PASSIVE END-TO-END PACKET LOSS ESTIMATION FOR GRID TRAFFIC MONITORING

A. Papadogiannakis, A. Kapravelos, M. Polychronakis, E. P. Markatos
Institute of Computer Science, Foundation for Research & Technology – Hellas
P.O. Box 1385, 71110, Heraklio, Greece
{papadog, kapravel, mikepo, markatos}@ics.forth.gr

A. Ciuffoletti
INFN-CNAF
Viale Berti Pichat 6/2, 40126, Bologna, Italy
augusto@di.unipi.it

Abstract Accurate network monitoring is vital for the operation of Grids. The packet loss ratio is among the most important metrics for identifying poor network conditions, since it highly affects data throughput performance and the overall end-to-end data transfer quality. In this paper, we present a scalable and non-intrusive technique based on passive network monitoring for estimating the packet loss ratio between different measurement points. The proposed approach is complementary to current active monitoring techniques and can be easily incorporated into the network monitoring components of Grid systems. We describe the design and implementation of the technique, outline its integration within a Grid environment, and present experimental evaluation results, including measurements with real Grid application traffic.

Keywords: Grid connectivity, network monitoring, passive monitoring, packet loss.

1. Introduction

Accurate monitoring of network characteristics, such as delay, packet loss rate, and available bandwidth, is critical for the efficient operation of modern Grid systems, which are usually composed of many resources interconnected by local area networks or, more often, through the Internet. One of the most important network performance metrics is the packet loss ratio. Packet loss occurs when correctly transmitted packets from a source never arrive at the intended destination. Packets are usually lost due to congestion, e.g., at the queue of some router, routing problems, or poor network conditions that may

result to datagram damages. Packet loss affects significantly the data transfer throughput and the overall end-to-end connection quality. Consequently, it is desirable to have accurate packet loss measurements for the network paths that Grid services use, in order to timely identify network inefficiencies.

Most of the existing techniques for packet loss estimation are based on *active* network monitoring, which usually involves the injection of a certain number of packets into the network for measuring how many of them are lost [1, 15, 16]. Such active monitoring tools incur an unavoidable network overhead due to the injected probe packets, which compete with the real user traffic.

In contrast to above approaches, in this paper we present a novel real-time, end-to-end packet loss estimation method based on distributed *passive* network monitoring. Our approach does not add any overhead to the network since it passively monitors the network traffic without injecting any probe packets. At the same time, it estimates almost in real-time the *actual* packet loss faced by the active applications. Moreover, it offers the capability for measuring the loss rates of particular Grid services, allowing for fine-grained per-application packet loss estimation, which is important in case different applications on the same path face different degrees of packet loss.

The rest of the paper is organized as follows. Section 2 describes in detail the design and implementation of the proposed approach. In Section 3 we outline the integration of the proposed technique within a Grid network monitoring infrastructure. Section 4 presents experimental evaluation results in a controlled environment and preliminary results with real Grid application traffic. Finally, Section 5 summarizes related work and Section 6 concludes the paper.

2. Passive End-to-End Packet Loss Estimation

Traditionally, passive monitoring tools operate at a selected vantage point in the network that offers a broad view of the traffic, such as the access link that connects an Autonomous System to the Internet. Besides monitoring a single link, emerging applications can benefit from monitoring data gathered at multiple observation points across a network [5, 9, 11]. Such a distributed monitoring infrastructure can be extended outside the border of a single organization and span multiple administrative domains across the Internet [17].

Grids can benefit from such a distributed passive monitoring infrastructure by deriving useful network metrics regarding the network conditions between different domains. These metrics include, the Round-Trip Time [12], per-application throughput, packet retransmissions [6], packet reordering [13], one-way delay and jitter, and packet loss ratio. In this paper, we focus on the passive estimation of the packet loss ratio between different domains. In the remaining of this section we discuss the advantages of a passive packet loss measurement approach and describe in detail the design and implementation of our approach.

2.1 Passive Packet Loss Measurement Characteristics

An inherent property of passive network monitoring is that it does not disrupt the existing traffic conditions. This non-intrusive nature of passive measurements makes them completely invisible on the network. Moreover, our passive packet loss estimation method exhibits several other advantages over active packet loss measurement techniques, which we discuss in the following.

Real-time measurement of the *actual* packet loss ratio. The proposed technique measures the actual packet loss faced by the traffic of an active application in real-time, as it passes through the passive monitors. In contrast, active monitoring approaches unavoidably disrupt the current traffic due to the probe packets. Thus, they can measure potential temporary side effects that may be caused by the injected traffic.

Scalability. In a Grid infrastructure, it is desirable to measure the end-to-end packet loss between many different resources or domains. In a system with N resources, the number of required end-to-end measurements grows with $O(N^2)$, since, as a general rule, each resource has a distinct path to any other resource. For active monitoring, it is clear that as the number of resource pairs increases, the injected traffic incurs a significant disruption in the network, so usually such measurements are performed sequentially, measuring one or a few paths at a time. In contrast, a passive monitoring approach can provide an instant estimation of the packet loss ratio across different paths, independently of their number.

Per-application measurement. Using appropriate filters, the proposed approach can measure the packet loss faced only by the traffic of a particular Grid service. This capability is of particular importance for cases in which different services may exhibit different packet loss ratio in the same path. This can happen due to the use of differentiated services, rate-limiting devices, or load-balancing configurations.

IP-level measurement. In contrast to techniques that passively estimate the loss ratio based on properties of the TCP protocol [2–3], our approach measures the packet loss at the IP layer, so it can also work for UDP or any other Transport Layer protocol.

Besides the above positive properties, our approach has also certain limitations. A necessary operational requirement is the presence of two passive monitors at the ends of the measured path. If passive traffic monitoring is not feasible in some domain, then we should rely on active monitoring tools. Furthermore, the presence of real traffic in the measured path is mandatory for the operation of our approach, since it measures the packet loss faced by the

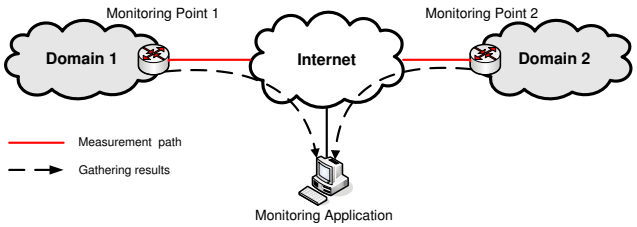

Figure 1. End-to-end architecture for passive packet loss estimation.

existing traffic. It is clear from the above that our approach is complementary to existing active probing techniques, and both approaches can perfectly coexist.

2.2 Approach

We adopt an end-to-end approach for estimating the packet loss ratio using two passive monitors at the two ends. The overall approach is shown in Figure 1. The two monitoring points gather information about the packets passing through them. Periodically, this information is sent to a central application which correlates these results and estimates the packet loss ratio.

A naive packet loss algorithm in this environment would just count at both ends the number of packets in each direction between the two domains, and then periodically subtract the number of packets received at the destination from the number of packets that were actually sent, and vise versa. However, this simple method has a major drawback: we cannot accurately synchronize the two monitoring points to count the same window of packets. Suppose that both passive monitors start and stop counting packets at exactly the same time. When they start counting, some packets are already in transit. These packets were not counted at the sender side, but they will be counted at the receiver, so the packet loss ratio will be underestimated. Similarly, when the measurement stops, the in transit packets will have been counted by the sender, but will be missed by the receiver, so the packet loss ratio will be overestimated. A possible solution would be to start and stop the measurement in the receiver's monitoring point after a delay close to the network's one-way delay. However, this solution is still inaccurate due to the network delay variability.

In order to solve the above problem, we take a different approach by measuring the packet loss of each *flow* separately. For the TCP and UDP protocols, a *flow* is defined as a set of IP packets with the same protocol, source and destination IP address, and source and destination port (also known as a 5-tuple). For protocols that do not support ports, a flow is defined only by the protocol and source and destination IP address. A flow is considered *expired* if no packet has arrived for that particular flow within a specified timeout (60 sec in our experi-

ments). In case of TCP, a flow can also be considered expired if the connection is explicitly closed, i.e., when an RST of FIN packet is seen.

Each of the two monitoring sensors classifies the IP packets into flows, according to the above definitions. In periodic time intervals, both sensors send statistical information about the identified *expired flows* to the monitoring application. Since expired flows are well defined, the monitoring application can correlate the statistics gathered at both sensors regarding the *same* expired flow. Thus, for each pair of statistics regarding the same expired flow, the application computes the packet loss for that flow based on the difference of the number of packet that each expired flow reports. This gives an accurate measurement of the *actual* packet loss faced by the particular traffic flow.

2.3 Implementation

2.3.1 Passive Monitoring Platform. In each measurement point we need a passive traffic monitoring platform for the identification and collection of the expired flows. We have implemented our prototype using MAPI [14], a flexible passive monitoring API. A communication daemon, part of the distributed MAPI version [17], is responsible for accepting monitoring requests from remote applications and sending back the corresponding results. Using this distributed monitoring API (DiMAPI), we are able to manipulate multiple monitoring sensors from the same application.

2.3.2 Identification of Expired Flows. Every packet is associated with exactly one flow. At each sensor, the monitoring daemon keeps a record for each active flow in a hashtable for fast lookup. In addition to the 5-tuple, a flow record holds the timestamps of the first and last packet of the flow. The arrival time of the last packet of the flow is necessary for deciding whether the flow has expired or not. Finally, the record holds the number of packets and bytes of the flow, from which we compute the packet and byte loss ratios.

For every new packet, the monitoring daemon looks up the corresponding flow record in the hashtable, increases the packet counter, and adds the packet size to the existing byte counter value. Also, the timestamp of the last packet is renewed. In case a flow record is not found, a new one is created.

In order to identify immediately the expired flows, the monitoring daemon maintains a linked list that contains pointers to the flow records in a temporal order. For every new packet, the timestamp of the last packet in the corresponding flow record is renewed, and that flow comes first in the linked list. A separate thread in the monitoring daemon runs periodically (e.g., every one second) and finds the expired flows in the end of that list. Starting from the last entry of the list, it checks whether the timestamp of the last packet of that flow is older than the specified timeout, and if so, it removes it from the list and puts it in the expired flow list. The same process is continued until a non-expired

flow is found. Finally, the monitoring daemon sends the list with the expired flows to the monitoring application.

2.3.3 Distributed Monitoring Sensor Management. The last component of the architecture is the monitoring application. The application collects periodically the expired flows from the distributed monitoring sensors, correlates them, and reports the packet loss ratio for every pair of sensors. The application uses a hashtable, similar to the one described earlier, for identifying pairs of statistics from different sensors for the same expired flow. For every matched pair, it subtracts the number of packets that they measured to compute the packet loss for this flow. Finally, the application reports the total packet loss ratio between pairs of measurement points and also the packet loss per every individual flow. It reports the byte loss ratio as well, which can be also an interesting metric for some applications.

3. Passive Packet Loss Measurement as a Network Monitoring Service

In this section we define how the packet loss measurement is configured inside a grid-wide Network Monitoring Service. This service is based on a Network Monitoring Element (NMElement), which is a Grid element that concentrates the Network Monitoring functionalities of a Grid: it offers an interface for measurement requests coming from applications (in the following we call them brokers, although this is not required in our model), and a plug-in based interface for publishing measurements. It has access to a database that contains the description of the domain partitioning of Grid resources, and the persistent attributes of other NMElements. A detailed description can be found in. [4].

The definition of a Network Monitoring session aiming to measure the packet loss ratio is composed of the following elements: the identifiers of the source and destination domains, the description of the type of service for which the packet loss measurement is requested, and the time period of the measurement: this can be historical, most recent, one-shot, or periodic. Certain combinations of these attributes are also allowed.

In principle, a measurement is not targeted to a flow between two specific hosts: the domain partitioning should guarantee the significance of the measurement for any pair of hosts in the two domains.

Figure 2 illustrates the message exchange between the agents that participate to the measurement, as described in the following. Messages are represented by arrows, in which the attached numbers are referenced in the following text.

The application that needs the measurement will send a measurement request (**2**) to one of the the NM services in charge of monitoring the request between the two domains. This can be either the source or the destination of the flow

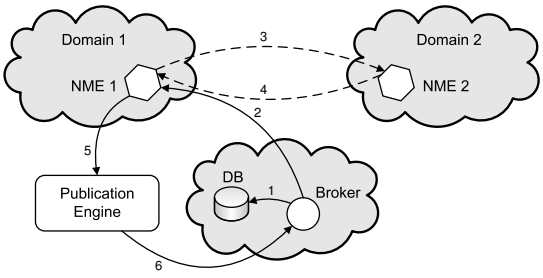

Figure 2. Embedding the packet loss measurement in a Network Monitoring Service

for which we want to compute the packet loss ratio. The information regarding the identity of the NMElement, necessary in order to handle the request, is first retrieved with a query (**1**) to the NM Database attached to a NMElement in the domain of the requesting broker.

When the Network Monitoring service receives such request, it first checks the availability of the module in charge of managing the measurement. The information is retrieved from an internal registry of available modules. The next step consists of verifying the availability of resources dynamically allocated to monitoring tasks: this information is retrieved by inspecting the current system state (using ps/netstat like commands).

In case any of the above steps fail, a "resource not available" reply is returned to the calling resource broker. This indicates that the measurement was not performed, but does not imply anything about the availability of the inspected resource. The application will redirect the request to another NMElement, or will repeat it using less demanding parameters. If the measurement is feasible, the successive step consists of locating a peer NMElement: the selection is carried out using the local NM Database, by querying for the peer NMElement, which is identified by the (source domain, destination domain) pair.

The measurement can be either extracted from a local cache of available results, or actually come from a new measurement. In the former case, the historical result is found as indexed by the Network Element, complemented by measurement attributes indicated in the request of the broker. Otherwise, a request for the activation of the peer module for packet loss measurement is delivered to the peer (**3**). In case of a negative reply, this is bounced back to the requesting broker. Otherwise, the measurement will proceed normally. The peer module will send back the measured data for packet loss estimation (**4**), with the process described in the previous section.

The result of the measurement is finally streamed outside the NMElement, either to the GIS, or to any other publication media (**5**), according to the available plugin in the NM Service module. The final step is the delivery of the result to the requesting broker (**6**).

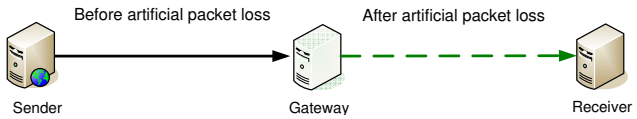

Figure 3. Using a transparent intermediate host for artificial packet loss generation.

Table 1. Validation results with artificial packet loss for UDP traffic.

Generated Loss (%)	**Estimated Loss** (min/avg/max %)	**Measurement Error** (%)
0	0.00/0.002/0.01	0.002
1	0.91/0.98/1.06	0.020
5	4.80/5.014/5.13	0.014
10	9.86/10.09/10.18	0.090
25	22.24/24.74/25.32	0.260

4. Experimental Evaluation

4.1 Packet Loss Measurement Accuracy

In our first experiment, we aim to explore the accuracy of our method and verify that it measures the actual packet loss of the traffic without significant errors. We validate our approach in a controlled environment using three PCs connected to a $100Mbit/s$ network, as shown in Figure 3. The "Sender" host generates traffic destined to the "Receiver" host. The traffic of both hosts is monitored by two passive monitors.

All traffic between the two hosts is transparently forwarded through the third "Gateway" PC. The purpose of the Gateway PC is to generate artificial packet loss in the path between the sender and the receiver, by dropping a specified percentage of the packets in both directions. Artificial packet loss is produced using the `netem` tool [10]. `Netem` is a kernel module that allows the emulation of various network characteristics, such as delay, loss, duplication, and re-ordering, through the Linux queuing discipline. We use different percentages of packet loss between 1% and 25%.

First, we generated UDP traffic in the lossy path using the `nttcp` tool. Table 1 presents the results for 10 parallel UDP flows, with about $10Mbit/s$ rate for each flow. We repeated the measurements every 12 minutes over a two hour period. We present the average, minimum and maximum packet loss measured, as well as the measurement error, which is defined as $error = |expected\ loss - measured\ loss|$.

Table 2. Validation results with artificial packet loss with HTTP traffic.

Generated Loss (%)	**Estimated Loss** (min/avg/max %)	**Measurement Error** (%)	**Served Requests**	**Rate** (Mbit/s)
0	0.00/0.06/0.17	0.060	2944	38.75
1	1.02/1.078/1.16	0.078	1666	22.23
5	4.92/5.07/5.23	0.070	1058	14.11
10	9.86/10.086/10.12	0.086	290	3.90
25	24.89/25.235/25.50	0.235	0	0.26

Next, we generated more realistic TCP traffic by performing normal HTTP requests using the `apache benchmark` (`ab`) tool [8]. We ran a web server at the receiver host and `ab` at the sender, downloading a file of $1MB$ size. We performed the same measurements for packet loss ratios that vary again from 1% to 25% while running `ab` for 10 minutes for each measurement. The concurrency level was equal to 10, that is, 10 parallel HTTP transfers were active at any given time. Table 2 shows the estimated packet loss ratios, for both directions, and the respective measurement errors. Each measurement was repeated for 10 times. We also present the average number of completed requests and the average transfer rate for each case, as reported by `ab`.

We observe that in both cases our technique produces accurate results, very close to the real packet loss introduced to the link. `Netem` imposes the specified packet loss ratio using a probability function, so we cannot expect to produce exactly the specified loss rate. This is the main reason for the slight aberrations from the expected loss value.

Another observation from the results in Table 2 is that the packet loss ratio affects significantly the number of requests that were completed and the TCP throughput. For instance, when we add 1% packet loss in the link, the number of accomplished requests is almost half than without packet loss. In case of 10% loss rate, only 290 (out of the 2944) were completed, while in case of a significant 25% loss, none of the requests could be completed. We can see that as the generated loss increases from 0% to 25%, the rate of serviced requests drops dramatically from $38.75Mbit/s$ to $0.26Mbit/s$. This shows the significant effect that packet loss has on the TCP throughput, since even 1% loss results to almost half the throughput. This is due to the TCP backoff mechanism which exponentially increases the TCP retransmission timeout after each consecutive drop (it can reach 64 seconds in high loss rates).

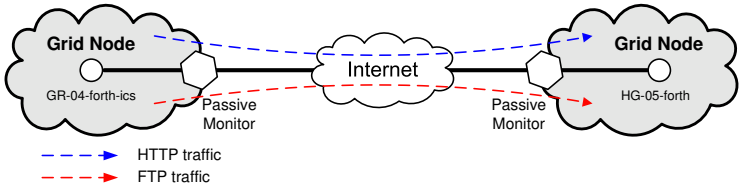

Figure 4. Packet loss measurements in a Grid network path

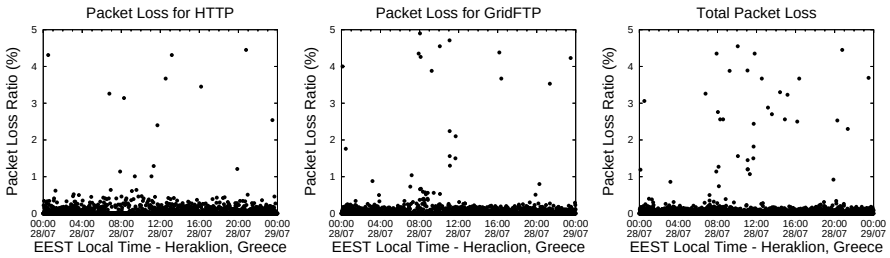

Figure 5. Packet loss variation in a daily period for HTTP and GridFTP protocols.

4.2 Experiences with Grid Network Traffic

We deployed our technique in an operational Grid network path between two different sites, as depicted in Figure 4. We installed the packet loss components in two sensors that monitor the traffic of *HG-05-forth* and *GR-04-forth-ics* Grid sites, which are parts of the EGEE (*Enabling Grids for E-sciencE*) infrastructure. The monitoring application was running on a host inside FORTH for 24 hours and reported the current loss ratio in both directions every 30 seconds.

In Figure 5 we plot the time series of the packet loss ratio for two popular GRID protocols, HTTP and GridFTP, and the total packet loss ratio of the link, for 30 seconds intervals. In order to create more traffic between these sites, we initiated a random number of parallel HTTP and FTP transfers (varying between 1 and 10) for large files ($20MB$ and $30MB$ respectively) every 2 minutes.

The results show that bursts of HTTP and FTP transfers result to higher packet loss rates, as expected. Besides our generated traffic, which mainly dominated the network path and caused the increased packet loss, we also notice some slight diurnal patterns in the loss rates due to the real Grid application traffic of the path. For instance, we can see higher packet loss ratios during the morning and afternoon hours (08:00 to 16:00 local time), while constant and low packet loss occurs in late hours at night, due to our generated traffic only.

The percentage of the 30-second intervals with zero packet loss for HTTP, FTP, and the total traffic, was 89%, 87%, and 83%, respectively. The overall loss ratio for the 24-hour period was 0.09% for the total traffic, 0.13% for

HTTP-only traffic, and 0.19% for FTP-only traffic. The loss ratio for the total traffic is lower than HTTP-only and FTP-only traffic, because packet loss was occurred mainly in our generated HTTP and FTP traffic. The two monitored services resulted to similar loss rates. This is because we generated comparable HTTP and FTP traffic during the same time periods.

5. Related Work

Previous work on packet loss estimation can be broadly categorized into approaches based on passive and active network monitoring, with the latter having a significantly larger literature body.

One of the most popular tools for inferring the basic network characteristics, such as round-trip time and packet loss, is `ping`. `Ping` uses the ICMP protocol to send probe packets to a target host at fixed intervals, and reports loss when the response packets are not received within a specified time period. However, ICMP packets are often rate limited, or blocked, by routers and firewalls. An other active tool is `zing` [1], which estimates the end-to-end packet loss in one direction between two cooperative end hosts, by sending UDP packets at Poisson modulated intervals with a fixed mean rate. `Badabing` [16]also measures the one-way packet loss by sending fixed-size packets at specific intervals. `Sting` is an active monitoring tool that measures the loss rate in both forward and reverse directions from a single host to any TCP-based server, by exploiting TCP's loss recovery algorithms [15]. Finally, network tomography using unicast probes has been used for inferring loss rates on end-to-end paths [7].

Besides active tools, there also exist methods that use passive network monitoring for measuring the TCP packet loss, based on the TCP retransmission mechanism [3]. However, there are several applications, such as `tftp`, which use UDP instead of TCP. Techniques for estimating the loss rate based on the TCP protocol are also presented in [2], however they work only in individual clients and they cannot be used by other external applications, e.g., for improving routing or selecting a replicated server with the best network conditions.

6. Conclusion

We presented a novel distributed passive monitoring technique for real time packet loss estimation between different Grid domains. The technique is based on tracking the *expired flows* at each monitoring sensor. Using a distributing monitoring infrastructure, a central monitoring application correlates the results and computes the actual packet loss ratio. Our experimental evaluation using controlled packet loss shows that our approach is accurate and reliable, while at the same time exhibits inherent advantages such as scalability and a non-intrusive nature. Finally, preliminary results from a network path with real Grid traffic are promising, and demonstrate the applicability of our approach.

References

[1] Andrew Adamns, Jamshid Mahdavi, Matthew Mathis, and Vern Paxson. Creating a scalable architecture for internet measurement. *IEEE Network*, 1998.

[2] Mark Allman, Wesley M. Eddy, and Shawn Ostermann. Estimating loss rates with tcp. *ACM Performance Evaluation Review*, 31(3), December 2003.

[3] Peter Benko and Andreas Veres. A passive method for estimating end-to-end tcp packet loss. In *Proceedings of IEEE Globecom*, 2002.

[4] Augusto Ciuffoletti and Michalis Polychronakis. Architecture of a network monitoring element. Technical Report TR-0033, CoreGRID Project, February 2006.

[5] Chuck Cranor, Theodore Johnson, Oliver Spataschek, and Vladislav Shkapenyuk. Gigascope: a stream database for network applications. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 647–651, 2003.

[6] Nick G. Duffield and Matthias Grossglauser. Trajectory Sampling for Direct Traffic Observation. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 271–282, 2000.

[7] Nick G. Duffield, Francesco Lo Presti, Vern Paxson, and Donald F. Towsley. Inferring link loss using striped unicast probes. In *INFOCOM*, pages 915–923, 2001.

[8] Apache Software Foundation. Apache http server benchmarking tool. `http://www.apache.org/`.

[9] C. Fraleigh, C. Diot, B. Lyles, S. Moon, P. Owezarski, D. Papagiannaki, and F. Tobagi. Design and Deployment of a Passive Monitoring Infrastructure. In *Proceedings of the Passive and Active Measurement Workshop*, April 2001.

[10] Stephen Hemminger. Netem: emulating real networks in the lab. In *Proc. Linux Conference Australia*, April 2005.

[11] Gianluca Iannaccone, Christophe Diot, Derek McAuley, Andrew Moore, Ian Pratt, and Luigi Rizzo. The CoMo White Paper, 2004. `http://como.intel-research.net/pubs/como.whitepaper.pdf`.

[12] Hao Jiang and Constantinos Dovrolis. Passive estimation of tcp round-trip times. *SIGCOMM Comput. Commun. Rev.*, 32(3):75–88, 2002.

[13] Laor Michael and Gendel Lior. The effect of packet reordering in a backbone link on application throughput. *Network, IEEE*, 16(5):28–36, 2002.

[14] Michalis Polychronakis, Kostas G. Anagnostakis, Evangelos P. Markatos, and Arne Øslebø. Design of an application programming interface for IP network monitoring. In *Proceedings of the 9th IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 483–496, April 2004.

[15] Stefan Savage. Sting: A tcp-based network measurement tool. In *USENIX Symposium on Internet Technologies and Systems (USITS)*, 1999.

[16] Joel Sommers, Paul Barford, Nick Duffield, and Amos Ron. Improving accuracy in end-to-end packet loss measurement. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, pages 157–168, 2005.

[17] Panos Trimintzios, Michalis Polychronakis, Antonis Papadogiannakis, Michalis Foukarakis, Evangelos P. Markatos, and Arne Øslebø. DiMAPI: An application programming interface for distributed network monitoring. In *Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 2006.