THE UNIVERSITY OF AUCKLAND
LIBRARIES AND LEARNING SERVICES

ResearchSpace@Auckland

Version
This is the Accepted Manuscript version. This version is defined in the NISO recommended practice RP-8-2008 http://www.niso.org/publications/rp/

Suggested Reference

Cocker, E., Ghazzi, F., Speidel, U. M., Dong, M.-C., Wong, V., Vinck, A., . . . Eimann, R. (2014). Measurement of buffer requirement trends for real time traffic over TCP. In *2014 IEEE 15th International Conference on High Performance Switching and Routing (HPSR)*. Vancouver. doi:10.1109/HPSR.2014.6900891

Copyright

Items in ResearchSpace are protected by copyright, with all rights reserved, unless otherwise indicated. Previously published items are made available in accordance with the copyright policy of the publisher.

© 2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

http://www.ieee.org/publications_standards/publications/rights/rights_policies.html

https://researchspace.auckland.ac.nz/docs/uoa-docs/rights.htm

Measurement of Buffer Requirement Trends for Real Time Traffic Over TCP

'E. Cocker, F. Ghazzi, U. Speidel
Department of Computer Science, The University of Auckland
Email: {ecoc005,fgha004}@aucklanduni.ac.nz, ulrich@cs.auckland.ac.nz

M.-C. Dong[1], V. Wong[1], A. J. Han Vinck[2], H. Yamamoto[3], H. Yokoo[4], H. Morita[5], H. Ferreira[6], A. Emleh[6], R. McFadzien[7], S. Palelei[8], R. Eimann[9]
[1] University of Macau, [2] University of Duisburg-Essen, [3] University of Tokyo, [4] Gunma University, [5] University of Electro-Communication, [6] University of Johannesburg, [7] Telecom Cook Islands, [8] Ministry of Revenue - Tonga, [9] in private capacity

***Abstract*—Conceptionally, the User Datagram Protocol (UDP) should be well-suited for real-time applications, e.g., for Voice over IP (VoIP). However, many such applications, e.g., Skype, use the Transmission Control Protocol (TCP) either as a primary protocol or as a backup protocol when UDP is blocked, despite TCP's flow control-related data delays. This paper proposes a technique for the estimation of the application buffer requirements of such TCP-based applications and the amount of data congestion in real-time TCP data streams. We apply this technique to data collected from a global network exchanging synthetic real-time traffic over TCP. Our results show that the buffering requirements vary widely with time and path but can be substantial in many cases.**

I. Introduction

Previous studies of Internet traffic distributions indicate that most Internet traffic uses TCP [1]. This is supported by the fact that Video on Demand (VoD) applications such as Youtube and MSN video use TCP for transmission of video packets over HTTP [2]. In addition, real-time applications, e.g Skype, evade network firewalls with TCP, e.g., in Skype's case using the commonly open ports 80 (HTTP) and 443 (HTTPS) for communication with Skype peers [3]. Other real-time applications such as Google+ also use TCP in networks where UDP is blocked [4].

Originally, TCP was developed for use over best effort protocols such as IP. It employs flow and congestion control mechanisms to exchange data reliably between end hosts. Protocols that commonly use TCP include HTTP [5], FTP [6] and SMTP [7]. For these protocols, the arrival order and completeness of data have priority over the need to exchange information fast. With increased use of TCP for the transmission of small chunks of voice data in VoIP applications, any delay introduced by TCP flow control becomes a potential concern. The same applies to other real-time applications. In each case, the real-time application involved must provide a FIFO-style replay buffer to smooth the data stream for replay. If this replay buffer is too small, there will be breaks in replay; a buffer that is too large means unnecessary delays in replay. In the case of VoIP, excessive replay delays prohibit meaningful conversation [8]. As the replay buffer operates at the application layer, any replay delay incurred there occurs *in addition* to the time that the data spends in the TCP socket's receive buffer.

Rather than investigating actual replay buffer sizes, this paper considers the minimal size/delay of a hypothetical replay buffer that would have guaranteed smooth voice replay, and proposes to use this measure as a quality indicator to observe quality trends over time along other indicators such as latency, jitter, or Mean Opinion Score (MOS) [9]. This replay buffer size is a function of the sizes and handover times of TCP buffers passed from the TCP stack to the receiving application, which we collect using synthetic VoIP-like traffic on an international network of measurement computers ("beacons"), which is described in [10].

This paper is organised as follows: Section II gives a brief overview of existing active measurement tools that use TCP for probing networks with artificial packets. In Section III, we describe the process by which we compute the minimum replay buffer requirements. Lastly, in Section IV we present the results of our experiments to date.

II. TCP performance measurement

Most present active measurement tools developed with TCP functionalities tend to focus on the estimation of link capacity and available bandwidth for diagnostic purposes. Two commonly employed techniques for estimation of available bandwidth in this context are Self-Loading Periodic Streams (SLoPS) [11] and Trains of Packet Pairs (TOPP) [12]. Variations of the SLoPS technique are used by tools such as pathchar [13], pchar [14], clink [15], pathload [16] and iperf [17]. SLoPS estimates bandwidth through self-induced congestion by increasing a packet stream's rate to a point where it is higher than the available bandwidth, either by increasing the packet rate or the packet size. TOPP estimates bandwidth by probing the network with well separated equally sized packets [18] and comparing their intertransmission times with the corresponding interarrival times (packet dispersion).

TOPP has been employed by utilities such as nettimer [19] and pathrate [20] (which also uses SLoPS).

Available bandwidth is a cumulative observable, however, and does not reflect transient ad-hoc congestion. Even if available bandwidth is high across the duration of an experiment, transient ad-hoc congestion may still occur, causing time-critical data in real-time applications to arrive late at the receiver. Our project aims to track how serious this problem is by observing over a longer timeframe as the global Internet evolves. The next section describes how we arrive at our chosen observable, the minimum buffering requirement at the receiver.

III. Minimum buffering requirements in real-time TCP applications

In order to supply a receiver with sufficient data for playback, a real-time application must transmit data at the same rate as the receiver consumes it. In practice, this takes the form of the transmitting application passing chunks of data to the TCP/IP stack at regular intervals. Typical values observed for, e.g., VoIP applications such as Skype, are in the range of about 115 bytes of data every 20 ms.

As we cannot rely on each data chunk experiencing a fixed delay across a TCP connection on its way to the receiving application, the receiving application must buffer some received data to ensure (hopefully) continuous replay. The buffer is a FIFO queue; its purpose is to ensure that the cumulative number of bytes delivered to it by TCP always exceeds the cumulative number of bytes consumed by the application for replay. If the buffer cannot meet this requirement, the application experiences a *buffer underrun* – a break in replay.

As the buffer queue represents an undesirable additional delay in a real-time application, it should be kept as small as possible. A larger buffer generally means better continuity of replay, but also more delay as the buffer queue must be filled first before one can take data for replay off the front of the queue. The size of this buffer is thus more or less proportional to the delay it introduces.

Our experiment tries to estimate the minimum buffer size that guarantees continuity of replay (in one direction) for an actual 200 second simulated Voice over TCP data flow. We collect the data from our beacon network, where selected pairs of beacons run such simulated Voice over TCP data transmissions three times a day. For each of these transmissions, we estimate buffer size and then track how this estimate evolves over time for each beacon pair.

The receiving beacon records both the times at which chunks of data become available to the receiving application, along with the amount of data passed to the application by the socket at each of these events.

Figure 1 illustrates the conceptual relationship between buffer size and buffer underruns.

The horizontal axis of Figure 1 covers the time period over which our real-time application operates, e.g., the duration of a VoIP phone call. On the vertical axis, we plot the number of bytes. The red "uneven staircase" graph represents the

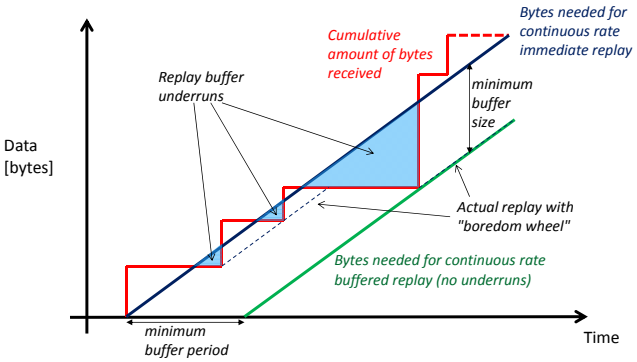

Fig. 1. Red: data available at receiver, dark blue: data demand for immediate playback, green: data demand for buffered playback.

cumulative amount of bytes passed to the application by the TCP socket up to the given point in time. Each step represents another chunk of data becoming available. The linear solid dark blue graph intersecting with the red graph represents the estimated amount of data required for continuous replay *without delay*, i.e., when replay starts immediately upon first delivery of voice data. Buffer underruns (blue shaded areas) thus occur whenever the red graph lies below the blue continuous replay graph. The dashed graph plots data use if one resumes full replay with increasing delay after the resulting discontinuities in replay.

Finally, the green graph in Figure 1 represents a second version of the continuous replay graph, with replay delayed by the minimum amount of time required to avoid buffer underruns. This time shift thus indicates the minimum replay delay required to achieve uninterrupted, continuous playback, and hence the minimum buffer time required. Correspondingly, the vertical distance between the blue and green graph represents the minimum number of bytes the buffer needs to hold. These two quantities are of course linearly related via the playback data rate and represent the estimate we require.

One issue to consider in this context is the difference in clock speed between transmitter and receiver. Let $t'(t)$ be the time at the transmitter when the receiver's clock has value t after start of playback. Suppose that we configured a fixed nominal data rate r (in bits per second) at both sides for transmission and playback. The receiver would then need a total of rt bits after t seconds of playback. If the receiver's clock runs faster than that of the transmitter, we have $t'(t) < t$ at this point, meaning that only $rt'(t) < rt$ bits would have left the transmitter. We would thus accrue a growing buffer underrun even before the network has an opportunity to delay the data. Similarly, we would produce a buffer overrun if $t'(t) > t$. Since we cannot avoid this difference in clock speed, we estimate the transmission rate at the receiver as the total amount of data received at the end of an experiment divided by the time difference between first and last data delivery, i.e., as the average rate at which we have received our data. We then use this estimate as the nominal playback data rate.

IV. Observations

The observations we report on in this section represent only a small sample of this type of experiment on our beacon network, relating to experiments conducted between beacon computers commissioned in April-July 2013 at the University of Macau (MO1), ETH Zurich (CH3), the University of Duisburg-Essen Germany (DE1), the Universities of Tokyo (CH1), Gunma (JP2), and Electro-Communication (JP3) in Japan, and the University of Johannesburg (SA1) in South Africa, as well as two existing beacon machines in New Zealand (NZ1 and NZ2) and beacons at the Ministry of Revenue in Tonga (TO3), Telecom Cook Islands (CK1), and a private site in Switzerland (CH1). All experiments transmit 115 bytes into a TCP socket every 20 ms and log the arrival time of data at the application on the receiving end, i.e., after the receiving TCP socket has reassembled the byte stream. This data yields the "red step curve" in Fig. 1, and thus the corresponding minimum buffer time.

The graphs in Fig. 2 to 9 show the trends in minimum buffer time for selected beacon pairs, giving examples of paths for which deterioration (increase) and improvement (decrease) in minimum buffer time was observed. Periods of "linear ramp" increase or decrease in these graphs indicate experiment downtimes.

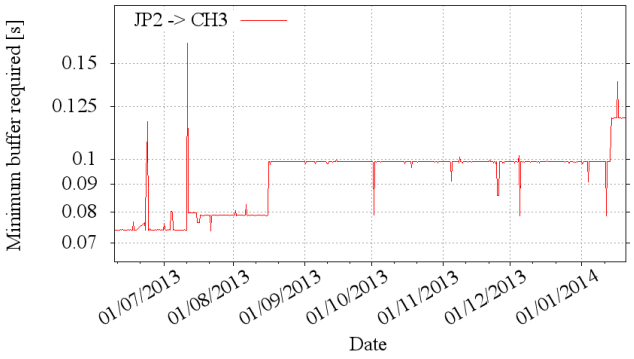

Fig. 2. CH3 receiving from JP2. The initial baseline minimum buffer requirement of well under 80 ms rises in steps to just under 120 ms, with only occasional outliers. Note the relatively stable baselines over extended periods. The recent rise at the beginning of 2014 is a feature we have also observed on a number of other paths, see e.g., Fig. 3 and 9.

The graphs shown here are only a small selection from the relevant experiments conducted between these beacons. From the 31 experiments between the selected beacons outside the Pacific Islands that had current data at the time of writing, 16 show a clear deterioration in minimum buffer time, 10 show no significant change, and only 5 show improvement. Moreover, some of those in the latter two categories show deterioration at the very end of the observation period - sufficient to be noticeable but not enough to confirm a definite trend not associated with the Christmas holidays.

The two beacons in the Pacific Islands were chosen because of recent known changes in their international connectivity. Not surprisingly, 14 of the 18 experiments of these beacons in cooperation with the others above show improvements, and only one shows a slight deterioration.

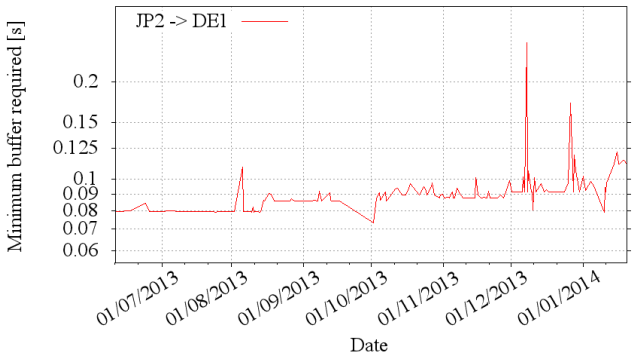

Fig. 3. DE1 receiving from JP2. The minimum buffer requirement is pretty stable at 80 ms until August and then rises to 90 ms for most of the latter half of the observation period.

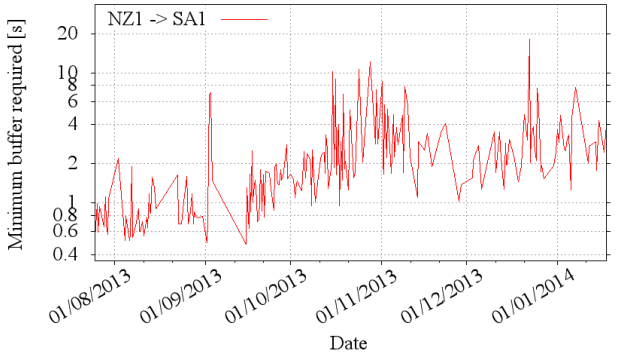

Fig. 4. SA1 receiving from NZ1. Initially, times were stable at at typically 500 - 800 ms with no outliers above 2 s. This was followed by a rise to typical values between 2 and 4 s from September to November. Values such as these rule out TCP for any meaningful voice conversation.

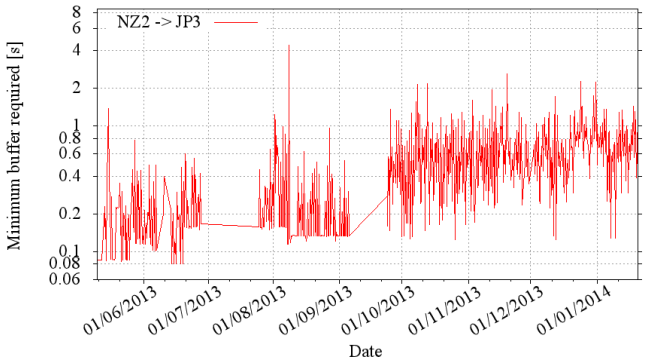

Fig. 5. JP3 receiving from NZ2. The initial buffer times typically fell within the range of 80 - 400 ms, with moderately frequent outliers above. No values below 100 ms were observed after June, and most current values fall in the band of 400 ms to 1 s – sufficient to impede a conversation significantly.

Note that the graphs in Figs. 2, 3, and 5 show typical minimum buffer times increasing or decreasing in discernible steps. This naturally raises the question as to whether it is possible to attribute a step to a particular known event. One suspect to investigate in this respect is daylight savings time (DST): Our experiments run to Universal Time Coordinated (UTC), while local DST applies in many of our beacon locations (Switzerland, Germany, and New Zealand). The

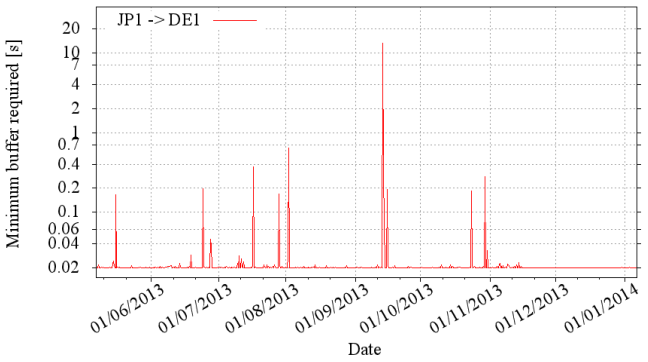

Fig. 6. DE1 receiving from JP1. This path does not exhibit deterioration: most buffer times remain around 20ms, with occasional outliers revealing no evident trend. The typical 20 ms buffering requirement here corresponds roughly to the transmission of a single voice data chunk – a delay that is negligible in practice.

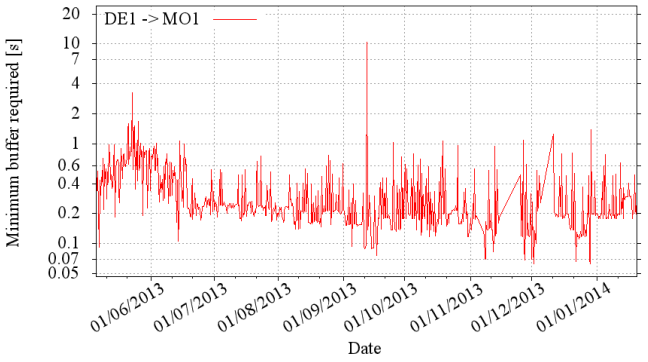

Fig. 7. MO1 receiving from DE1. This path shows a lot of fluctuation but no clear trend. After an inital increase, the minimum buffer time fluctuates around a baseline of 200 ms with outliers in the order of 1s – in practice a noticeable inconvenience.

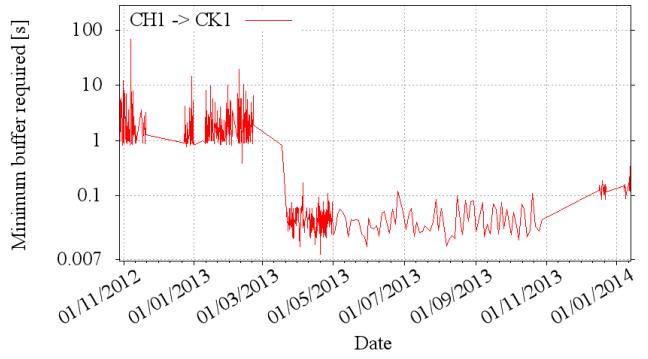

Fig. 8. CK1 receiving from CH1. An intial baseline of 1s drops to around 50ms for reasons that are not entirely clear – possibly a significant increase in de-facto geostationary satellite bandwidth and/or perhaps a switch in offshore ground station to a location topologically closer to Switzerland. Note however that more recent estimates fluctuate less and exhibit a higher baseline of around 100 ms.

question thus arises whether some of these steps may simply reflect temporary changes in local traffic near the transmitter or receiver as an experiment runs an hour earlier or later in local time after a DST switchover. However, closer investigation shows that the steps observed in the results presented here do not correlate with DST changes in either transmitter or receiver

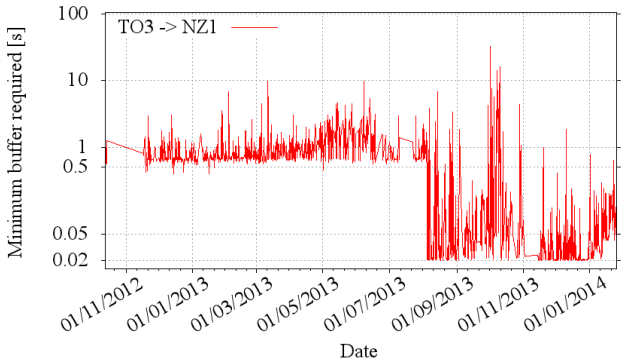

Fig. 9. TO3 receiving from NZ1. This graph is an example of an attributable event leading to improvement - Tonga's connection to the international fibre network (to the Southern Cross Cable Network via Fiji) in early August. The intial baseline of around 1s drops significantly at the time, but note the subsequent rise at the beginning of 2014.

locale.

In fact, we have no obvious explanation for most of these steps, with the notable exception of Tonga's TO3 beacon, where a significant step coincides with Tonga switching international connectivity from geostationary satellite link to submarine fibre in August 2013. This step is evident in all of TO3's other experiments as well.

V. Conclusion

The delay in voice or video playback that is acceptable in an application does of course depend on the circumstances. Readers familiar with the delays incurred on intercontinental phone calls via classical geostationary satellite circuits will know that these increase the risk of talking "on top" of the other party when a response does not arrive within the time frame one is accustomed to. This sort of effect may be acceptable in some contexts (e.g., a short call to a family member) but may be prohibitive in others, e.g., in offshore call centre operations, where communication difficulties of this sort can lead to longer calls, increased staffing cost, and lower customer satisfaction.

In this context, it is important to note that one incurs the minimum buffering times estimated in the previous section *in addition* to the one-way delay of the communication. Our experiments look predominantly at very long distance communication, which features significant round-trip delay of generally several hundred milliseconds. At present, our beacon network does not have accurate time synchronisation and measurement of one-way delay is thus impossible. We plan to address this issue for at least some of our paths in the near future.

A real application buffer would also need to provide a conservative safety margin, as the minimum buffer requirement cannot be determined until after the communication has taken place. Moreover, our values relate to the equivalent of a 200 s call. As the risk of congestion increases with the length of a call, the given buffer times would still offer insufficient protection for many types of longer calls.

In this context, our results show that pure end-to-end TCP VoIP communication may well still be feasible on some long distance paths, e.g., from Japan to Europe. However, over the last half of 2013, this seems to have become impossible for communication between New Zealand and most other regions of the world. It remains to be seen how sustained this deterioration is – we expect to monitor this observable for some years to come. In the context of video, not all applications require smooth replay, however we contend that the technique proposed here is still applicable as a quality indicator in this domain, with data rates and depth/frequency of allowable buffer underruns adjusted accordingly.

Last but not least, we would like to stress that our results do not mean that long-distance VoIP calls have become impossible for users behind firewalls that block UDP, or will necessarily do so in the foreseeable future. As long as any TCP VoIP connection originating at their hosts terminates at a node close enough to allow the long haul part of the call to be carried on a connectionless protocol such as UDP, VoIP should remain feasible provided the random jitter on the connectionless section of the call path remains within reasonable limits. Our beacon network also collects UDP timing data to monitor trends in this respect. Its measurements are publicly available [21].

VI. Acknowledgments

The authors would like to acknowledge the support received from: University of Auckland Faculty of Science Faculty Research Development Fund, Internet NZ, Pacific Island Partners (PIP), and the Pacific Chapter of the Internet Society (PICISOC) as well as that of our other colleagues who host beacons on the network, listed here in roughly eastbound order from the Greenwich meridian: Frank H. P. Fitzek, Gerrit Schulte, Kashif Nisar, Andrew Rarumae, Wayne Reiher, Nevil Brownlee, Andy Linton, Rhythum Kumar, Opetaia Simati, Lopiseni Kavapalu, Timoti Tangiruaine, Niko Rebenich, Stephen Neville, Aaron Gulliver, Ljiljana Trajkovich, Majid Arianezhad, Randy Bush, Jason Cloud, and Muriel Médard. A special mention belongs to Bernhard Plattner and Bernhard Ager for contributing the CH3 beacon and hence essential data for Fig. 2.

References

[1] D. Lee, B. Carpenter, and N. Brownlee, "Observations of UDP to TCP ratio and port numbers," in *Fifth International Conference on Internet Monitoring and Protection (ICIMP), Barcelona, May*, pp. 99 – 104, 2010.

[2] B. Cheng, L. Stein, H. Jin, and Z. Zhang, "Towards cinematic internet video-on-demand," *SIGOPS Oper. Syst. Rev.*, vol. 42, pp. 109–122, Apr. 2008.

[3] E. Freire, A. Ziviani, and R. Salles, "Detecting Skype flows in web traffic," in *IEEE Network Operations and Management Symposium (NOMS), Salvador, April*, pp. 89 – 96, 2008.

[4] Y. Xu, C. Yu, J. Li, and Y. Liu, "Video telephony for end-consumers: measurement study of google+, ichat, and skype," in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC '12, (New York, NY, USA), pp. 371–384, ACM, 2012.

[5] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1." http://www.w3.org/Protocols/rfc2616/rfc2616.html (last visited on December 17th, 2013), June 1999.

[6] J. Postel and J. Reynolds, "File Transfer Protocol (FTP)." http://www.ietf.org/rfc/rfc959.txt (last visited on December 17th, 2013), October 1985.

[7] J. Klensin, "Simple Mail Transfer Protocol." http://www.ietf.org/rfc/rfc2821.txt (last visited on December 27th, 2013), April 2001.

[8] C.-C. Wu, K.-T. Chen, C.-Y. Huang, and C.-L. Lei, "An empirical evaluation of voip playout buffer dimensioning in skype, google talk, and msn messenger," in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '09, (New York, NY, USA), pp. 97–102, ACM, 2009.

[9] ITU, "Methods for subjective determination of transmission quality," ITU P.800, International Telecommunication Union, Geneva, Switzerland, 08/96.

[10] E. Cocker, U. Speidel, N. Rebenich, S. Neville, A. Gulliver, R. Eimann, K. Nisar, S. Hassan, Z. Azziz, M. Dong, and V. Wong, "Measurement of packet train arrival conditions in high latency networks," in *9th International Conference on Information, Communications and Signal Processing, Tainan, Taiwan, 10-13 December*, no. P0190, 2013.

[11] Y. Xiao, S. Chen, X. Li, and Y. Li, "A new available bandwidth measurement method based on self-loading periodic streams," in *WiCom 2007. International Conference on Wireless Communications and Mobile Computing, Shanghai, September*, pp. 1904 – 1907, 2007.

[12] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *IEEE Global Telecommunications Conference, San Francisco, CA, December*, vol. 1, pp. 415 – 420, 2000.

[13] R. Matthieu, "Pathchar." http://www.caida.org/tools/utilities/others/pathchar/ (last visited on January 21st, 2014), May 1997.

[14] B. Mah, "pchar: A tool for measuring Internet path characteristics." http://www.kitchenlab.org/www/bmah/Software/pchar/ (last visited on January 21st, 2014), February 2005.

[15] A. B. Downey, "Using pathchar to estimate internet link characteristics," *SIGCOMM Comput. Commun. Rev.*, vol. 29, pp. 241–250, Aug. 1999.

[16] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 537 – 549, August 2003.

[17] NLANR/DAST, "Iperf home page." http:/http://iperf.sourceforge.net/ (last visited on January 3rd, 2014), March 2008.

[18] S. Chakravarty, A. Stavrou, and A. Keromytis, "LinkWidth: A method to measure link capacity and available bandwidth using single-end probes." http://hdl.handle.net/10022/AC:P:29541 (last visited on January 4th, 2014), January 2008.

[19] K. Lai and M. Baker, "Nettimer: A tool for measuring bottleneck link bandwidth," in *3rd USENIX Symposium on Internet Technologies and Systems, USITS 2001*, pp. 123–134, USENIX, 2001.

[20] C. Dovrolis and R. Prasad, "Pathrate." http://www.cc.gatech.edu/~dovrolis/bw-est/pathrate.html (last visited on January 21st, 2014), January 2004.

[21] "International internet beacon experiment."