

MONITORING AVAILABLE BANDWIDTH OF UNDERLYING GRID NETWORKS

Marcia Zangrilli
Bruce B. Lowekamp, advisor
Department of Computer Science
College of William and Mary

Abstract

Harnessing the complete power of grids depends, in part, on an application's ability to adapt to changing network conditions. To this end, we are interested in monitoring available bandwidth of the underlying grid networks in the most accurate and least obtrusive way. Available bandwidth is either measured by actively injecting data probes into the network or by passively monitoring existing traffic, but there is a definite trade-off between the active approach, which is invasive, and the passive approach, which is rendered ineffective during periods of network idleness. Our solution is to develop the Wren bandwidth monitoring tool, which uses a hybrid approach to network monitoring that combines elements of passive and active techniques to offer accurate, timely available bandwidth measurements while limiting the invasiveness of probes. We have completed a packet trace facility, designed new passive bandwidth algorithms to measure the available bandwidth, and evaluated the effectiveness of these new algorithms in diverse environments. Our results indicate that a low overhead, passive monitoring system supplemented with active measurements can be built to obtain a complete picture of the network's performance.

1 Introduction

Grids are becoming increasingly important to research and engineering institutions because they provide a way of coupling various resources geographically distributed into a single high-performance environment. These grid infrastructures are valuable to chemists, physicists, engineers, and other scientists running large-scale simulations or other high-performance applications, such as mesh generators. These types of applications are often computationally or memory intensive, which may necessitate that data is distributed between many machines. Computational grids, such as the NASA Information Power Grid [1], are being developed to provide users with a secure way of sharing data and running intensive simulations. These grids provide more computational power and storage facilities, allowing scientists to tackle larger problems.

While switching to grid computing potentially provides more computational power and storage facilities, the evolution to grid computing introduces new problems for scientific applications because of the heterogeneity and dynamic nature of the grid resources. Grid environments are often composed of various clusters of heterogeneous machines interconnected by LAN and WAN networks. These environments can consist of machines with drastically different architectures connected by an unbalanced communication network. In addition, grids are dynamic in nature; resources can come and go. Pro-

processors may be shared with other users and may become unavailable at any time. Network conditions will reflect the amount of other traffic competing for the network as a result bandwidth may vary greatly in time. It is this dynamic nature of the grid that makes it challenging for an application to efficiently utilize all resources and maximize its performance.

Grid applications require timely network measurements so that they can adapt to changing network conditions and make efficient use of resources. While several tools have been created to actively measure the end-to-end available bandwidth of a network path, they require instrumentation at both ends of the path, and the traffic injected by these tools may affect the performance of other applications on the path. A less intrusive approach to measuring bandwidth is to passively monitor existing application traffic, however this approach is rendered ineffective during periods of network idleness. *We propose a hybrid approach to network monitoring that combines elements of passive and active techniques to offer accurate, timely available bandwidth measurements while limiting the invasiveness of probes.*

As part of the Watching Resources from the Edge of the Network (Wren) system, we are developing a monitoring tool that utilizes an architecture where application-generated traffic is captured and used for performance prediction when possible, and when no existing traffic is available, active probes are used to provide a constant series of measurements [6]. This architecture allows an application to modify its adaptation strategy at runtime based on knowledge of additional available bandwidth and report its measurements to a monitoring system for use by other or future applications.

In order to implement this hybrid system, we need to monitor existing application traffic, calculate bandwidth based on the information collected, and detect when there is insufficient traffic so we can then inject active probes. This paper provides an overview of the Wren packet trace facility and describes how to apply techniques commonly used to actively measure bandwidth to passive traces of application traffic, a task complicated

because we have no control over the application traffic pattern. We have evaluated our approach on LAN- and WAN-based distributed applications and present the results of applying our analysis to traffic captured from these applications. The principle contributions of our initial research are

- a system that captures packet traces unobtrusively,
- new algorithms for applying the principles of active probing techniques to passive traces, and
- evaluation of our passive, two-sided available bandwidth algorithm when applied to several types of application traffic on paths with varying amounts of cross traffic.

The remainder of this paper will describe the status of the Wren bandwidth monitoring tool. We will first review terminology and related work on monitoring systems. In Section 4, we describe our packet trace implementation and the new passive, available bandwidth algorithms we developed. In Section 5, we present the results of using Wren to monitor available bandwidth.

2 Terminology

The utilization of the network path refers to how much of the capacity is being consumed by traffic. Available bandwidth describes what portion of the path is currently unused by other competing traffic. More precisely, available bandwidth is determined by subtracting the utilization from the capacity of the network path [7, 11]. In practice, available bandwidth may also be affected by traffic shapers that allow some traffic to consume more or less bandwidth than other traffic can consume.

Another important measurement is achievable bandwidth, which is the throughput an application can actually obtain over a path. As the majority of applications rely on TCP for communication, most measurements of achievable bandwidth are made by measuring the throughput of a bulk data transfer using TCP. These measurements are intrusive, having a significant impact on the performance of other applications, but they are the

only direct way to measure what a real application will receive if it attempts the same type of data transfer.

Achievable bandwidth and available bandwidth are related and are both useful characteristics in analyzing network and application performance. TCP throughput depends on the available bandwidth, the other traffic utilizing the network, the TCP implementation, and the way the application sends the data. Available bandwidth is a measurement of how much additional traffic could be added without perturbing other network connections and is useful for applications that are considering increasing their utilization, a new application approximating what bandwidth might be available, and for network engineers monitoring the performance of their network.

3 Monitoring Systems

Packet trace tools, like tcpdump, monitor network traffic to verify the operation of network protocols and to characterize application traffic patterns. In Linux systems, tcpdump uses libpcap to interact with the Linux Socket Filter (LSF), a variation of BSD Packet Filter (BPF) [9], which is composed of a network tap that sits at the link device layer and collects packets specified by user-defined filter rules. The use of the LSF filtering mechanism improves performance because unwanted packets are filtered in the kernel instead of being copied into the user-level for processing by libpcap. A drawback of using LSF to trace packets is the need for applications to be reading the socket to collect the packets as they arrive. In contrast, an application that uses the Wren system can read data from a kernel buffer at any point after the trace is completed. More importantly, it may be difficult for system that uses LSF to coordinate traces of the same range of packets on two machines. We have designed Wren with a triggering mechanism that specifies the same range of packets will be monitored on both machines.

Shared Passive Network Performance Discovery (SPAND) [13] uses information passively collected from several hosts on a network to measure network conditions. Performance data is collected from client applica-

tions and packet capture hosts. The performance reports sent by the client applications are based on application-level observations and may lack the detail to provide accurate estimates of available bandwidth. Packet capture hosts, which use BPF to observe all traffic to and from a group of hosts, are the primary means of collecting information. These hosts are not at the end-points of the path, and therefore, must use heuristics to infer end-to-end properties such as achievable bandwidth. The packet capture host is responsible for processing the data before sending the performance report to the server. In our Wren system, the host collecting the kernel-level packet trace can send the data to another machine where all the processing will occur. More importantly, Wren uses a two-sided approach to monitoring traffic at the both end hosts of path, allowing for more accurate measurements of end-to-end properties.

Web100 [8] is designed to monitor, diagnose, and tune TCP connections. Web100 comprises a kernel-level component, which is responsible for exposing the characteristics of the TCP connection, and a user-level component, which retrieves and graphically displays the connection information. The Web100 tool instruments the network stack of the 2.4 series of Linux kernels to capture an instantaneous view of the TCP connection internals and exports that information to the user-level through an interface in the /proc file system. The Web100 tool has an auto-tuning functionality and also provides a mechanism for hand-tuning kernel variables. The appeal of the Web100 tool is the ability to track the current state of variables in TCP connections and to tune buffers accordingly in real-time at the user-level.

We are developing the Wren bandwidth monitoring tool as an extension to the Web100 kernel so that a single kernel can provide the variety of network services required for high-performance networking. We chose to implement the kernel-level portion of the Wren bandwidth monitoring tool as an additional feature to Web100 because monitoring available bandwidth and buffer tuning are both used in the same situations to improve application performance.

Table 1: Information collected during Wren packet trace of TCP traffic.

Incoming packets				Outgoing packets			
timestamp	seq number	ack number	TCP cwnd	timestamp	seq number	ack number	data size

4 Wren Bandwidth Monitoring Tool

Our Wren bandwidth monitoring tool is designed to incorporate the key portions of packet tracing into the kernel, with the analysis and management of the traces handled at user-level. The design goal is easy deployment and integration in a secure, production system so that an individual user can monitor only packets associated with their application. We set two primary requirements for our implementation.

- No modification to the application code or the network infrastructure may be required. Code modification is one of the largest hurdles to grid application development. Our implementation should be as transparent to the user as possible.
- Application performance must not suffer. The monitoring software should not compete with the application for network or CPU resources.

We believe that a system meeting these design goals would be as acceptable to system administrators as Web100 [8], a system commonly used to monitor and tune TCP connections.

4.1 Wren Packet Trace Facility

Wren extends the functionality of the Web100 kernel to incorporate kernel-level packet traces. Because Web100 is a secure, well-accepted system, our Wren packet trace facility should be acceptable to system administrators. In our implementation, we added a buffer to the Web100 kernel to collect characteristics from incoming and outgoing packets. Table 1 shows the information collected during a trace of TCP traffic. In the UDP and TCP code we timestamp the packets using the CPU clock cycle counter and record the timestamps and TCP congestion window size in our buffer. Access to this buffer is

through two system calls: one starts the tracing and specifies the connection to trace and the other retrieves the trace from the kernel. The Wren architecture is designed so that user-level components do not need to be reading from the packet trace facility all of the time. Wren can capture full traces of small bursts of traffic or periodically capture smaller portions of continuous traffic and use those to measure available bandwidth. We have previously analyzed the efficiency of our design [14] and found that Wren adds little overhead to the kernel.

The precise timestamp and cwnd values are only possible through kernel-level instrumentation. Another key design feature of the Wren packet trace facility is the ability to coordinate measurements between machines. In our implementation, one machine triggers the other machine by setting a flag in the headers of outgoing packets to start tracing the same range of packets. The other machine traces all packets with the header flag set, which prevents lost packets from adversely affecting the coordinated tracing. This packet coordination ensures that the buffers at each end of the connection store information about the same packets regardless of the bandwidth-delay product of the particular connection.

4.2 Wren Trace Analyzer

The user-level component of the Wren bandwidth monitoring tool is responsible for initiating the tracing, collecting the trace from the kernel, and processing the data. We apply the bandwidth techniques at the application level to avoid slowing the performance of the operating system. The overhead imposed by the user-level code is minimal, and the data can be transferred to another machine for processing, if necessary.

4.2.1 Existing Available Bandwidth Tools

Self-induced congestion (SIC) is a common technique used by active tools to measure available bandwidth. The basic principle of the SIC technique is that if packets are sent at a rate larger than the available bandwidth, the queuing delays will have an increasing trend, and the rate the packets arrive at the receiver will be less than the sending rate. If the one-way delays are not increasing and the rate the packets arrive is the same as the sending rate of the packets, then the available bandwidth is less than or equal to the sending rate. Tools that utilize this concept probe the network path for the largest sending rate that does not result in queuing delays with an increasing trend because this sending rate reflects the available bandwidth of the path.

Tools that use principles similar to self-induced congestion include Netest [5, 4], Pathchirp [12], Pathload [3], PTR [2], and TOPP [10].

4.2.2 Wren's Passive SIC Algorithm

The Wren trace analyzer applies principles of SIC algorithms to passive traces of TCP traffic. The major obstacle with using passive TCP traces is that we have no control over the traffic pattern. Because we lack control over the traffic patterns produced by TCP applications, the key question is whether or not the collected TCP traffic can be used to apply the SIC principles. In this chapter, we address this question by comparing the probe traffic injected by active SIC algorithms with TCP traffic. We found the primary differences between probe traffic and the TCP traffic traces are:

- Active SIC algorithms use UDP traffic, but we are tracing TCP traffic,
- The probe traffic is designed to be unintrusive, but the TCP traffic we monitor will affect other traffic on the network path, and
- The probe traffic can be controlled by the active SIC algorithms, but we cannot control the application traffic we monitor.

We have developed a new passive SIC algorithm that uses the timestamps of packets on the sending and receiving hosts to calculate the one-way delays and the initial sending rate of the stream of packets. Our implementation is based on the pathload tool [3], which sends out UDP probes and uses trends in one-way delays to determine the available bandwidth.

We group 10–100 packets into a stream and identify the trend in one-way delays of that stream. We impose the condition that grouped packets are the same size so that all packets we consider have experienced the same store-and-forward delays at the links along the path. Because congestion window size often determines the sending rate of the TCP application, we also ensure that all packets grouped together have the same congestion window size. For each stream of packets, we calculate the one-way delays of each packet, calculate the initial sending rate, and determine if there is an increasing trend in the one-way delays.

We group several streams together and try to identify the maximum value for the available bandwidth. For each group, the stream with the largest sending rate and no increasing trend determines the available bandwidth. In our passive SIC implementation, we have the flexibility to relax the number of packets in a stream and the number of streams in a group to better suit the amount of traced traffic available to analyze. However, this feature is not demonstrated in this paper.

4.3 Security

We recognize that the interface to packet traces that the Wren bandwidth monitoring tool provides may be considered a security issue. In the current implementation of our Wren bandwidth monitoring tool, there is no restriction on which users can capture traces. The danger here is that any user has access to and can trace any other user's application traffic. However, the amount of information the user can obtain is limited to sequence and acknowledgment numbers and does not include the data segment of the packets. This is not much more information than a user could obtain from the netstat program.

Were we to restrict access, deploying a grid-wide monitoring system would require either root access or restrict the monitoring to a single user’s applications. But in a production release, we could add the ability to check permissions for access.

5 Passive SIC Evaluation

The measurements our SIC algorithm produces reflect the amount of bandwidth available to the application. These measurements are the sum of the bandwidth currently being consumed by the application we are monitoring and the amount of bandwidth not being used by any other traffic.

For this paper, we monitor TCP traffic produced by traffic generators. The available bandwidth we measure could be different if the TCP stream we monitor is removed because TCP may interact with and affect the other traffic on the path. However, our goal is to use our algorithm with preexisting network traffic. We are emulating for control purposes in these experiments.

We demonstrate how our algorithm works by applying our algorithm to traffic monitored from a traffic generator that sends TCP traffic with an average throughput of 15 Mbps. We ran the traffic generator for 90 seconds on a 100 Mb LAN and plot the measurements our passive SIC generates in Figure 1. In this graph, there are distinct bands that reflect the change in the amount of cross traffic. Figure 1 clearly demonstrates that our passive SIC algorithm can detect changes in available bandwidth.

Because we run our experiments on an isolated testbed, we can use the capacity of the testbed and throughput of the cross traffic application to determine the actual amount of available bandwidth present. The error metric we use is absolute error:

$$error = |cross\ traffic\ tput - abw\ estimate|$$

We applied this error metric to the measurements produced in Figure 1. Table 2 shows the mean error and the standard deviation obtained from comparing our passive bandwidth measurements with the available bandwidth present on the path. In this table, the error is quite low

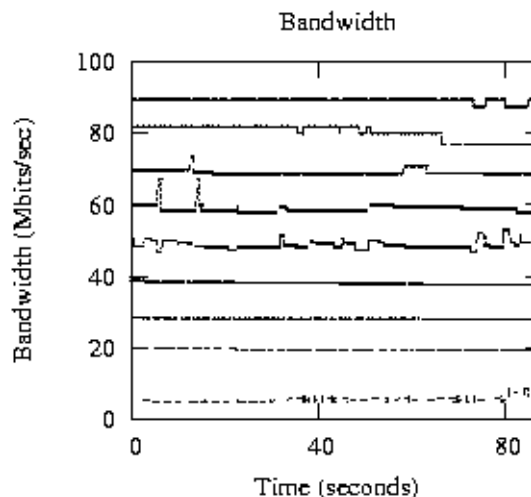


Figure 1: The passive SIC technique applied to traces of TCP traffic on 100 Mb LAN with 10, 20, 30, 40, 50, 60, 70, 80, and 90 Mbps of cross traffic present. The average throughput of the TCP traffic on an uncongested LAN is 15 Mbps.

indicating that our passive approach to measuring available bandwidth produces valid measurements.

5.1 Bursty Traffic

Our previous experiments used traffic patterns consisting of uniformly spaced messages, but in practice many application have a more bursty communication pattern. We have previously analyzed our algorithm using real application traffic and found applications with bursty communication patterns more challenging to monitor [15]. However, if the bursty application traffic has either large message sizes or bursts of smaller message sizes sent larger than the available bandwidth, we can use our passive SIC algorithm to accurately measure the available bandwidth.

To emulate traffic generated by bursty applications, we created a traffic generator that sends 256K messages with a variable delay. The variable delay causes the throughput of the generators to oscillate.

Table 2: Error of measurements obtained by monitoring TCP traffic with average throughput of 15 Mbps

cross traffic (Mbps)	Mean error	Std. Deviation
10	0.912	0.599
20	1.626	1.028
30	1.191	0.418
40	0.849	0.511
50	1.592	1.068
60	1.940	0.310
70	1.761	0.096
80	0.429	0.172
90	4.187	0.887

In Figure 2, the line represents the throughput of the traffic generator and the points are the measurements produced by our passive algorithm. In this experiment, there is 20 Mbps of cross traffic present for the first 15 seconds and 40 Mbps of cross traffic present for the last 15 seconds. This graph shows that our SIC algorithm can accurately measure changes in the available bandwidth using traces of application traffic with bursty communication patterns.

6 Conclusion

This paper discusses the status of the Wren bandwidth monitoring tool. We describe the implementation of the Wren kernel-level packet trace facility and our new passive SIC algorithm for measuring available bandwidth. We have evaluated our algorithm using bulk data transfer traffic, bursty traffic, and traffic on paths with controlled congestion and present the results of how responsive our technique is to changes in cross-traffic on the path. Our results demonstrate that our passive approach is able to accurately measure the available bandwidth.

Our future work will focus on developing one-sided measurements, determining when to inject supplemental traffic, and combining all of the components together to create a real-time hybrid monitoring system. We believe one-sided measurements will enable us to provide

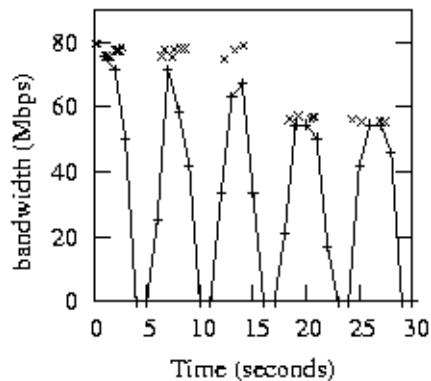


Figure 2: The graph shows the SIC measurements (points) produced by monitoring bursty application traffic (tput represented as the line) on a testbed with 20 Mbps of cross traffic present during 0–15 seconds and 40 Mbps of cross traffic during 15–30 seconds.

more timely measurements because the overhead associated with coordinating traces on two machines is removed. We have a prototype implementation of the one-sided algorithm and are in the process of evaluating the algorithm to determine how much traffic is needed for our algorithm to produce accurate bandwidth measurements. Based on the evaluation of the traffic types required by our algorithms, we will develop an algorithm that detects when there insufficient traffic present and injects supplemental traffic only when necessary, thus limiting the invasiveness of our probes. The next step is to incorporate online trace analysis in a real-time system that will provide a constant stream of accurate available bandwidth measurements.

References

- [1] NASA Information Power Grid (IPG) <http://www.ipg.nasa.gov/>.
- [2] HU, N., AND STEENKISTE, P. Evaluation and Characterization of Available Bandwidth Techniques. *IEEE JSAC Special Issue in Internet*

- and WWW Measurement, Mapping, and Modeling (2003).
- [3] JAIN, M., AND DOVROLIS, C. Pathload: a Measurement Tool for End-to-end Available Bandwidth. In *Proceedings of the 3rd Passive and Active Measurements Workshop* (March 2002).
- [4] JIN, G. Algorithms and Requirements for Measuring Network Bandwidth. LBNL Report: LBNL-48330.
- [5] JIN, G., AND TIERNEY, B. Netest: A Tool to Measure the Maximum Burst Size, Available Bandwidth and achievable Throughput. In *International Conference on Information Technology Research and Education* (2003).
- [6] LOWEKAMP, B. B. Combining Active and Passive Network Measurements to Build Scalable Monitoring Systems on the Grid. *Performance Evaluation Review* (March 2003).
- [7] LOWEKAMP, B. B., TIERNEY, B., COTTRELL, L., HUGHES-JONES, R., KIELMANN, T., AND SWANY, M. Enabling Network Measurement Portability Through a Hierarchy of Characteristics. In *Proceedings of the 4th International Workshop on Grid Computing (GRID2003)* (2003).
- [8] MATHIS, M. "Web100" <http://www.web100.org>, 2000.
- [9] MCCANNE, S., AND JACOBSON, V. The BSD Packet Filter: A New Architecture for User-level Packet Capture. In *USENIX Winter* (1993).
- [10] MELANDER, B., BJORKMAN, M., AND GUNNINGBERG, P. A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks. In *Global Internet Symposium* (2000).
- [11] PRASAD, R., MURRAY, M., DOVROLIS, C., AND CLAFFY, K. Bandwidth Estimation: Metrics, Measurement Techniques, and Tools. In *IEEE Network* (June 2003).
- [12] RIBEIRO, V., RIEDI, R. H., BARANIUK, R. G., NAVRATIL, J., AND COTTRELL, L. pathChirp:Efficient Available Bandwidth Estimation for Network Paths. In *Passive and Active Measurement Workshop (PAM)* (2003).
- [13] STEMM, M., KATZ, R., AND SESHAN, S. SPAND: Shared Passive Network Performance Discovery. In *INFOCOMM* (2000).
- [14] ZANGRILLI, M., AND LOWEKAMP, B. B. Comparing Passive Network Monitoring of Grid Application Traffic with Active Probes. In *Proceedings of the 4th International Workshop on Grid Computing (GRID2003)* (2003).
- [15] ZANGRILLI, M., AND LOWEKAMP, B. B. Using Passive Traces of Application Traffic in a Network Monitoring System. In *Proceedings of the Thirteenth IEEE International Symposium on High Performance Distributed Computing (HPDC 13)* (June 2004), IEEE.