BPB: A method for transparently obtaining network path characteristics close to the sender

Sven Hessler
University of Innsbruck
sven.hessler@uibk.ac.at

Jean-Alexander Müller
Dresden University of Applied Sciences
jeanm@htw-dresden.de

Michael Welzl
University of Innsbruck
michael.welzl@uibk.ac.at

Abstract

Due to the growth of unresponsive UDP traffic in the Internet, it becomes increasingly important for ISPs to amply shape the traffic that leaves their network. Ideally, flows should be forced to be TCP-friendly; to this end, knowledge about certain end-to-end path characteristics is needed. We present a suitable mechanism (the Burst-PiggyBack (BPB) technique) that obtains the necessary information at a device that is located close to the sender without requiring any changes at the communicating peers or in routers. The method's hypothesis is that an injected probe packet at the end of a burst is treated similar to the burst. The results acquired from simulations showed strong correlations between the loss rate and the RTT of bursts and probe packets.

1. Introduction

The increasing amount of unresponsive UDP traffic in the Internet (e.g. the growing number of VoIP flows) gives rise to concerns about the potential danger of a future congestion collapse [6]. If this trend does not change, there will be increasing pressure for ISPs to install traffic shapers which limit the magnitude of UDP aggregates that leaves their network. Then, the question of determining the maximum per-flow rate that can be allowed arises.

Ideally, an end-to-end data flow should be *TCP-friendly*, i.e. not send more than a standard TCP under comparable conditions [4]. A TCP sender updates its rate based on packet loss information and an RTT estimate, both are determined via feedback from the receiver — thus, for a traffic shaper to calculate the theoretical rate of a standard TCP under similar conditions, these network properties must be determined.

Without detailed knowledge about the application in question, it may however not be feasible to carry out strictly passive measurements, as the receiver may not always generate enough feedback that can be monitored.

In this paper, we present a first step towards such a traffic shaper: a novel method for measuring the RTT and estimating loss within the network (close to the sender side) without help from the receiver. Roughly, our approach is to inject ping-like packets into the network. These packets cause a reaction from the receiver without knowing about our mechanism. While TCP decisions are based on per-packet feedback, having our mechanism generate a probe packet for each payload packet would obviously be too much overhead. The existence of schemes such as TFRC [5] and TEAR [15], which were shown by their authors to attain TCP-friendly behavior with only one feedback packet per RTT (for TFRC, this is specified in RFC 3448 [7]) leads us to believe that such a small amount of signaling may be enough. Our method could be integrated in these schemes as well as RTCP-based TCP-friendly adaptation mechanisms such as LDA+ [18].

The remainder of this paper is organized as follows: Section 2 presents a survey of related research regarding the measurement of network parameters. Our BPB technique and a detailed description of the validation environment, including the architecture and the setup of the simulation is described in Section 3. In Section 4 we present the results of our measurements by ns-2 simulations and a proposal for real-life tests. Section 5 concludes and outlines future directions of our research.

2. Background and related work

The measurement of network metrics of an Internet end-to-end path is an important research area. Knowledge of performance information, such as delay, loss, and bandwidth, can be used in several ways: ISPs can keep track of and adjust their links, media servers can adapt their sending rate [20], and overlay networks can be created more robustly [19].

Previous work on estimation of network properties has focused on measurement of link capacity [3, 1, 9], loss [17], and discovery of network properties like queuing behavior of routers [10]. Most of that research is based on the

packet-pair technique, i.e. sending two packets one after the other within a very short time whereby the difference of the received packets could be used to calculate the bottleneck bandwidth. The reliability of this method is underlined by the following reasoning: If a pair of packets is sent across a network, it is highly likely that the second packet will be received if the first packet reached the sink. This thesis was validated experimentally [13], and theoretically for a $M/M/1/K$ queue model [2]. The idea of using pairs of packets can be applied passively, or actively by injection of probe packets. However, Saroiu et al. [16] argue that passive tools might suffer under realistic scenarios from traffic which is not suitable for measurement. Our approach is based on these results; it actively injects a probe packet right after a burst of data which virtually builds a pair of packets (or a series thereof).

In addition to estimating loss, our method measures the round-trip time (RTT), which is an indicator for network utilization, and can be used to calculate the sender rate as a reference to enforce TCP-friendliness. The most well-known tool to measure the RTT along a network path is *ping*, which calculates the RTT from the timestamps of an ICMP Echo-request and the related ICMP Echo-response. Besides this classical method, the SYN/ACK mechanism of the TCP [14] handshake protocol is employed by *ping*-like utilities, such as Sting [17]. While in our approach both methods can be applied to calculate the RTT, it mainly uses a combination of them – it sends a TCP or a UDP packet as request, and receives an ICMP message as response.

3. The Burst-PiggyBack (BPB) approach

The classical way to measure RTT is, to send an ICMP Echo-request packet, receive the corresponding ICMP Echo-reply packet, and calculate the difference of the timestamps. It is likely that a single ICMP packet will be treated differently in comparison to other flows on the same path if their behavior is not related to each other. We propose to append probe packets that will force the receiver to generate a response packet to bursts of non-responsive flows, e.g. by sending TCP or UDP connection requests to a blocked port. The difference between the timestamp of the probe-packet and the generated response packet can be used to estimate the RTT of the unresponsive flow (Figure 1), and the monitored loss events can be used to estimate the loss of the data flow, respectively. Finally, our hypothesis is, that the transmission delay and loss probability of the probe packets is similar to, and particularly not lower than, that which a packet within a burst of data will experience. If this hypothesis is correct, the loss behavior and the round-trip time of the data stream can be derived from that of the probe packets.

Obviously, a few issues must be considered when using bursts of packets instead of the packet-pair technique. The method of using packet-pairs is based on the assumption that both packets receive the same treatment in the network due to their close arrival times. But the time difference of the two departure times (inter-packet-delay) at the source is important too. Due to the statistical multiplexing, accuracy in measuring the current network behavior decreases as the inter-packet-delay at the source increases. Therefore, in order to infer from one probe packet per burst the RTT and loss probability of the packets in the preceding burst, the departure times between the first packet and the last packet of the burst (referenced as burst-time) has to be short – a fraction of the RTT.

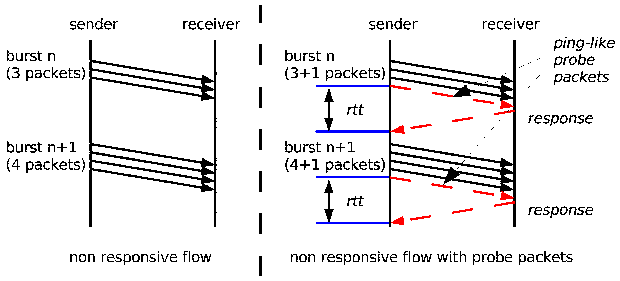

Figure 1. Probe packets are injected after bursts (right) of a non-responsive flow (left)

We chose to send TCP probe packets (TCP-Syn) that were addressed to a likely blocked port number instead of sending regular ICMP messages because manual pre-tests showed that some hosts which ignored ICMP echo packets responded with an ICMP error message to our TCP-Syn probes. Since we regard our work as a feasibility study, we decided that this method would suffice. Applications that use our approach, however, may use ICMP echo packers if applicable or should use more reliable ways to detect the appropriate probe packet type – for instance, a sender can try a series of different packet types (UDP, TCP, ICMP) until a response is received. If the recipient in question is a web server, a mechanism like Sting [17] can be used.

4. BPB evaluation and results

In this section, we first present the test setup we used to validate the RTT and packet loss estimations of our approach by intensively performing simulations using the network simulator ns-2 [11], show the results of these experiments, and finally propose a methodology to proof the feasibility of our approach in real-life tests.

4.1. Simulation test setup

The simulations consisted of a 6-node path as shown in Figure 2, i.e. link bandwidths = 10, 20 Mbps, and propagation delays = 1, 10 ms. The bottleneck link between nodes 2 and 3 is shared by all the data flows.

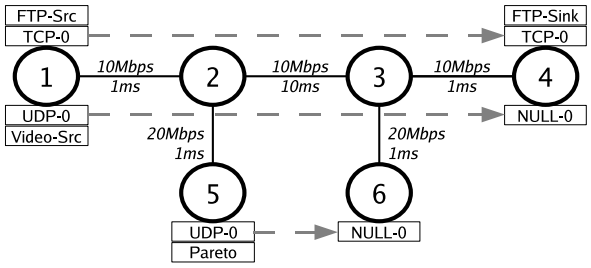

Figure 2. Simulation scenario

We examined two types of – non-concurrent – application flows, first a congestion-aware, continuous FTP/TCP flow, and then a non-responsive video stream using UDP. Further, one non-responsive cross-traffic flow of one or more Pareto sources was injected to the network. All of these data flows share the bottleneck-link (nodes 2–3). See further Table 1 for all parameters used in the simulations.

We chose the two types of applications flows, TCP-based FTP and UDP, due to the following reasoning: the FTP stream was selected to compare the RTT and the loss which was obtained by our method with the same values of TCP. In the second part of the experiments a constant bitrate UDP flow sending bursts of data was used to emulate the behavior of streaming MPEG videos encoded with a constant bitrate. The cross-traffic models were web traffic based on a set of Pareto sources with different shapes and rates; cross-traffic was used to cause high utilization of the bottleneck link until congestion occurred.

Name	App	Proto	Start	Stop	Src	Dest
FTP	FTP	TCP	1.0s	40.0s	1	4
Video non-responsive	cbr	UDP	1.0s	40.0s	1	4
	2 Mbps avg. rate, 10 Mbps peak rate, burst of 5 . . . 10 packets					
Cross-Traffic	Pareto	UDP	10.0 s	40.0 s	5	6
	rate == [0.8 – 1.0] * rate of bottleneck link shape 1.5 . . . 1.9, bursts of 1 . . . 10 packets					
	Size of all packets: 1000 bytes					

Table 1. Types of traffic used in simulations

For the simulations we extended the network simulator ns-2 because it did not provide the required methods to a) discover bursts and b) to append a probe packet to the last packet of each burst (both to the same destination). We calculated the difference δ between the dequeue-times of the current packet and the last packet. A timer was scheduled to inject the probe packet after $\delta + 0.001$ milliseconds, if a packet was dequeued from node 1 and $\delta < 10$ ms. A difference higher than 10 ms indicated two independent packets, because the RTT was at least 24 ms (c.f. Figure 1). Our burst detection was therefore not entirely strict, i.e. it was also decided that packets would belong to a burst if there was a gap of up to 10 ms between them.

4.2. Results

In the following we present our results obtained by simulations to validate the applicability of BPB, i.e. its ability to estimate the RTT and the likeliness of packet loss.

4.2.1 Experiments with FTP/TCP

The first set of experiments evaluates the applicability of BPB by comparing the measured RTT and loss with that of an observed FTP/TCP flow. We selected three representative runs (S1, S2, S3; with the initial values of 17476, 24473 and 28704 for the random-seed) from the set of simulations and discuss them further in detail.

The measured RTTs of the selected runs are depicted in Figure 3, 4 and 5. We classified the likeliness of packet loss derived from the loss of the piggybacked probe packet shown in the figures as follows:

3: the probe packet and all or some packets in the burst, where the probe packet was attached to, were dropped

2: the probe packet was dropped and all or some packets in the following burst

1: only the probe packet was dropped

0: no loss (neither of the probe packet nor of the burst)

-1: all or some packets in the burst were dropped, but not the probe packet

A fundamental part of our hypothesis is: it is unlikely that the probe packet is served but some or all packets in the burst are dropped. Our simulations support this, as we never observed a contradictory case, no matter what application flow was used. Even as a very large set of simulations could not be taken as unimpeachable evidence, it strongly indicates applicability to networks with drop-tail queuing.

Another part of our hypothesis is, that the loss of a probe packet correlates with the loss of the packets where that probe was piggybacked to, and consequently indicates network congestion. The results regarding the loss of probe packets and the loss of burst packets are shown in Table 2. As support of our hypothesis, the runs S1 and S3 show that

case	-1	0	1	2	3	loss ratio (TCP)
run S1	0	1923	7	4	72	0.279
run S2	0	1015	51	20	61	0.188
run S3	0	1837	6	3	67	0.286

Table 2. No. of events that match a loss case

more than 86% of the probe packets were dropped when packets of the burst – it was attached to – were dropped.

For run S2, only 50% of the lost probe packets indicated loss of packets it was attached to. Contrary to the the runs S1 and S3, run S2 experienced a heavy congestion period between simulation time $10-30\,\mathrm{s}$ (c.f. Figure 5). In reaction to this congestion and the resulting high RTT estimation of TCP, the number of packets per burst and the average rate were reduced, and some of the bursts were served without loss. Most of the probe packets were lost, as there was space for the TCP packets as estimated by the TCP algorithm, but no additional space for a probe packet. This does not contradict our hypothesis, because the loss of probe packets indicated network congestion very well. In comparison to run S1 and S3 the number of lost probe packets per observation period is much higher.

Finally, we can conclude, that a) the loss of probe packets attached to bursts could indicate network congestion – and it is likely that packets of the burst are dropped too – and b) a high loss rate of probe packets indicates high network congestion. On this basis, a sender can react to congestion and reduce the peak rate and the number of packets per burst, thereby lowering the load of the network.

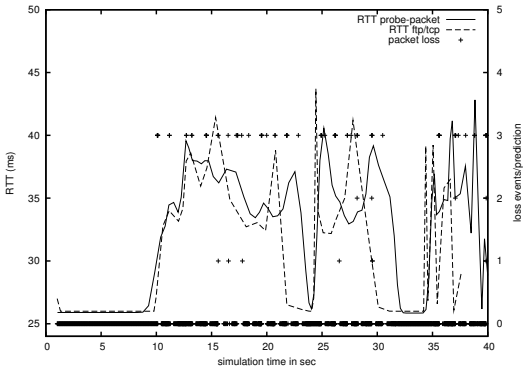

Figure 3. Smooth bezier curves of RTTs and loss events (3 = loss of FTP and ping packets, 2 = loss of ping packets and correct prediction of loss of packets of the next burst, 1 = loss of ping packets, 0 = no loss) [run S1]

While packet loss is a good evidence for network congestion, it is a coarse grained mean to estimate the currently available bandwidth and its variation. Our hypothesis was, that the transmission delay of the probe packets is similar to, and particularly not lower than, that which a packet within a burst of data will experience. Supporting our hypothesis, Figure 3, 4 and 5 show that the RTT estimated by the TCP-algorithm[1] is close to the RTT measured using our BPB approach, if no or very little loss occurs.

A few exceptions exist because the TCP algorithm is designed to yield a very conservative estimate in the face of high packet loss rates. Thus, when the network was heavily congested and high loss rates occurred, e.g. as in run S2 (Figure 5), the RTT estimated by TCP was up to two times higher than the RTT measured with the BPB method. In the case of moderate congestion, e.g. as seen in run S1 and S3, the difference between the TCP-estimated RTT and the BPB-measured RTT is a) sometimes similar, but shorter in

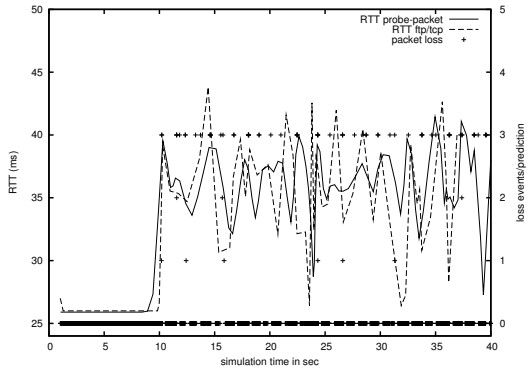

Figure 4. Smooth bezier curves of RTTs and loss events (see Figure 3) [run S3]

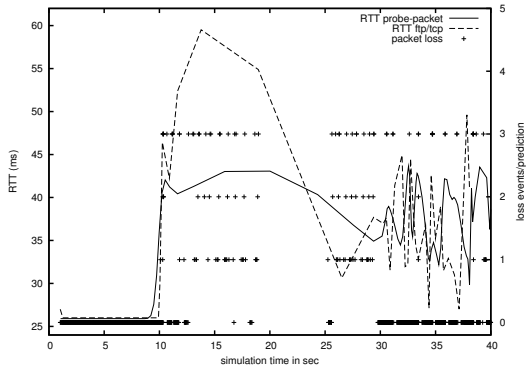

Figure 5. Smooth bezier curves of RTTs and loss events (see Figure 3) [run S2]

[1] tcptrace [12] was used to estimate the RTTs of TCP.

time, and b) visually time shifted compared to other curve. For further discussion we calculated the statistical means, such as arithmetic mean, standard deviation (sdv), and the interquartile mean (iqm) for the difference of the RTT estimated by TCP and the RTT measured by the BPB technique (c.f. Table 3). These values can be used to quantify our method, i.e. if the difference is very low it will indicate that both methods to obtain the RTT (TCP's estimation and BPB's measurement) have the same quality.

case	mean	sdv	iqm
run S1	-0.95	12.66	-0.14
run S2	-0.53	12.96	0.02
run S3	-0.42	13.0	-0.16

Table 3. Difference between BPB-measured and TCP-estimated RTT in ms

Given that the bottleneck link can serve one packet (1000 bytes) in $0.8\,\mathrm{ms}$, the arithmetic mean shows that a number of packets, smaller than 1250 bytes in total were multiplexed between the burst and the attached probe packet.

Taking the iqm values as basis, the accuracy of the BPB technique looks even better. The relatively high values for the standard deviation (up to $13\,\mathrm{ms}$, around 16 packets of 1000 bytes) implies that at most 16 packets of cross-traffic were multiplexed to the queue between the arrival of the last packet in the burst and the arrival of the attached probe packet. We have found some rare events with an inter-packet delay of $1-6\,\mathrm{ms}$ for "bursts" of two packets because the rate adaption of TCP in case of heavy congestion. Since the "burst detection algorithm" was limited to an upper bound of $10\,\mathrm{ms}$ for the inter-packet delay, the probe packets again were sent after such an interval and eventually delayed due to cross traffic, causing such high RTTs.

4.2.2 Experiments with non-responsive flows

A further set of experiments with unresponsive data flows using UDP was performed. From these experiments we chose one sample to give a – due to limited space – short discussion on the applicability of our approach, for a scenario that our proposed technique is intended for. The source of the observed flow was configured to send out bursts of 10 packets (1000 bytes each) with a peak rate of $10\,\mathrm{Mbps}$ and an average rate of $2\,\mathrm{Mbps}$ which resembles a video source sending 25 frames per second. A network load of up to 100% was caused by cross-traffic from the Pareto source with packets of 1000 bytes and a shape of 1.9.

Figure 6 depicts the loss events and the smoothed bezier curves of the RTT, and the one way delay (OWD) measured in the upstream path (probe packets and packets of the video

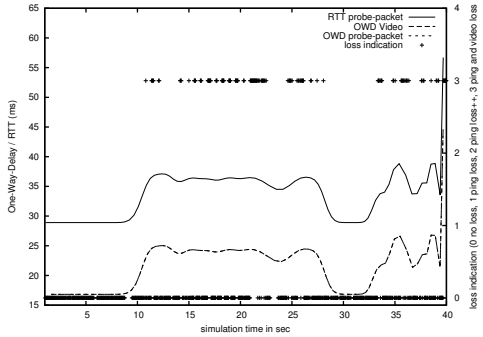

Figure 6. Smooth bezier curves of RTTs and loss events (see Figure 3) [run S1p2]

stream). The curves for the OWD are very close to each other and the RTT has the same characteristics, except for an offset of $12\,\mathrm{ms}$ (caused by the transmission delay). Further, the loss of probe packets indicated the loss of some or all packets in the burst correctly. In general, we found this behavior in all of our simulations with non-responsive flows. According to the acquired results, our hypothesis proves to be true.

The results obtained in the UDP-based experiments are even more significant than those derived from the simulations using TCP. This can be shown as follows: contrary to the FTP/TCP flow where the inter-packet delay varied highly ($0.8-6\,\mathrm{ms}$ instancing the runs with $10\,\mathrm{Mbps}$ bottleneck), the inter-packet delay of the video stream was rather low, and constant ($0.4\,\mathrm{ms}$ for the runs with $10\,\mathrm{Mbps}$ peak rate), i.e. only bursts of 10 packets, sent back-to-back, were seen. Accordingly, the probe packets were attached back-to-back to the burst and were not delayed to meet the rate estimated by TCP.

Nevertheless, as we have shown in the TCP-based experiments, the BPB technique gives a good estimation of the RTT and loss even if the packets are not sent back-to-back, but with a higher inter-packet delay caused by shaping. The efficiency of the burst-detection algorithm depends on the chosen inter-packet delay, which has to be set according to the available bandwidth, estimated by the measured RTT, and the size of a burst.

4.3. Proposal for a real-life test setup

In the next step real-life experiments can be used to validate the promising results shown by simulations. In the following we present our concepts of how to conduct such tests.

Our approach of bandwidth and packet loss estimation is – as motivated before – primarily aimed at UDP based data flows. However, as routers inherently treat bursts equally,

no matter what layer-3 protocol the application uses, the validation of real-life tests can benefit from the advantages of TCP, namely to have a secure and feedback-based transmission of packets. By using TCP-based applications and tcptrace [12], we can easily compare the values of the RTT and loss obtained with our BPB technique with the corresponding values of TCP. Deducing general conclusions from measurements is always a critical issue which should impact the test setup. Thus, we propose either to make use of applications like BitTorrent, where the clients of peer-to-peer networks are inherently geographically well distributed or to deploy a ftp mirror service, e.g. for a Linux derivate with a similar intention of wide-spread measurement points.

For the measurement we propose to use the Linux packet filtering firewall "iptables" [8] and extend it with two modules. The first detects a packet burst and injects a TCP-Syn or UDP probe packet at the end of it, and the second module calculates the time difference between the probe packet and the related response. A response is regarded as "related" if it is either an ICMP message of type 3 (destination unreachable), or a TCP packet where the ACK, and the SYN or RST flag is set. The burst-detection algorithm could be similar to that method successfully applied in the simulations, i.e. not keeping track of the queue-length, but defining a burst as a certain amount of packets where the inter-arrival time for these packets does not exceed a specific value, which depends on the nominal link capacity.

5. Conclusion and future work

In this paper we presented the Burst-PiggyBack (BPB) technique – a novel approach to obtain end-to-end path characteristics, such as loss and round-trip time (RTT) in uncooperative environments. It can be used, for instance, by a multimedia sender to estimate the network's load and adjust its sending rate accordingly. We confirmed the method's hypothesis – an injected probe packet at the end of a burst of data is treated similar to the burst, i.e. it has a similar loss rate and RTT as the packets of the burst – by simulations. Our results showed strong correlations between the loss rate and the RTT of the probe packets and the packets which are part of the corresponding bursts under several network conditions for TCP- and UDP-based flows in networks with drop-tail queues. We are currently studying the feasibility of the BPB approach in real-life tests as outlined in this paper. Additionally, we will investigate methods of efficient burst detection to be used with BPB technique.

6 Acknowledgments

This work was partly funded by the Austrian Science Fund (FWF).

References

[1] L.-J. Chen, T. Sun, D. Xu, M. Y. Sanadidi, and M. Gerla. Access link capacity monitoring with tfrc probe. In *Proceedings of the 2nd Workshop on End-to-End Monitoring Techniques and Services (E2EMON)*, October 2004.

[2] M. Coates and R. Nowak. Network inference from passive unicast measurement. Technical report, Rice University, 2000.

[3] C. Dovrolis, P. Ramanathan, and D. Moore. Packet dispersion techniques and capacity estimation. *IEEE/ACM Transactions on Networking*, December 2004.

[4] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networks*, 7(4):458–472, 1999.

[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proceedings of ACM SIGCOMM*, 2000.

[6] S. Floyd and J. Kempf. IAB Concerns Regarding Congestion Control for Voice Traffic in the Internet. RFC 3714, Mar. 2004.

[7] M. Handley, S. Floyd, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 3448, Jan. 2003.

[8] The netfilter/iptables project. `http://www.netfilter.org`.

[9] R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, and M. Y. Sanadidi. Capprobe: A simple and accurate capacity estimation technique. In *Proceedings of ACM SIGCOMM*, 2004.

[10] J. Liu and M. Crovella. Using loss pairs to discover network properties. In *Proceedings of Internet Measurement Workshop*, 2001.

[11] The network simulator ns-2. `http://www.isi.edu/nsnam/ns/`.

[12] S. Ostermann. tcptrace. `http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html`.

[13] V. Paxson. End-to-end internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7:277–292, 1999.

[14] J. Postel. Transmission Control Protocol. RFC 793 (Standard), Sept. 1981. Updated by RFC 3168.

[15] I. Rhee, V. Ozdemir, and Y. Yi. Tear: Tcp emulation at receivers - flow control for multimedia streaming. Technical report, NCSU Department of Computer Science, April 2000.

[16] S. Saroiu, P. Gummadi, and S. D. Gribble. A fast technique for measuring bottleneck bandwidth in uncooperative environments. In *Proceeding of IEEE Infocom*, 2002.

[17] S. Savage. Sting: a tcp-based network measurement tool. In *USENIX Symposium on Internet Technologies and Systems*, pages 71–79, October 1999.

[18] D. Sisalem and A. Wolisz. Lda+: A tcp-friendly adaptation scheme for multimedia communication. In *ICME*, 2000.

[19] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, 2001.

[20] X. Wang and H. Schulzrinne. Comparison of adaptive internet multimedia applications. *IEICE Transactions on Communications*, E82-B(6):806–818, June 1999.