernel buffer to the application buffer. For example, frequent requests for retransmissions places a large load in the process that receives datagrams on the destination side. On the other hand, an excessively long retransmission string delivered by TCP drastically slows down the sending process on the source side. Furthermore, the disk I/O speed on either side may ultimately restrict the overall transport performance to the maximum speed allowed by the relatively slow disk control device as described in the previous section. To account for the limitations posed by these host side factors, we carefully trigger a retransmission event based on the number of missing datagrams within a strategically determined time window of multiple RTT (round trip time) estimates, and we write only in-order datagrams on the fly onto the local storage device to sustain a near-peak receiving rate.

We conducted Hurricane transport experiments on 1Gbps dedicated link between unet1 and unet2 with various levels of target rates using a 2G bytes test file. For the test link from unet1 to unet2, the unet1 is the client (data sender) and the unet2 is the server (data receiver); vice versa for the test link from unet2 to unet1. Each experiment on one target rate is repeated for 3 times. The performance measurements for file transfers are listed in Figure 7. The high throughput and bandwidth utilization are achieved in both cases with reasonably low loss rates. Also, we obtain quite stable throughput when targeting at low rates. However, since the source side maintains a merely fixed sending rate with a very loose flow control mechanism, there is no guarantee of achieving stable throughput in all cases especially when the link is shared by other traffic. The transport control parameters in these experiments were manually tuned

for the best performance. We observed that the impact of parameter tuning on throughput and loss rate at source rates far below the peak bandwidth is not as sensitive as those approaching the peak bandwidth.

3.2 Stable Control Streams

The architecture of the stabilization protocol is similar to the one shown in Figure 6 except that the control channel for datagram acknowledgment is also built over UDP. The rate control is based on the Robbin-Monro Stochastic approximation method [4]. At time step $n + 1$, the new sleep time is computed as follows to update the sending rate to a new value (this method is described in detail in [10, 7]):

$$T_{s,n+1} = \frac{1.0}{\frac{1.0}{T_{s,n}} - \frac{a/W_c}{n^\alpha} * (g_n - g^*)}$$

where g^* is the target rate and g_n is the goodput measurement at time step n at the sender side. Coefficients a and α are carefully chosen so that the source rate specified by Equation 3.2 eventually converges to the target rate. The step size denoted by a/n^α must eventually become zero such that $a/n^\alpha \rightarrow 0$ as $n \rightarrow \infty$. But the rate of change must be controlled to be neither too fast such that $\sum_{n=1}^{\infty} a/n^\alpha = \infty$, nor too slow such that $\sum_{n=1}^{\infty} a^2/n^{2\alpha} < \infty$. Under these Robbins-Monro conditions on step sizes, it can be analytically shown that this protocol achieves the goodput stabilization at g^* under random losses and profiles similar to ones described in Section 2.2 (see [7, 10] for details).

We tested this method for flow stabilization on the same dedicated channel between unet1 and unet2. There is no competing traffic on this dedicated channel during the time of experiments. A set of control parameters $a = 0.8$ and $\alpha = 0.8$ are selected and the rate adjustments are applied on sleep time only. Instead of using the default MTU of 1500 bytes in the Internet, we use a MTU of 9000 bytes on this dedicated link. We conducted two stabilization experiments targeted at 1.0 and 10.0 Mbps, respectively. The initial sleep time is set to be 100 ms for each experiment and the window size is fixed at 2 datagrams, 6 datagrams, and 12 datagrams, respectively. The performance measurements of source rate and goodput are plotted in Fig. 8 where the time axis is in units of microseconds and the rate axis is in units of Mbps. In both cases, the goodput stabilized at the target rate within seconds and remained constant subsequently.

4 Conclusions

We described some experimental results on large data transfers and stable control streams over 1 Gbps dedicated

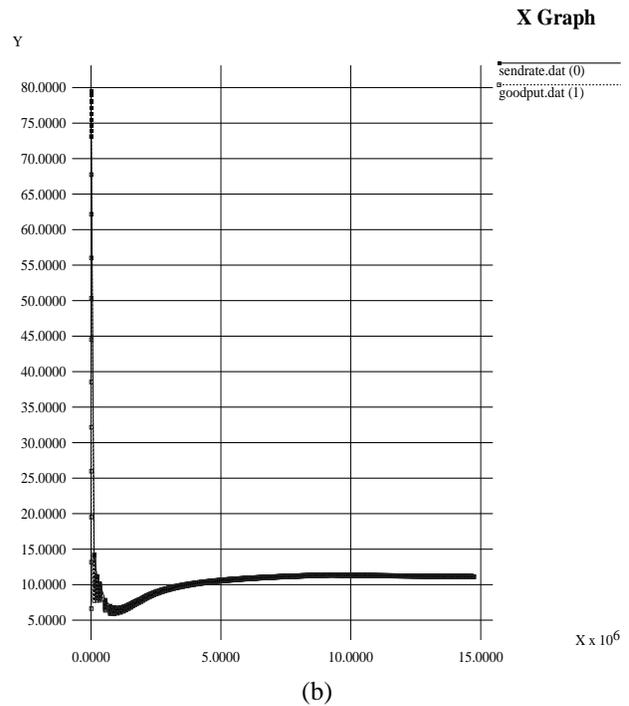
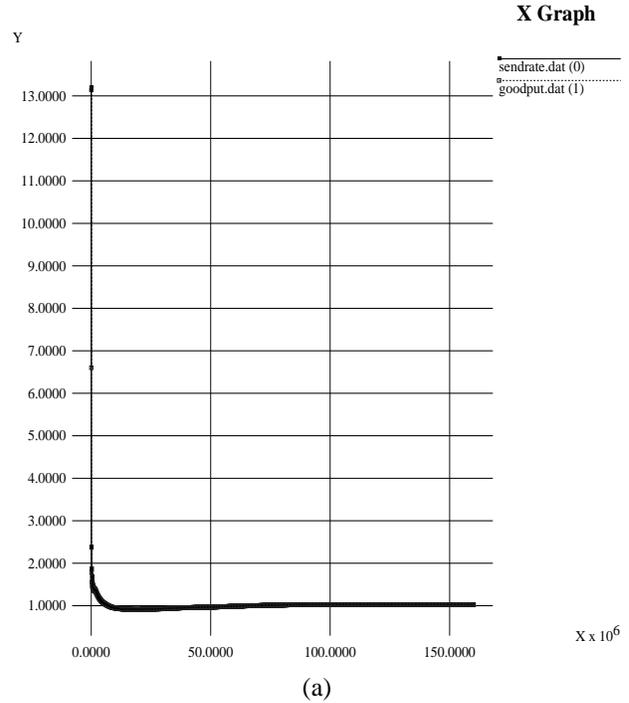


Figure 8. Stabilization over the dedicated link: target goodput is 1.0 and 10.0 Mbps in (a) and (b) respectively; $a = 0.8$, $\alpha = 0.8$, adjustment is made on sleep time.

IP channel implemented over ORNL-Atlanta production OC192 link. The provisioning task involved configuring the routers to realize a dedicated IP loop-back channel over the OC192 link using its spare capacity. We collected measurements over the channel and computed its throughput profile, which indicated both non-zero packet losses and non-trivial jitter levels. We presented throughputs achieved by a number of existing UDP-based transport protocols for file transfers; almost all of them needed some level of parameter tuning to achieve throughputs around 900 Mbps. We then described our protocol that achieved close to 100 percent bandwidth utilization for file and data transfers. We also tested an existing stabilization protocol for implementing stable control flows. These experimental results provided us with valuable qualitative insights into both channel and host aspects of supporting data transfers over dedicated channels. These overall results could be applicable to dedicated channels of much larger capacity and length. Due to the complicated interactions between the hosts and channels at these data rates, more analysis is needed to fully understand the measurements; these aspects are part of our ongoing efforts.

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 protocols area 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. Currently, most of the parameter tuning is accomplished manually, and it would be interesting to explore automatic methods for accomplishing the tuning. In applications such as interactive visualization, multiple streams may have to be supported on a dedicated channel, for example, a large visual channel to stream data to the users and a small but stable control channel for steering the visualization. It would be interesting to develop protocols that will implement such streams on dedicated channels. Note that the default TCP would not be able to optimally accomplish such tasks, since it does not achieve a stable control flow and does not achieve close to 100% utilization for the data stream. Rigorous analytical methods to help in the design and analysis of transport protocols specifically for dedicated channels would be of future interest.

Acknowledgments

Authors thank the anonymous reviewers for their constructive comments that resulted in a better presentation of the results in this paper. Authors thank Susan Hicks for implementing the filters on ORNL Juniper M160 router to implement the loop-back IP channel. This research is sponsored by the High Performance Networking Program

of the Office of Science, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC, the Defense Advanced Projects Research Agency under MIPR No. K153, and by National Science Foundation Under Grants No. ANI-0229969 and No. ANI-0335185.

References

- [1] End-To-End Provisioned Optical Network Testbed for Large-Scale eScience Application, <http://www.ece.virginia.edu/mv/html/files/ein-home.html>.
- [2] 2002. Report of the High-Performance Network Planning Workshop, August 13-15, 2002, <http://DOECollaboratory.pnl.gov/meetings/hpnpw>.
- [3] A. Falk, T. Faber, J. Banister, A. Chien, R. Grossman, and J. Leigh. Transport protocols for high performance. *Communications of the ACM*, 46(11):43–49, 2003.
- [4] H. J. Kushner and C. G. Yin. *Stochastic Approximation Algorithms and Applications*. Springer-Verlag, 1997.
- [5] NSF Shared Cyberinfrastructure Division PI Meeting, February 18-20, 2004, <http://hpn.east.isi.edu/nsf-sci>.
- [6] N. S. V. Rao, J. Gao, and L. O. Chua. On dynamics of transport protocols in wide-area internet connections. In L. Kocarev and G. Vattay, editors, *Complex Dynamics in Communication Networks*. 2004.
- [7] N. S. V. Rao, Q. Wu, and S. S. Iyengar. On throughput stabilization of network transport. *IEEE Communications Letters*, 8(1):66–68, 2004.
- [8] User Controlled LightPath Provisioning, <http://phi.badlab.crc.ca/uclp>.
- [9] DOE UltraScienceNet: Experimental Ultra-Scale Network Testbed for Large-Scale Science, <http://www.csm.ornl.gov/ultranet>.
- [10] Q. Wu. *Control of Transport Dynamics in Overlay Networks*. PhD thesis, Louisiana State University, 2003.