Active Window Management: Reducing Energy Consumption of TCP Congestion Control

Roberto Bruschi[1], Alfio Lombardo[2], Carla Panarello[3], Fabio Podda[1], Enrico Santagati[2], Giovanni Schembra[2]
[1]CNIT Research Unit of the University of Genoa, Italy, email: {name.surname}@unige.it
[2]DIEEI University of Catania, Italy, email: {name.surname}@dieei.unict.it
[3]CNIT Research Unit of the University of Catania, Italy, email: {name.surname}@dieei.unict.it.it

***Abstract*—In the last few years, the ever increasing attention to the eco-sustainability of Internet has pointed out the need of providing advanced power management schemes in network devices, in order to be able to reduce energy consumption according to the traffic load. In this context, starting from the observation that most of the traffic of today's Internet is carried by TCP, we focused on the impact that the dynamics of the TCP congestion control may have on energy efficiency of end hosts. In this paper we demonstrate that great energy consumption is related to network congestion conditions. Moreover, we propose to use a technique, called Active Window Management (AWM), with the aim of driving TCP sending rate according to the network available bandwidth, so improving congestion control and providing high TCP performance. AWM requires no changes to any TCP version. We demonstrate that the TCP traffic shaping performed by AWM allows a great reduction of the energy consumption in end hosts, with respect to the case where only TCP is used as congestion control.**

I. Introduction

In the last few years, the ever increasing attention to the eco-sustainability of Internet has detected the lack of proportionality between the energy consumption of network devices and their workloads as one of the primary causes of a great waste of energy. In this perspective, a number of studies point out the need of providing network devices of hardware platform including advanced power management schemes, in order to be able to reduce energy consumption according to the traffic load. At the same time, the optimization of network protocols is desirable since they determine the dynamics and features of traffic patterns which may impact the energy efficiency of power management schemes.

To this purpose, since most of the traffic of today's Internet is carried by TCP, we focused on the impact that the dynamics of the TCP congestion control may have on energy efficiency of end hosts. More specifically, we will demonstrate that great energy consumption is related to network congestion conditions which determine, through the TCP congestion control, the dynamics of TCP traffic. In particular, losses and retransmissions are responsible for a large waste of energy. On the other hand, a huge number of techniques have been proposed in the past to control network congestion and reduce TCP losses and retransmissions. However, some of them miss the challenging target of both avoiding losses and keeping goodput close to the network capacity (e.g., active queue management (AQM) techniques); others require changes in both network routers and hosts that make them difficult to deploy [e.g., eXplicit Control Protocol (XCP)]. In [1], [2], [3], [4], a novel mechanism, called Active Window Management (AWM), has been proposed with the aim of controlling the queue length in network routers, maintaining it almost constant to achieve no loss, while maximizing network utilization. In this paper we will demonstrate that the reduction of TCP losses and retransmissions, together with the reduction of the TCP connection life-time provided by AWM, besides improving network performance, allows to reduce energy consumption of end host.

The paper is organized as follows. Section II describes power management facilities and energy consumption of modern Personal Computers and presents a discussion about the impact of traffic patterns on the energy consumption. In Section III we motivate the choice of using the AWM mechanism to overcome the energy inefficiency of TCP and briefly resume the main characteristics of the AWM algorithm. In Section IV we present some numerical results and, finally, in Section V some conclusions are drown.

The research leading to these results was funded through the European Union Seventh Framework Programme (FP7/2007-2013), under grant agreement n. 257740 - Network of Excellence TREND (Toward Real Energy-efficient Network Design) and the Econet project (low Energy COnsumption NETworks).

II. PC Power Management and Energy Consumption

The Advanced Configuration and Power Interface (ACPI) is the most widespread interface for power management of modern computing systems, and provides a standardized interface between the computer hardware, where power management capabilities are realized, and the software. This standard interface completely hides the details of CPU internal hardware techniques to reduce power consumption, which may differ depending on processor, to the Operating System (OS) and SW applications, offering a simplified view of the available energy saving capabilities. The ACPI standard provides energy-saving modes by means of the Performance (P-) and the Power (C-) states, which could independently be used and tuned in the largest part of modern processors. However, due to its simplicity and effectiveness, the C-states constitute the most significant part in processor's power management techniques.

C-states substantially represent the processor activity: C_0 is the active power state, i.e. when the CPU executes instructions,

978-1-4673-3122-7/13/$31.00 ©2013 IEEE

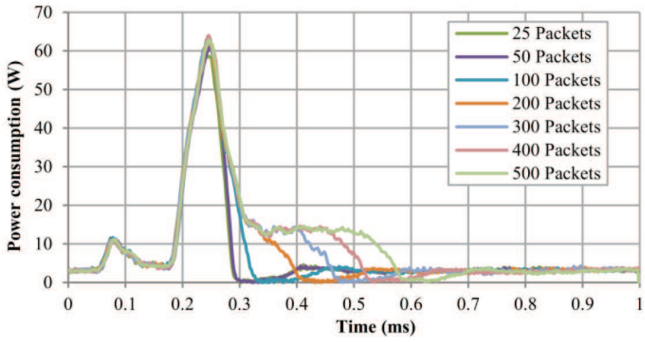

Fig. 1. Energy consumption of the CPU when receiving a burst of a variable number of packets.

while C_1,..., C_n correspond to some sleeping states, when CPU does not execute instructions and dissipates less power and heat. In the C_0 state, the ACPI allows to change the processor performance by means of P-states. P-states modify the operating energy configuration by altering the working frequency and/or voltage. Thus P-states allow us to change the power consumption and the performance of a CPU, but the transition time between different P-states is generally very slow, due to silicon electrical stability issues. For this reason their frequent automatic tuning is not enabled in many OSs. As the sleeping state becomes deeper (moving from C_1 to C_n) and thus power consumption becomes lower, the transition to the active state requires more time and energy. These primitives are very effective (more than P-states) because allow us to literally shut-off a set of internal CPU components when they are not used, avoiding some energy wastes. However, the major performance limitations are due to wake-up times and energy requirements to re-activate the sleeping components. As an example, let us consider a general-purpose CPU supporting the ACPI that receives bursts of packets. When the first packet is received by the network interface, the sleeping CPU is forced to wake up to serve incoming traffic. Then, the CPU starts to switch its internal sub-components on, and hence it requires some additional energy (as well as time) for this purpose. Finally, when CPU is completely up, it can start processing the incoming packets. Fig. 1 shows the behavior of an Intel Core i5 processor when receiving IP packet bursts with different lengths. Here, the first packet is signaled to the CPU and after approximately 50 μs we have the first hardware startup consumption due to the re-activation of CPU memory controllers. After that, at about 250 μs, we have a huge spike of energy due to the re-activation of CPU cores that finishes the transition from sleeping state to the active one; now the packet processing can start and its energy consumption is clearly visible in Fig. 1.

A. Traffic pattern impact on power consumption

C-state transitions are mainly driven by the OS and device I/O operations. When the CPU finishes serving its job backlog, the scheduler of the OS may decide to enter in a sleeping state (C_1,,C_n) and in such state the CPU can wake only by the scheduler or by the interrupt coming from an I/O component (like Network Interface Card (NIC), keyboard, etc.). Since network traffic dynamics (and then the TCP itself) can heavily influence interrupt generation from I/O hardware and, then, the number of C-state transitions, they consequently impact power consumption of the processor and of the entire computing system.

More specifically, energy consumption of such a system is driven by both the actual workload of the CPU (e.g., the number of packets to be processed), and the rate of transitions between idle and active modes. The density of idle-active transitions depends on two processes, namely the one related to packet inter-arrival times and the one concerning the packet service (i.e., how much time the CPU requires to process all the information contained into an incoming packet). When the average packet inter-arrival time gets larger than the packet service time, the transition density increases, and the computer system would experience high energy inefficiencies.

It is worth noting that the packet service process mainly depends on the CPU capacity and the computational complexity of the operations to be performed on the incoming traffic, whereas packet-level dynamics may depend on much different aspects, usually related to the network scenario and sometimes very difficult to be predicted, such as available bandwidth, network congestion, packet losses, Round-Trip-Time (RTT) variations.

Unfortunately, in this context, the TCP congestion control dynamics play a relevant role. In fact, TCP traffic is well-known to provide different performance and working behavior (e.g., life time, number of retransmissions) depending on network parameters, such as end-to-end bandwidth, delay and packet loss ratio. Now, the effects of network parameters on TCP behavior affects also the energy consumption of the end-hosts. More specifically, the presence of any bottlenecks in the network, causing loss of packets, produces, through the TCP congestion control, the retransmission of lost segments, which causes the sender to wake-up an additional number of times, so wasting additional energy. In order to validate the above considerations, we performed some experimental measures (by using the same testbed environment used in Section IV) by using a standard PC with an Intel I5 processor receiving a TCP data transfer when the end-to-end bandwidth is larger enough to avoid any loss and related retransmissions (1Gb/s), and when a constraining bottleneck of 10 Mb/s is introduced in the access router. The obtained results, reported in Fig. 2, clearly show that in case of end-to-end bandwidth equal to 1 Gb/s, we have few CPU idle-active transitions (approximately one per RTT), while in the second case the transitions and then the energy consumption spikes happen much more frequently due to both the enlargement of packet inter-arrival time and the presence of a high number of retransmissions after any expired time out.

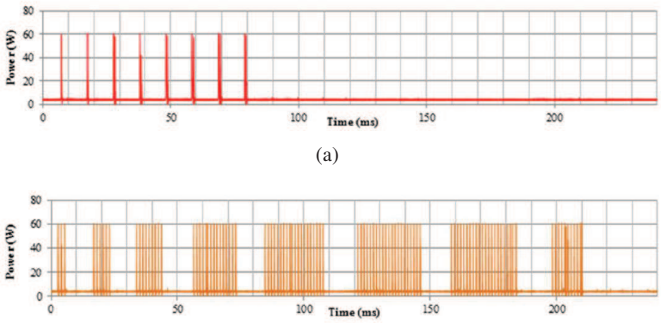

Fig. 2. CPU energy consumption during a TCP connection carrying 100 packets, in the cases the end-to-end bandwidth is set to 1Gb/s (a) and 10Mb/s (b).

III. Overcoming TCP energy inefficiency

The considerations in the previous section suggest that great energy saving can be achieved if the TCP sending rate is tuned according to the network available bandwidth, and losses and retransmission are reduced or even avoided. In other words, the implementation of a very efficient congestion control may allow to reduce the power consumption, besides improving TCP performance. Addressing performance degradations in end-to-end congestion control has been one of the most active research areas in the last decade. A huge number of techniques have been proposed in the past, to control the network congestion and reduce TCP losses and retransmissions ([5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]). However, some of them miss the challenging target of both avoiding losses and keeping goodput close to the network capacity (e.g., active queue management (AQM) techniques); others require changes in both network routers and hosts that make them difficult to deploy [e.g., eXplicit Control Protocol (XCP)]. In [1], a novel mechanism, called Active Window Management (AWM), has been proposed with the aim of controlling the queue length in network routers, maintaining it almost constant to achieve no loss, while maximizing network utilization. In this paper we demonstrate that, thanks to AWM, TCP losses and retransmissions are substantially reduced and the resulting TCP traffic pattern allows a great energy saving in end hosts. In Section III-A, we briefly describe the main characteristics of AWM.

A. Active Window Management

The goal of AWM is to maximize link utilization and at the same time avoid losses. More specifically, let us focus on two generic interfaces, A and B, of an access bottleneck node implementing the AWM algorithm (in the following, we will refer to this node as AWM node). For each packet crossing the AWM node, the AWM algorithm monitors the queue length of the output buffer ($Q_O^{(B)}$ in Fig. 3) loaded by data packets coming from the TCP sources and estimates the number of bytes, called *Suggested Window*, that the AWM node should receive from each TCP source to maintain the queue length

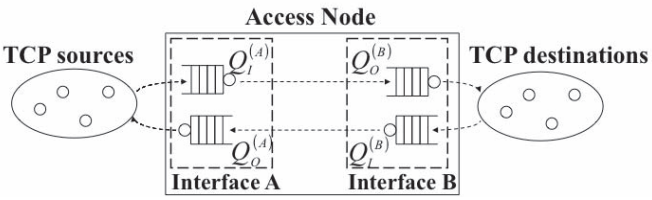

Fig. 3. Buffers considered in the AWM algorithm.

of the output buffer $Q_O^{(B)}$ as constant as possible and close to a *target* value. The *target* value should be significantly smaller than the buffer size (to avoid losses), and higher than zero (to avoid under-utilization). Then the mechanism exploits the TCP flow control mechanism to drive the TCP sending rate according to the network availability. In particular, it acts on ACK packets sent by receivers to the corresponding TCP sources (crossing the AWM node from the interface B to the interface A and enqueued in the output buffer $Q_O^{(A)}$) and controls the transmission rate of the TCP sources by manipulating the TCP header field, called *Advertised Window*, designed for the flow control mechanism of TCP. To achieve its goal, AWM action has the purpose of making the *Suggested Window* parameter more constraining than both the *Congestion Window* and the receiver *Advertised Window* each time the queue length exceeds the *target* value, thus driving the TCP sending rate. Obviously, in order not to interfere with the original TCP flow control algorithm, the AWM algorithm overwrites the value of the *Advertised Window* field with the calculated *Suggested Window*, if and only if *Advertised Window* is greater than *Suggested Window*, whereas in the opposite case the *Advertised Window* field remains unchanged. Let us stress that the TCP sender works as usual, that is, it determines the sending rate by taking whichever is the lower between the *Advertised Window* and the *Congestion Window*.

Let us note that the algorithm is very simple, and does not affect router performance appreciably. In addition, the AWM algorithm maintains one *Suggested Window* value for each network interface and updates are applied to *Advertised Window* in ACKs coming from a given interface, irrespective of the particular TCP connection they belong to. Therefore, no per-flow state storage is needed, ensuring scalability as the number of flows increases. Moreover, AWM does not require any modification either in the TCP or in host implementations. In [1], the authors showed results obtained comparing AWM to AQM techniques, demonstrating that when the access node implementing AWM is the bottleneck in the network, TCP performance is very close to a pseudo-constant-bit-rate protocol, providing no loss and maximum utilization given that buffer queue never saturates and never empties.

The AWM technique is implemented in the network access nodes, that is, in the nodes through which both incoming and outgoing packets related to a given TCP connection are forced to go, whatever the routing strategy used in the network. Let us note that the above applicability conditions occur in many relevant cases, for example in the access gateway of many

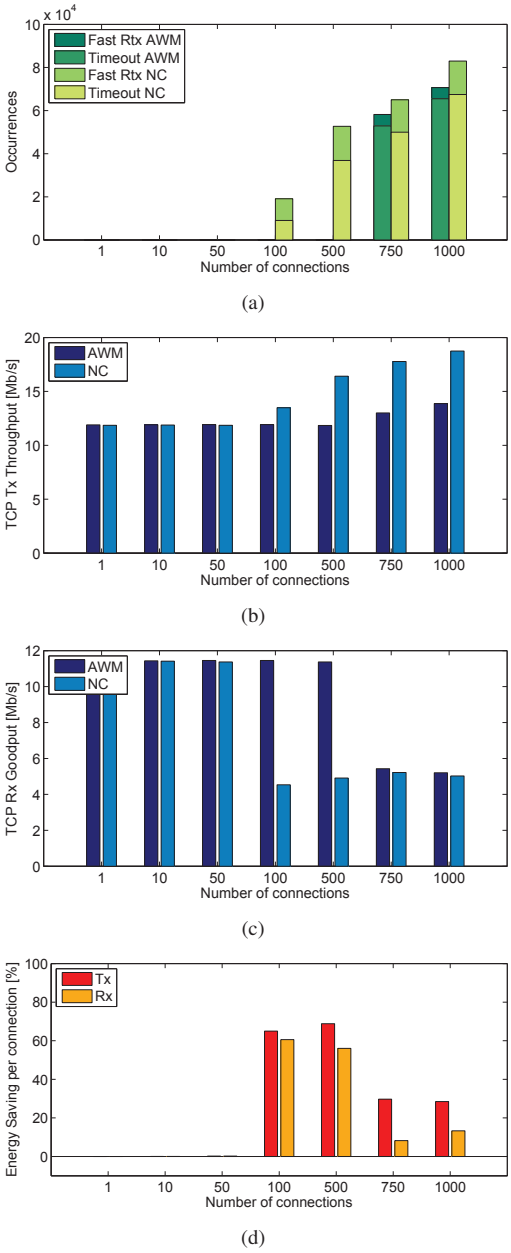

Fig. 4. Timeouts and Fast Retransmit occurrences (a), Transmitted Throughput (b), Received Goodput (c) and AWM Energy saving per Connection (d) when the network is loaded by greedy TCP traffic.

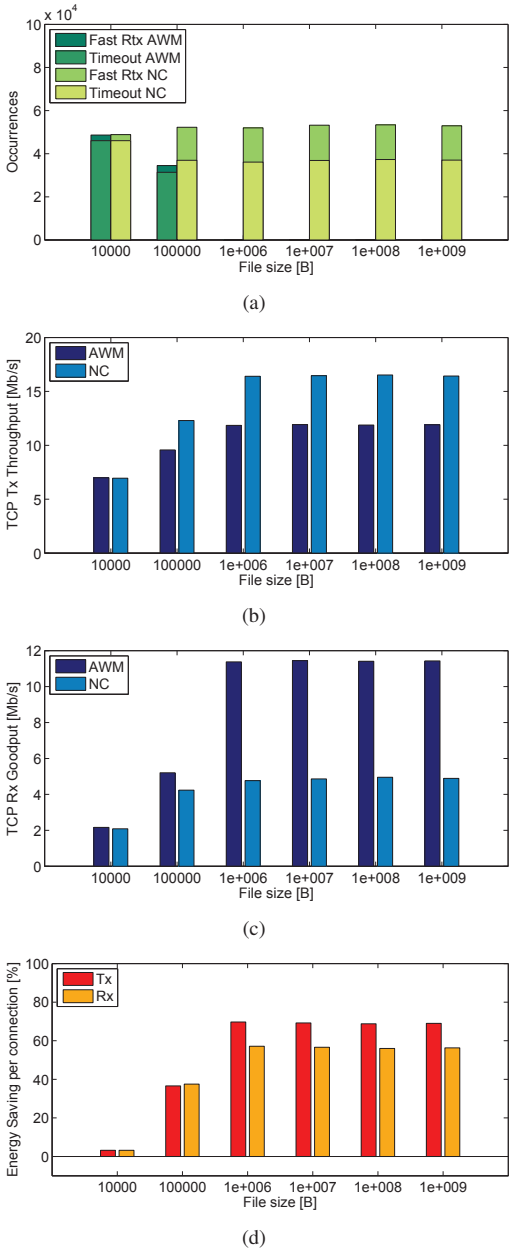

Fig. 5. Timeouts and Fast Retransmit occurrences (a), Transmitted Throughput (b), Received Goodput (c) and AWM Energy saving per Connection (d) when the network is loaded by 500 connections.

university campuses, factories, offices, and home networks, that is, where small or large networks are connected to the Internet with a single router or an Internet service provider (ISP). Moreover, since the deployment of the AWM requires changes at access nodes only, this is a major advantage for facilitating its adoption.

IV. Numerical results

In this section we show that applying AWM to drive TCP source sending rate according to the bottleneck available bandwidth and avoiding losses, besides effectively reducing the number of retransmissions, allows to improve energy efficiency of the end hosts. To this aim, we performed a number of test sessions in different scenarios. More specifically, we used a simple topology with several concurrent TCP flows sharing the available bandwidth in the bottleneck node. This node, in turn, either implements the AWM mechanism or does not implement any additional congestion control mechanism besides the classical one performed by TCP (the two cases are indicated in the figure as AWM and NC, respectively). The bottleneck available bandwidth is set to 12Mb/s (to represent typical residential broadband networks, e.g., DSL),

whereas the number of flows and the amount of data to be transmitted varies in each experiment. In order to evaluate energy efficiency of the end hosts separately at the sender and receiver sides, we used unidirectional data transfer.

In Fig. 4 we report results obtained for long-lived TCP connections, when the number of concurrent flows is set to 1, 10, 50, 100, 500, 750, 1000, whereas the amount of data to be transferred is set to 1GB. Fig. 4(a) shows the number of occurrences of TCP Timeouts (TO) and Fast Retransmit (FastRtx) events for the aggregate flow. Results show that when AWM is implemented in the bottleneck node, its congestion control mechanism allows to avoid retransmissions even when the bottleneck is loaded by a great number of concurrent flows (Timeouts and Fast Retransmit events occur for a number of flows higher than 500). Instead, when the congestion is controlled by TCP, Timeouts and Fast Retransmit events occur with a number of concurrent flows smaller than 100, and occurrences are always greater than in the AWM case. Fig. 4(b) and Fig. 4(c) show the aggregate throughput and goodput. According to the results of 4(a), when the number of TCP connections is lower than 100, no differences in throughput or goodput are present between the AWM and the NC cases, since no congestion is present in any of these cases. As far as the number of TCP connection increases, the NC throughput increases because of an increase of losses and retransmissions, and the goodput decreases as well, whereas AWM, by controlling congestion and effectively reducing losses and retransmissions, provides better performance even with high number of TCP connections. The effects of a more efficient congestion control of AWM, with respect to TCP (NC case), on energy efficiency, is visible in Fig. 4(d), where the average energy saving for each connection, measured in the AWM case, with respect to the NC case, is reported for both the sender and the receiver sides. The results show that, for all the cases where a reduction of retransmission is guaranteed by AWM, great energy saving is provided by AWM with respect to the NC case.

Fig. 5 shows results obtained in the case of 500 TCP connections, when the amount of data to be transmitted is set to 10kB, 100kB, 1MB, 10MB, 100MB, 1GB. The number of TCP connections is constant for the duration of the tests (i.e. for each connection closing, a new one is opened). In Fig. 5(a) the number of occurrences of TCP Timeouts and Fast Retransmit events for the aggregate flow is reported. Note that for small file sizes (10kB and 100kB) AWM is not able to avoid losses and retransmissions, since the life times of TCP connections are not long enough to allow the AWM mechanism to reach its steady state. Nevertheless, even in these cases, the performance provided by AWM are not worse than the one achieved by TCP in the NC case. Fig. 5(b) and Fig. 5(c) show again that AWM is able to improve performance with respect to the NC case, and provides lower throughput (since retransmissions are avoided) and higher goodput. The effects of these performance improvement on the energy efficiency is shown in Fig. 5(d), where the energy saving per connection is shown, demonstrating that AWM, besides improving performance, allows to sensitively reduce energy consumption of end hosts with respect to TCP.

V. Conclusions

In this paper we have focused on the impact that the dynamics of the TCP congestion control may have on energy efficiency of end hosts. More specifically, we have demonstrated that great energy consumption is related to network congestion conditions which determine, through the TCP congestion control, the dynamics of TCP traffic. In particular, losses and retransmissions are responsible for a large waste of energy. On the other hand we have argued that great energy saving can be achieved if the TCP sending rate is tuned according to the network available bandwidth and losses and retransmission are reduced or even avoided. In other words, the implementation of a very efficient congestion control may allow to reduce the power consumption, besides improving TCP performance. To this purpose, we have evaluated effects on energy efficiency of end hosts, of using a mechanism, called Active Window Management (AWM), designed for controlling congestion and improving TCP performance. In this paper we demonstrate that, thanks to AWM, TCP losses and retransmissions are substantially reduced and the resulting TCP traffic pattern allows a great energy saving in end hosts.

References

[1] M. Barbera, A. Lombardo, C. Panarello, G. Schembra, *Queue Stability Analysis and Performance Evaluation of a TCP-Compliant Window Management Mechanism*, IEEE/ACM Trans. Networking, vol.18, no. 4, pp. 1275-1288, Aug 2010.

[2] M. Barbera, A. Lombardo, C. Panarello, G. Schembra, *Active Window Management: an efficient gateway mechanism for TCP traffic control*, in Proc. of IEEE ICC 2007, Glascow (Scotland), June 2007.

[3] M. Barbera, M. Gerla, A. Lombardo, C. Panarello, M. Y. Sanadidi, G. Schembra, *Active Window Management: Performance Assessment through an Extensive Comparison with XCP*, in Proc. of IFIP Networking 2008, Singapore, May 2008.

[4] A. Lombardo, C. Panarello, G. Schembra, *Applying Active Window Management for Jitter Control and Loss Avoidance in Video Streaming over TCP Connections*, in Proc. of IEEE Globecom 2010 Workshop, Miami, Florida, USA, 6-10 December, 2010.

[5] V. Jacobson, *Congestion Avoidance and Control*, in Proc. of ACM SIGCOMM '88, 1988, pp. 314-329.

[6] L. Brakmo and L. Peterson, *TCP Vegas: End to End Congestion Avoidance on a Global Internet*, IEEE Journal on Selected Areas in Communication, vol. 13, No. 8 (October 1995) pages 1465-1480.

[7] S. Floyd and V. Jacobson, *Random early detection gateways for congestion avoidance*, IEEE/ACM Trans. Networking, vol.1, no. 4, pp. 397-413, Aug 1993.

[8] D. Lapsley, S. Low, *Random early marking for internet congestion control*, in Proc. of GLOBECOM 1999, vol. 3, pp. 1747-1752.

[9] S. Ryu, C. Rump and C. Qiao, *Advances in Internet Congestion Control*, IEEE Communication Surveys, vol. 5, N.1 (2003).

[10] A. Bitorika, M. Robin, M. Huggard, C. Mc Goldrick, *A Comparative Study of Active Queue Management Schemes*, in Proc. of ICC2004.

[11] D. Katabi, M. Handley and C. Rohrs, *Congestion Control for High Bandwidth-Delay Product Networks*, in Proc. of ACM Sigcomm 2002.

[12] N. R. Katabi, S. S. Lam, *CYRF: a theory of window-based inucast congestion control*, IEEE/ACM Trans. Networking, April 2005.

[13] S. Floyd, *TCP and Explicit Congestion Notification*, ACM Computer Communication Review, Vol. 24 No. 5, pages 10-23, Oct. 1994.

[14] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, *Explicit window adaptation: A method to enhance TCP performance*, IEEE/ACM Trans. Networking, 10(3):338-350, June 2002.

[15] M. Savoric, *Fuzzy Explicit Window Adaptation: Using router feedback to improve TCP performance*, Technical Report TKN-04-009 July 2004.