IP Queueing Delay Applied to TCP Congestion Control

Guillermo Rigotti
ISISTAN, Fac. de Ciencias Exactas
Universidad Nacional del Centro de la Pcia. de Bs. As.
grigotti@exa.unicen.edu.ar

Abstract

TCPś congestion control was traditionally based on packet losses. This approach leads to an inefficient bandwidth use, because losses must be produced to infer network load. In last years, another approaches have been considered to infer that load, mainly the estimation of sender-receiver delay, known as "one way delay" (OWD). The estimation of this delay enables the sender anticipate losses and lower its sending rate before they occur. A known problem is to accurately estimate OWD, because it requires synchronization between sender and receiver. Various alternatives exist to achieve synchronization (NTP, GPS), but these are not applicable to TCP's flows due to the significant number of flows and because they mainly reside on users' equipment. Another approach that has been recently incorporated in TCP's flow control mechanisms, consists in infer OWD from the RTT value by means of some heuristic. However, this approach has lead to develop complex heuristics that produce inaccurate results, due mainly to the assymetry in the paths and the variability of network load.
In this paper a simple mechanism to estimate OWD is proposed. It is based in the cooperation of the network layer, and consists in obtain OWD as the sum of queueing delays experienced by packets in each router along the path from sender toward receiver.
Keywords: Congestion control, TCP, OWD

1 Introduction

TCP's [1] congestion control mechanisms aim that each flow obtains the higher possible bandwidth, without compete unfairly with another flows (TCP friendly). To meet this objective, each flow has to adjust its sending rate according the network load he perceives.
Initially, TCP congestion control mechanisms (TAHOE, RENO), relied completely on packet losses to adapt its sending rate. Additionally, losses affect all flows that share the congested router, causing that those flows synchronize, generating cycles composed of congestion periods followed by periods of sub-use of network capacity.
In 1989 Jain proposed the use of round trip time (RTT) to anticipate congestion events. This approach enables the sender anticipate a congestion event and to

diminish its sending rate, avoiding packet loss [2]. Some of the most currently used congestion control mechanisms such as TCP Vegas and TCP Fast adopt this approach in addition to packet loss detection.
As described below, the network load is closely related to OWD. Various methods to estimate its value have been proposed, mainly: some RTT-based heuristics, and the use of timestamps (TCP timestamp option) plus synchronization between sender and receiver.
RTT estimation is a simple process because it does not require clocks synchronization. It consists in finding out the elapsed time between sending a packet and receiving its acknowledgement (with certain considerations) [9]. RTT's value is of fundamental importance and is employed to derive parameters such as retransmission timeout (RTO) [3].
However, its value is not useful to estimate the delay from sender to receiver, mainly due to possible asymmetry between sender-receiver and receiver-sender paths. This asymmetry has three main components: 1 - asymmetry in transmission links, for example optic fiber from sender to receiver and satellite from receiver to sender; 2 - asymmetric paths due to routing protocols (acknowledgement packets traverse different routers than data packets); 3 - asymmetry in load, due to transient states of the network.
Due to the low correlation between RTT and OWD [4], the various proposed heuristics to infer OWD based on RTT do not produce accurate results, and in most cases are too complex to be executed on line with the arrival of packets.
The approaches that try to estimate OWD experienced by packets in function of the difference between the sending time at the sender and the receiving time at the receiver, have to resolve the problem of clocks synchronization between sender and receiver. There are two alternatives to achieve synchronization: Network Time Protocol (NTP) and Global Positioning System (GPS). These are suitable for a variety of applications such as VoIP, but cannot be applied satisfactorily to TCP flows, due to its significant number, and its location (user equipment generally unable to GPS).
In this paper we propose an approach to estimate network load. It is based on the buffer occupancy in routers involved in the sender-receiver path. The number of packets waiting for transmission in routers is directly related to network load. To implement the proposed alternative, we define and implement a mechanism at network level (IP) that enables to register the load in each router, from sender to receiver. This is implemented as a new IP option, referred to as "QD option". This is complemented by a second mechanism at transport level (TCP), that enables that information collected by IP be accessible to the congestion control mechanism in use.
These mechanisms are implemented as follows: the former as a new IPv4 [5] option and a new option in the IPv6 [6] hop-by-hop header; the last as a backward compatible modification of TCP timestamp option. Although our proposal implies to carry out some process in each router, we can state that it is negligible (a few instructions in Linux kernel). Furthermore, no additional state needs to be created or maintained in routers.
The rest of the paper organized as follows. In Section 2 related work is presented; in Section 3 different components of delay are analyzed and a way to quantify it is proposed. Section 4 explains how the proposed mechanisms are incorporated in IP and TCP levels, while Section 5 presents some aspects of the implementation in Linux. Section 6 describes simulations carried out to conduct

a first evaluation of our proposal. Section 7 presents our conclusions. Finally, Section 8 refers to relevant bibliography.

2 Related Work

The mechanism proposed in this paper can be characterized as follows; 1- It does not generate additional traffic. It should be noted however, that a minimal overhead is generated (only one IP option) and is only present in certain datagrams. 2- Requires cooperation of routers to compute the estimated delay. It should be noted however that does not require state in the routers. 3- Our method is not based on clocks' synchronization as those using NTP, GPS or time derived from CDMA (radio links).
Among similar work, we can quote [7] . This approach aims to estimate delays in wireless ad-hoc networks to improve TCP efficiency. Unlike our proposal, it maintains generic state (valid for any TCP stream) on routers. This state is specific to the protocol. The delay is calculated through the cooperation of the involved routers. Each router maintain two variables: Q, that represents the exponential average of the queuing delay experienced by packets traversing the router, and T, that represents the exponential average of the queuing delay experienced by the head-of-line packet. For every packet that a router forwards, the sum Q + T is stamped if that of the packet is smaller.
In [8] ,the authors propose an algorithm to estimate characteristics of OWD without the need of synchronization. A difference of our approach, they use ICMP probe packets with two different lengths, sent from two sources to a target. The target dont need any additional functionality. To carry the necessary information, ID field of the IP datagrams is used.

3 Estimation of network load

Our goal is to measure the network load. This is directly related to the size of transmission queues in routers belonging to the path from sender toward receiver. To compute this value, various parameters could be considered, such as number of packet waiting for transmission, percentage of use of queues, transmission delay, etc,
Although any of them could be selected, we choose the last (delay) due to the following: 1- It is a generic measure, suitable to be added by each router. 2- Transmission delay can be used to calculate some commonly used parameters, such as link asimmetry, variations on delay, etc. 3-Taking in account that transmission delay is a frequently used parameter in congestion control mechanisms; our measure could be applied to those mechanisms in place of the original ones (for example RTT). This could be useful to compare their performance with respect to their original parameters.
Mechanisms that depend on network level support, require that all involved routers be able to collaborate, to achieve optimal results .If some routers are not able to do so, it is likely that wrong or distorted results be obtained. In cases such as ECN, where the result may depend on a single router, erroneous results can be obtained (for example, in the particular case that the congested router does not implement ECN). In the case of transmitter-receiver delay, the

situation is different because the value does not depend on a single router. If one or more routers are unable to register delays, the obtained result gets distorted, but possibly can be useful. The above situation may arise for example when certain routers do not implement the QD option or within a MPLS domain, where routers (acting as switches), do not have access to the IP level. The amount of distortion depends, among other factors, on the proportion of routers that do not cooperate.In order to quantify that distortion, simulations described in Section 6 were carried out.
Another aspect to consider is the overhead incurred in routers to estimate the queueing delay experienced by packets; this is discussed in Section 5.
To relate the queuing delay with other components of the total sender-receiver delay, consider a sending host A and a receiving host B. If we indicate the intermediate routers as $R_{\underset{\rightarrow}{xy},i}$, where the first subscript represents the direction (from x to y) and the second the order of the router along the way, we can express the paths from A to B and B to A as follows:

$$R_{\underset{\rightarrow}{ab},1},\ \ldots,\ R_{\overset{\rightarrow}{ab},i},\ \ldots,\ R_{\underset{\rightarrow}{ab},n} \quad and \quad R_{\underset{\rightarrow}{ba},1},\ \ldots,\ R_{\underset{\rightarrow}{ba},i},\ \ldots,\ R_{\underset{\rightarrow}{ba},m}$$

If we consider the total delay experienced by a packet sent from A toward B, as mentioned in Section 1, we can express this delay as the sum of several factors, as detailed below: Let P the number of bits in a packet, $Vt_{(R_x,R_y)}$ the bandwidth of the link that connects adjacent routers R_x and R_y, $dp_{(R_x,R_y)}$ the propagation delay of the link connecting adjacent routers R_x and R_y, $Q_{(R_x,R_y)}$ the number of packets queued to be sent from the router R_x to the router R_y and PR_x the processing time of the packet in the router R_x (delay since the packet is received by the router until it is ready for shipping), we can express the delay (OWD) from A toward B (and analogously from B toward A) as follows:

$$OWD_{\underset{\rightarrow}{ab}} =$$

$$P/Vt_{(A,R_{\underset{\rightarrow}{ab},1})} + dp_{(A,R_{\underset{\rightarrow}{ab},1})} + \tag{t}$$

$$\sum_{i=1}^{n-1} P/Vt_{(R_{(\underset{\rightarrow}{ab},i)},R_{\underset{\rightarrow}{ab},i+1})} + dp_{(R_{\underset{\rightarrow}{ab},i},R_{\underset{\rightarrow}{ab},i+1})} + \tag{t}$$

$$P/Vt_{(R_{(\underset{\rightarrow}{ab},n)},B)} + dp_{(R_{(\underset{\rightarrow}{ab},n)},B)} + \tag{t}$$

$$\sum_{i=1}^{n} PR_{(R_{\underset{\rightarrow}{ab},i})} + \tag{p}$$

$$\sum_{i=1}^{n-1} Q_{(R_{\underset{\rightarrow}{ab},i},R_{\underset{\rightarrow}{ab},i+1})} * P/Vt_{(R_{\underset{\rightarrow}{ab},i},R_{\underset{\rightarrow}{ab},i+1})} + \tag{q}$$

$$Q_{(R_{\underset{\rightarrow}{ab},n},B)} * P/Vt_{(R_{\underset{\rightarrow}{ab},n},R_B)} \tag{q}$$

.

In the above equation we distinguish three types of components that integrate the delay: those identified with references (t) -transmission time, (p) -time of processing- and (q) time spent in transmission queues.
Those identified by (t), represent the time taken to send the packet from host A to the first router, from each router to the following, and from the last router to host B; This transmission time depends only on the characteristics of the links traversed by the packet and kept constant while the route do not change; It is also independent of the network load.

The term identified by (p) represents the packet processing time in each of the routers: from the arrival of the packet until it is queued for transmission in the corresponding output interface. This time depends on several factors, such as hardware, operating system, equipment load, etc. It may be considered independent of the network load.
The components of the delay (t) and (p) depend on the path followed by the packages and change when the routing decision is modified.
Finally, the components identified by (q) correspond to queuing delays in each of the routers: these delays are directly dependent on the number of packets waiting to be transmitted by the corresponding output interface, and provide an accurate measure of the load on the network.

4 Functions in routers and hosts

As mentioned above, the proposed mechanism is based on defining a new IP option - queueing delay:QD - (see Section 4.1) that records the delay experienced by the datagram in the queues of routers, from sender toward receiver. A datagram that carries this option is modified in each router, which adds its own delay to the delay carried in the IP option.
TCP sender decides according to the congestion control algorithm used, when measure the delay toward the receiving side (in all segments carrying data, at certain intervals of time, etc.). To do so, it asks to IP generate a QD option initialized to zero in the corresponding datagram. When the datagram reaches its destination, the QD option contains the accumulated queuing delay recorded by routers, which is made accessible to the TCP receiver. The receiver should immediately send back that information, so it can be known by the sender in a period equal to RTT. To achieve this, the delay is sent into the next segment. Timely receipt of the delay is critical to correctly adjust congestion control parameters at the sender side

4.1 Changes to IP level

The amendments necessary at the IP level consist of the addition of the option "queuing delay" (QD) . It is described below for IP versions 4 and 6.
According to the IPv6 specification [6], a new type of option is defined (QD) within the hop by hop extension header. This header (and therefore the QD option) is processed by all routers. The presence of this header is specified in the IPv6 header, with a next-header value of zero. The hop-by-hop header, including their fields and the QD option, are shown in Fig.1.
Like the other options, QD is encoded in TLV format. It consists of the following fields:
type: the two first bits set to 0 indicate that if a router does not recognize the option, he must process the next header; the third bit, set to 1, indicating that the value of the option (cumulative delay) can be modified by routers; the 5 bits remaining indicate the type of option (QD);
length: set to 4, indicating data length,
value: the last field, which corresponds to data, composed of 32 bits containing the cumulative queuing delay experienced by the packet.
The option must be aligned to 8n + 2, according to the IPv6 specification.

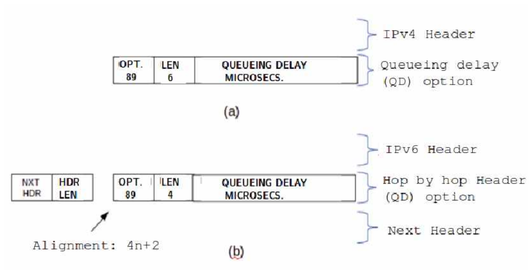
IPv4 Header
OPT. 89
LEN 6
QUEUEING DELAY MICROSECS.
Queueing delay (QD) option
(a)
IPv6 Header
NXT HDR
HDR LEN
OPT. 89
LEN 4
QUEUEING DELAY MICROSECS.
Hop by hop Header (QD) option
Next Header
Alignment: 4n+2
(b)

Figure 1: Queueing delay option (a):IPv4 (b):IPv6

According to IPv4 specification [5], we define the option "queing delay" (QD) with the following fields:
1-copy: set to 0 (do not copy the option in fragments),2-class: set to 2 (debugging and measurement),3-length: set to 8, indicating that 4 bytes are reserved to hold the value of the option, that represents the accumulated queuing delay expressed in microseconds.
The selected option number is 25, still unused. Thus, the value of the first byte is 89 (decimal). value: finally there are 4 bytes corresponding to the value of the option.

4.2 Changes to TCP level

The TCP sender must be able to ask IP generate and send a QD option. On the receiving side, TCP should be able to access the value of the QD option. The interaction between TCP and IP is performed through kernel resident code. Additionally, in the connection-establishment phase, both TCP sides should negotiate the use of QD option in order they be able to exchange delay information in the connection-established phase.
To perform these two functions an alternative use of the timestamp option is proposed. This is compatible with RTTM and PAWS [9].
The initial negotiation is based on the fact that the value of tsecr (timestamp option) is set to zero in the segment that carries SYN flag. The interchange of delay information carried out during the connection-established phase, is based on the fact that not every segment have to carry a timestamp value in order to perform traditional processes. We propose a mixed use of tsecr field: in some segments, it contains time information; in others, information about delays.

At the beginning of the connection, each TCP checks whether he is in position to announce its ability to process the IP QD option; the following conditions must be satisfied: TCP has been configured to use QD, and its local IP must support the QD option. The side that initiates the connection, if able to use the option, announces this fact in the SYN segment sent for establish the connection: the echo field (tsecr) of the timestamp option, instead of being sent with zero value, is sent with its first two bits set to 1. The side receiving the connection request, if able to accept QD processing, send a modified value in the echoed tsval (tsecr field of its generated timestamp option): the new value is obtained xoring the two most significant bits of the original one (tsval field of the received timestamp option) with "11". This result is sent in tsecr field.

In this way both TCP sides agree to the use of QD option. This negotiation is also true in the case of simultaneous connections.

As previously mentioned, contrary to the assertion in [10], it has been shown that taking multiple samples of RTT does not improve the calculation of RTO [11]. In [3] the new way to calculate RTO based on fewer RTT samples is explained. Therefore, it is possible to use the timestamp option to carry the queuing delay besides time information.
The sending rate of QD samples from side A to side B, is determined by the congestion control algorithm used in A. When B receives a datagram carrying the QD option, gets its value and immediately sends it to A in the timestamp option corresponding to the next segment to be sent. This ensures that the delay is received within an interval equal to RTT from the time it is requested.
To sum up, the timestamp option returned by B to A has the following values:
tsrec: the value received from A into the tsval field.
tsval: delay received in IP QD option of the datagram encapsulating the segment sent by A.
To identify a timestamp option that carries a value of delay instead of the usual, the side that originates the request must save the tsval value contained in its timestamp option. In this way he is able to match the response with his request.
The receiver side must not use such timestamp option to calculate the RTO. In addition, he must have provisions to avoid generating two segments with the same tsval.

5 Implementation

The processing carried out by TCP is executed at the beginning of the connection and then by each segment that carries the QD option. Despite that these processes are not critical, its implementation should be simple and efficient.
The process of QD option in routers (at IP level) is critical because the code must be executed in real time with the arrival of each packet containing that option.
We considered the following aspects in order to achieve efficiency: amount of instructions, places where kernel code needs to be modified, and state that should be kept for each datagram, from its arrival until it is forwarded.
In our case, the implementation was done in Linux [12], kernel 3.14.5. In this context, the following alternatives were considered:
1-Record the send and receipt times of a datagram. Sum the difference of these values to the QD option value, and
2-Estimate queueing delay based on the number of packets to be transmitted before the datagram be sent (number of packets in the transmission queue). Sum this value to that of the QD option.
Despite the first alternative is more simple, we choose the second due the following reasons:
The first choice requires maintain state for each datagram: when a datagram is received, its arrival time has to be registered and kept in the associated sk_buffer. If we want to avoid storing state (sk_buffer), it would be necessary to subtract the arrival time to the QD option value; then add the departure time. In this way, QD value has to be considered signed, because routers are

not synchronized. Moreover, it should be noted that in this way, the packet processing time is included in QD value. Additionally, if mechanisms such as Generic Segmentation Offload (GSO) or TCP Segmentation Offload (TSO) are used, the second step of the process (obtain sending time and modify QD value) must be inserted in the appropriate code (belonging to GSO or TSO), which is not always possible because hardware implementations.
The second option was choosen due to its efficiency and simplicity. It requires only one point of modification in kernel code: just before the datagram is queued to be transmited. The process consists in estimate the time that transmission of previous packets will demand, and adding this value to that of QD option. The estimated value is obtained multiplying the number of packets queued by the estimated packet transmission time.
The advantages of the last choice over the first are the following: 1-only one modification to kernel code is needed, 2-do not require keep state relative to each datagram, 3-do not include processing time in the estimated queueing delay, 4-does not depend on the accuracy of each computer's clock.
One aspect to consider is how to estimate the transmission time of the previously queued datagrams. The proposed solution is to calculate that value as a function of the number of bytes (or datagrams) queued, the bandwidth, and the MTU of the transmission link. In our case, we consider datagrams of same length as the MTU. As in the first case, we must consider the use of GSO or TSO. If these Mechanisms are active, the amount of bytes or datagrams previously queued must be requested to them.

6 Simulation

Given the impossibility of checking the behavior of our proposal with real machines (physical or virtual) because it is necessary to involve a considerable number of routers, simulations were performed. We use Network Simulator 2 (ns-2) [13], version 2.3.5 running on Linux with kernel 13.14.5.
In a first step the simulation was used to check how is affected the measurement of delay queueing, by the fact that some of the routers do not implement the QD option. At a later stage, the developed code will be used to compare the queuing delay that we obtain, with other OWD estimations carried out by some algorithms of TCP congestion control (eg Vegas or Fast).
The simulated topology is composed of thirty routers and hosts running applications which generate network load. Considering topology, paths between hosts used to estimate delay are symmetric; asymmetry is obtained varying the load generated in the network. This load is originated by multiple UDP flows with CBR applications. In all cases the location of applications and their sending rates and duration were generated randomly, with periods of asymmetry in both directions.
Results are displayed in Figure 2. The left side shows the relationship between the round trip delay (RTT) taken by the source host, the queuing delay perceived by the destination host, and the OWD from origin host toward destination host. The value of RTT is taken by the source host in standard way, queuing delay is taken according to what is proposed in this paper, and the OWD is taken as a function of simulation time and serves only as a reference
It can be seen the correspondence between the value of OWD and the value

of the queuing delay, regardless of the asymmetry in both directions, which is observed between 27 and 29 seconds of simulation.
An important aspect to evaluate, is how queueing-delay estimation is affected by the fact that some routers are not able to process the QD option; this can be seen on the right side of the figure, where queuing-delay is shown for cases in which 100%, 75% and 50% of the routers are capable to process the QD option. Estimation accuracy decreases with decreasing the number of routers, but remains proportional, even in periods of asymmetry.This is because the resulting delay is the aggregate of those taken by all routers.
On the other hand, when the number of routers capable of measuring the delay increases, the value obtained tends to the actual value (OWD).
Moreover, it is observed that during periods of asymmetry caused by variations in load, the estimated value is stable, which does not happen with OWD estimates based on RTT.

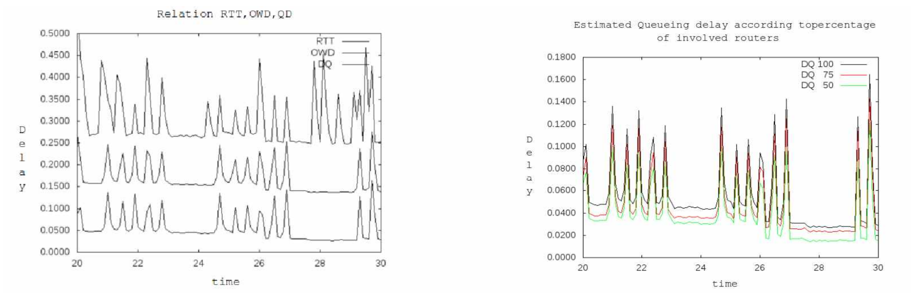

Figure 2: Relation RTT-OWD-QD (left). QD values (right).

7 Conclusions

Prevent Internet overloading depends largely on transport level protocols. A key role is performed by TCP, which is the most used protocol and therefore which generates most of network traffic.
TCP congestion control algorithms are responsible for avoiding network congestion, and so they have been under constant research and improvement. In recent years, it has begun to use the delay between transmitter and receiver as a parameter to adjust the load introduced into the network. It has been used alone or in combination with packet loss. Similarly, IP level mechanisms working in collaboration with TCP (ECN, AQM, etc.) have been improved to allow a more effective reaction against losses.
This paper has proposed a mechanism at IP level, that helps TCP assess more accurately the transmitter receiver delay caused by network load, and hence react timely to changes in network load, avoiding packet loss. To carry out the proposed functions both IP and TCP have been modified. In both cases the changes were minimal, and code efficiency was taken into account. Simulations were performed that demonstrate that our approach works adequately in asymmetrical networks and in situations where a number of routers have not implemented the QD option.
The next steps to complete the validation of the proposal are the use of queuing

delay in different congestion control algorithms employed by TCP (including Vegas and FAST). In order to materialize this step we will implement the congestion control algorithms as Linux kernel modules; this approach allows separation of congestion control of the rest of the TCP code, simplifying the code. For the simulation, the characteristic of ns-2, that allows a quick port of Linux kernel modules into the simulator, will be exploited.

References

[1] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, DARPA, September 1981.

[2] Jain, R., "A Delay Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks", Computer Communications Review, ACM SIGCOMM, pp. 56-71, October 1989.

[3] Paxson, V; Allman, M.; Chu, J.; M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, June 2011.

[4] Pathak1, Himabindu Pucha1, Ying Zhang2, Y. Charlie Hu1, Z. Morley Mao, "A Measurement Study of Internet Delay Asymmetry", (https://web.eecs.umich.edu/-zmao/Papers/pam08-owd.pdf)

[5] "Internet Protocol - DARPA Internet Program Protocol Specification", RFC 793, DARPA, September 1981.

[6] S. Deering R. Hinden, "Internet Protocol, Version 6 (IPv6):Specification", Network Working Group, IETF, RFC 2460, December 1998

[7] K. Sundaresan et al., ATP: A Reliable TransportProtocol for ad hoc Networks , ACM Intl. Symp. Mobile Ad Hoc Net. and Comp. (MOBI HOC), 2003, pp. 64 75.

[8] Antonio.de.Rochaet al. " A Non-cooperative Active Measurement Technique for Estimating the Average and Variance of the One-Way Delay", Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet Lecture Notes in Computer Science Volume 4479, 2007, pp 1084-1095

[9] D. Borman, B. Braden, V. Jacobson, R. Scheffenegger, Ed."TCP Extensions for High Performance",RFC 7323,September 2014

[10] Jacobson, V.;Braden, B.; D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992

[11] Allman, M.; V. Paxson, "On Estimating End-to-End Network Path Properties", Proceedings of the ACM SIGCOMM, Technical Symposium, Cambridge, September 1999

[12] http://www.kernel.org

[13] S. McCanne; S. Floyd, "ns-LBNL Network Simulator 2", http://www.isi.edu/nsnam/ns/