A Survey on TCP Congestion Control Schemes in Guided Media and Unguided Media Communication

Monal Jain
Research Scholar
Department of Computer Science & Engineering,
TIT Bhopal

Deepak Singh Tomar
Asst. Professor
Department of Computer Science & Engineering,
TIT, Bhopal

Shiv Kumar Singh Tomar
Asst. Professor
Department of Computer Science & Engineering,
TIT Bhopal

ABSTRACT
Transmission Control Protocol (TCP) is a widely used end-to-end transport protocol in the Internet. This End to End delivery in wired (Guided) as well as wireless (Unguided) network improves the performance of transport layer or Transmission control Protocol (TCP) characterized by negligible random packet losses. This paper represents tentative study of TCP congestion control principles and mechanisms. Modern implementations of TCP hold four intertwined algorithms: slow start, congestion avoidance, fast retransmits, and fast recovery in addition to the standard algorithms used in common implementations of TCP. This paper describes the performance characteristics of four representative TCP schemes, namely TCP Tahoe, Reno, New Reno and Vegas under the condition of congested link capacities for wired network as well as wireless network.

General Terms
Congestion Control in hybrid networks

Keywords
TCP, Congestion, Wired, Wireless, Reno, NewReno, Tahoe, Vegas.

1. INTRODUCTION
TCP is a connection-oriented, end-to-end reliable protocol intended to fit into a layered hierarchy of protocols which support multi-network applications [1]. The TCP provides for trustworthy inter-process communication between pairs of processes in host computers attached to distinct but interconnected computer communication networks. Very slight assumptions are made as to the reliability of the communication protocols below the TCP layer. TCP assumes it can acquire a simple, potentially untrustworthy datagram service from the lower level protocols. In standard, the TCP should be able to operate above a wide spectrum of communication systems ranging from hard-wired connections to packet switched or circuit-switched networks [2]. TCP protocol is a transport layer protocol that provides a reliable and in sequence delivery of data between two nodes [3] in wired as well as wireless network. Many applications running over the Internet today rely on TCP [4]. TCP is also a defensive protocol i.e. highly responsive to network congestion. A set of mechanisms is put into place to detect occurrence of congestion and alleviate its affects, effectively preventing communication breakdown. TCP is the widely used protocol that provides trustworthy end to end in sequence data transfer. The strength of TCP is to adjust its sending rate according to the perceived congestion in the network as it is highly responsive to network congestion. The incorporated mechanism is based on the *receiver's window* concept, which is essentially a way for the receiver to share the information about the available input buffer with the sender. Fig. 1 illustrates this concept in schematic manner. When establishing a connection, the receiver informs the sender about the available buffer size for incoming packets (in this diagram the receiver's window reported initially is 8).

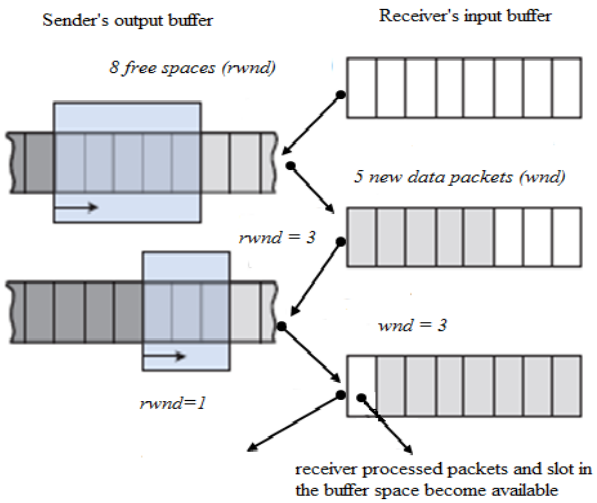

Fig. 1 The window "slides" along the sender's buffer

The sender transmits a portion (window) of prepared data packets. This portion must not exceed the receiver's window and may be smaller if the sender is not willing (or ready) to send a larger portion. In the case where the receiver is unable to process data as fast as the sender generates it, the receiver reports decreasing values of the window (3 and 1 in the example). This induces the sender to get smaller the sliding window. As a result, the whole transmission will eventually synchronize with the receiver's processing rate.

In case of a lost packet, the next packet received will return acknowledgement (ACK) of the packet received before the loss, helping the sender to recognize two identical acknowledgements. These are called duplicate ACKs and are considered as a signal of a packet loss.

2. CAUSING FACTORS OF NETWORK CONGESTION
The Congestion is a situation in Communication Networks in which too many packets are present in a part of the subnet, performance [5] degrades. In other words when too much traffic is obtainable, congestion sets in and performance degrade sharply. Factors Causing Congestion

- The input traffic rate exceeds the aptitude of the output lines.
- The routers are too slow to execute book keeping tasks.
- The router's buffer is too restricted.
- Bursty traffic
- Slow processor

When congestion occurs during packet transfers between the source and the destination, the existing routing protocols do not appear to handle it successfully. Overcrowding becomes more easily visible in large-scale transmission of traffic intensive data, such as multimedia, and the impact of packet loss on service quality is of prime concern. TCP congestion control algorithms have been originally proposed based on the assumption that congestion is the only cause for packet loss. However, wireless (Unguided) networks have higher bit error rates due to weather situation, obstacles, and multipath interferences, mobility of the wireless end-devices, and signal attenuation and fading, which may lead to packet loss. Data transfer technologies and the Internet itself have evolved, the research focus for congestion control algorithms has been changing from basic congestion to more sophisticated problems.

3. CONGESTION CONTROL SCHEME USING TCP

Congestion control and avoidance [6] is a way to deal with lost packets. In Congestion Avoidance algorithm a retransmission timer expiring or the reception of duplicate ACKs can implicitly signal the sender that a network congestion situation is taking place. The sender instantly sets its transmission window to one half of the current window size (the minimum of the congestion window and the receiver's advertised window size), but near at least two segments. If congestion was indicated by timeout, the congestion window is reset to one segment, which automatically puts the sender in Slow Start mode. If congestion was indicated by second copy of ACKs, the Fast Retransmit & Fast Recovery algorithms are invoked. The primary purpose of the TCP is to provide reliable, securable logical connection service between pairs of processes. To provide this service on the top of a less trustworthy internet communication system requires basic TCP facilities in the following areas:

- Basic Data Transfer
- Reliability
- Flow Control
- Multiplexing
- Connection
- Precedence and Security

This flow control technique of congestion control has numerous mechanisms. First, the sender buffers all data before the communication, assigning a sequence number to each buffered byte. Continuous blocks of buffered data are packetized into TCP packets that include a sequence number of the first data byte in the packet. Second, a portion (window) of the prepared packets is transmitted to the receiver using the IP protocol. Finally, the sender holds responsibility for a data block until the receiver explicitly confirms delivery of the block. As a result, the sender may finally decide that a particular unacknowledged data block has been lost and start recovery procedures (e.g., retransmit one or several packets). To acknowledge the data delivery, receiver forms an ACK packet that carries one sequence number and (optionally) several pairs of sequence numbers. A *cumulative ACK*, indicates that all data blocks having smaller sequence numbers have already been delivered.

4. TCP VARIANTS

Congestion control involves slow start and congestion avoidance phases. In order to improve the performance, several mitigation techniques have been suggested over standard TCP versions:-

A. TCP TAHOE:

Tahoe refers to the TCP congestion control algorithm which was suggested by Van Jacobson in his paper [7]. TCP is based on a principle of maintenance of packets, i.e. if the connection is running at the available bandwidth capacity then a packet is not injected into the network unless a packet is taken out. TCP implements this principle by using the acknowledgements to clock outgoing packets because an acknowledgement means that a packet was taken off the wire by receiver. It also maintains a congestion window CWD to reflect the network capacity. However there are certain issues, which require to be resolved to ensure this equilibrium.

- Determination of the offered bandwidth.
- Ensure that stability is maintained.
- How to react to congestion.

B. TCP RENO:

This Reno [8] retains the basic principle of Tahoe, such as slow starts and coarse grain re-transmit timer. It adds some intelligence over it so that lost packets are detected earlier and the pipeline is not emptied every time a packet is lost. Reno requires that it receive immediate acknowledgement whenever a segment is received. The reason behind this is that whenever we receive a duplicate acknowledgment, then this duplicate acknowledgment could have been received if the next segment in sequence likely, has been delayed in the network and segments reached there out of order or else that the packet is lost. If it receives a number of duplicate acknowledgements then that means that sufficient time has passed and even if the segment had taken a longer path, it should have gotten to the recipient by now. There is a very high probability that it was lost. So Reno suggests an algorithm called **'Fast Re- Transmit'.** Whenever we receive 3 duplicate ACK's we take it as a sign that the segment was lost, so it re-transmit the segment without waiting for timeout. Thus we manage to re-transmit the segment with the pipe almost full. Another alteration that RENO makes is in that after a packet loss, it does not decrease the congestion window to 1. Since this empty the pipe. It enters into an algorithm which we call **'Fast-Re- Transmit'.**

C. THE NEW RENO

The New Reno TCP includes a small change to the Reno algorithm at the sender, that eliminates Reno' s wait for a retransmit timer when multiple packets are lost from a window (burst error loss) [9]. The change concerns the sender's behavior during Fast Recovery when a partial ACK is received that acknowledges some but not all of the packets that were outstanding at the start of the Fast Recovery period. In TCP Reno, partial ACKs take TCP out of Fast Recovery by deflating the usable window back to the size of the congestion window. In TCP New-Reno, partial ACKs don't take TCP out of Fast Recovery. Instead, partial ACKs received during Fast Recovery are treated as an indication that the packet immediately following the acknowledged packet in the sequence space has been lost and should be retransmitted. Therefore, when multiple packets are missing from a single window of data, New-Reno can recover without a retransmission timeout, retransmitting 1 lost packet per round-trip time until all of the lost packets from the window have been retransmitted. TCP New-Reno remains in Fast Recovery until all of the data outstanding when Fast Recovery was initiated has been acknowledged [2]. The TCP New-Reno version can cover from multiple losses, and is therefore more suited than TCP Reno to the mobile wireless environment, where packet losses may occur in bursts. A major drawback of TCP New-Reno is that the sender retransmits only one packet per Round Trip Time (RTT)

D. TCP WESTWOODNR

TCP Westwood New Reno (NR) [10] is a sender-side modification of the TCP congestion window algorithm that improves upon the performance of TCP New-Reno in wired as

well as wireless networks (in fact, there are two variants of TCP-Westwood, one is based on TCP Reno and other is based on TCP New-Reno). The improvement is most significant in wireless networks with lossy links, since TCP WestwoodNR relies on end to end bandwidth estimation to discriminate the cause of packet loss (congestion or wireless channel effect), which is a major problem in TCP New-Reno. The key idea of TCP WestwoodNR is to exploit TCP acknowledgement packets to derive rather sophisticated measurements as follows:

1. The source performs an end-to end estimate of the bandwidth available along a TCP connection by measuring and averaging the rate of returning ACKs.
2. After a congestion episode (i.e. the source receives three duplicate ACKs or a timeout) the source uses the measured bandwidth to properly set the congestion window and the slow start threshold, starting a procedure that is called fast recovery.

E. TCP VEGAS

The TCP Vegas [11] extend Reno's retransmission mechanisms as follows. First, Vegas read and records the system clock each time a segment is sent. When an ACK arrives, Vegas read the clock again and do the RTT calculation using this time and the timestamp recorded for the relevant segment. When a duplicate ACK is received, Vegas checks to note if the difference between the current time and the timestamp recorded for the relevant segment is greater than the timeout value. If it is, then Vegas retransmits the segment without having to wait for *n* (3) duplicate ACKs. In many cases, losses are either so great or the window so small that the sender will never receive three duplicate ACKs, and therefore, Reno would have to rely on the timeout mentioned. When a non-duplicate ACK is received, if it is the first or second one after retransmission, Vegas again checks to examine the time interval since the segment was sent is larger than the timeout value. If it is, then Vegas retransmit the segment. This will catch any other segment that may have been lost previous to the retransmission without having to wait for a duplicate ACK. In other words, Vegas treats the receipt of certain ACKs as a trigger to check if a timeout should happen. It still contains Reno's coarse-grained timeout code in case these mechanisms fail to recognize a lost segment. Notice that the congestion window should only be reduced due to losses that happened at the current sending rate, and not due to losses that happened at an earlier, higher rate. In Reno, it is possible to decrease the congestion window more than once for losses that occurred during one RTT interval. In contrast, Vegas only decrease the congestion window if the retransmitted segment was previously sent after the last shrink. Any losses that happened before the last window decrease do not imply that the network is congested for the current congestion window size, and therefore, does not mean that it should be decreased again. This change is needed because Vegas detect losses much sooner than Reno.

F. TCP SACK

SACK algorithm [12] allows a TCP receiver to acknowledge out-of order segments selectively rather than cumulatively by acknowledging the last correctly in order received part. The receiver acknowledges packets received out of order and the sender then retransmits only the missing data segments instead of sending all unacknowledged segments. TCP Reno with SACK behaves similarly to TCP Tahoe and TCP Reno, which are strong in case of out of order packet arrivals. However, TCP with SACK helps improve performance in case of multiple packet losses. During the fast recovery stage, SACK maintains a variable called *pipe* that represents the estimated number of outstanding packets. The sender only sends new or retransmitted data when the estimated number of packet in a router is smaller than the congestion window. The *pipe* variable is incremented by 1 when the sender either sends a new segment or retransmits an old one. It is decremented by one when the sender receives the duplicate ACK with a SACK option.

5. PREVIOUSLY PROPOSED SCHEMES

This paper [13] proposes a new TCP congestion control scheme suitable for wireless as well as wired networks and is intended to distinguish congestion losses from error losses. The proposed arrangement is based on using the reserved bits of the TCP header to indicate whether the established connection is over a wired or a wireless link. Moreover, the scheme harnesses the SNR (Signal-to-Noise) ratio to detect the reliability of the link and decide whether to reduce packet burst or retransmit a timed-out packet. In brief, when a timeout occurs, the TCP protocol checks the nature of the link, if it is wired, then packet loss is due to congestion so the classical slow-start congestion control algorithm is executed to decrease the size of the congestion window. On the other hand, if the link is wireless and SNR ratio is less than 5dB, then packet loss is due to error and thus the timed-out packet is retransmitted leaving the congestion window intact. If the link is wireless and SNR ratio is greater than 5dB, then packet loss is due to congestion so the size of the congestion window is reduced to slow down the burst of packets.

In this paper [14], we propose modifications to the basic TCP congestion control algorithm so that its performance is enhanced in wireless networks. In particular, algorithm refines the multiplicative decrease algorithm of TCP NewReno. We are using various statistical counters to track the frequencies of the occurrences of timeouts and 3-dupacks. Different ratios of these counter values are then used to differentiate a congestion event from a non-congestion incident. We are also tracking the time difference between two consecutive timeouts to figure out whether timeouts are caused by network congestion or random bit errors. The proposed scheme shows better performance than any other TCP variants in the wireless networks.

In this paper [15] we present the competent design of a novel Explicit Wireless Congestion Control Protocol (EWCCP) for multi hop wireless network. The protocol achieve both proficient and fair allocation of bandwidth among flows by using 1) Explicit multi bit congestion feedback from routers, which is computed based on the coordinated congestion information and 2) an explicit signaling protocol to coordinate flows that content for the shared wireless channel. EWCCP gains fine-grain and is robust to the dynamics of the wireless channel. WWCCP stabilizes at a smaller but more optional sending window size as compared to TCP and achieve low buffer occupation and low delay.

In this paper [16], we study TCP in-cast congestion may severely degrade their performances by increasing response time also we study among TCP throughput, round trip time (RTT) and receive window. Our idea is to design an ICTCP (In cast congestion Control for TCP) scheme at the receiver side. Our method adjusts TCP receive window proactively before packet losses occur. The implementation and techniques demonstrate that we achieve al-most zero timeout and high good put for TCP in cast. In this paper, we discuss a cross layer congestion control technique of TCP and MAC in wireless networks also provides avoidance of system control congestion.

In this paper [17] proposed a novel framework termed semi-TC. Semi-TCP jointly considers the efficiency of congestion control and the functionalities of a transmission control protocol. An approach to attempt TCP's disability to obtain the exact knowledge of network congestion is to use hop-by-hop congestion control instead of end-to-end congestion control. In this case, semi-TCP suggests decoupling congestion control from TCP and moving it down to lower layers, and only its

trustworthiness control task is retained. With semi-TCP, the TCP congestion window is no longer used to regulate packet sending rate so that the throughput will not be limited by the roundtrip time and the performance can be further improved. Besides, with the hop-by-hop congestion control, the congestion control effectiveness will not rely on the availability of path connectivity.

This paper [18] reviews five approaches to TCP congestion control and review their implementations based on four techniques of managing the send window namely slow start, dynamic window sizing, fast retransmit and fast recovery. It is structured as follows; Section 2 describes five approaches to TCP congestion control for wireless networks including characteristics, algorithms & assumptions.

In this paper [19], the TCP enhancement proposed a new TCP algorithm called TCP Hybla. The proposed TCP Hybla represents a promising solution to the problem of performance disparity in heterogeneous networks due to different RTTs. Unlike many other proposals, this new algorithm is based on an analytical study of the congestion window evolution in time as a function of connections delays. After having modified the standard congestion control algorithm in accordance with the suggestions arising from the analysis, we also adopted the SACK option, timestamps and packet spacing to decrease the impact of multiple losses, unsuitable timeouts and burstiness.

In this paper [20] we propose a new approach that enables TCP, in conjunction with Explicit Congestion Control (ECN), to operate seamlessly over heterogeneous networks. The approach combines three key ideas: First, it uses ECN as the common end-to-end signaling mechanism for conveying congestion information from both wired and wireless links; second, marking for the wireless connection is performed using a load-based marking algorithm, where the marking possibility is a function of the aggregate utilization; third, load based marking algorithm dynamically adapts to varying traffic and load conditions in order to achieve an average packet delay over the wireless link within a target range.

6. COMPARISION BETWEEN DIFFERENT TCP

The problem with Tahoe is that it takes a complete timeout interval to detect a packet loss and in fact, in most implementations it takes even longer because of the coarse grain timeout. Also since it doesn't send instant ACK's, it sends cumulative acknowledgements, therefore it follows a 'go back n 'approach. Thus every time a packet is lost it waits for a time out and the pipeline is emptied. This offers a key cost in high band-width delay product links.

Reno performs very well over TCP when the packet losses are slight. But when we have multiple packet losses in one window then RENO doesn't perform too well and its performance is almost the same as Tahoe under conditions of high packet loss. The reason is that it can only detect single packet losses. If there is multiple packet drops then the first info about the packet loss comes when we receive the duplicate ACK's. But information about the second packet which was lost will come only after the ACK for the retransmitted first segment reaches the sender after one RTT. Also it is possible that the CWD is reduced twice for packet losses which occurred in one window. Assume we send packets 1,2,3,4,5,6,7,8,9 in order. Suppose packets 1, and 2 are lost. The ACK's generated by 2,4,5 will cause the re-transmission of 1 and the CWD is reduced to 7. Then when receive ACK for 6,7,8,9 our CWD is sufficiently large to allow to send 10,11. When the re-transmitted segment 1 reaches the receiver we get a fresh ACK and we exit fast recovery and set CWD to 4. Then we get two more ACK's for 2(due to 10,11) so once again we enter fast-retransmit and re-transmit 2 and then enter fast recovery. Thus when we leave fast recovery for the second time our window size is set to 2. Thus reduced window size twice for packets lost in one window. Another problem is that if the widow is very small when the loss occurs then we would never receive enough duplicate acknowledgements for a fast retransmit and we would have to wait for a coarse grained timeout. Thus it cannot effectively detect multiple packet losses.

New-Reno suffers from the fact that it's taking one RTT to sense each packet loss. When ACK for the first retransmitted segment is received only then can we deduce which other segment was lost.

The major problem with SACK is that currently selective acknowledgements are not provided by the receiver To implement SACK we'll need to implement selective acknowledgment which is not a very easy task.

TCP Vegas differs from the other algorithms during its slow-start period. The reason for this modification is that when a connection first starts it has no idea of the available bandwidth and it is possible that during exponential increase it over shoots the bandwidth by a big amount and thus induces congestion. Toward this end Vegas increases exponentially just every other round trip time (RTT), between that it calculates the actual sending through put to the expected and when the difference goes more than a certain threshold it exits slow start and enters the congestion avoidance phase.

7. CONCLUSION

The TCP protocol includes mechanisms, such as a trustworthy in-order delivery service guarantee, which introduce variations in delay and less control over the data flow from an application point of view. The different TCP schemes are better on different network conditions. An important network element is congestion control. The principle of congestion control is to ensure network stability and achieve a reasonably fair use of the network resources among the users. TCP is a well established protocol, which offers trustworthy transport of data and applies congestion control. The previous researches give some congestion control and avoidance schemes in wired as well as wireless network but they degrade the performance at high speed of data transmission. The recent proposal provides the idea about the new scheme with observe to TCP. It is interest to follow up on proposed changes to the protocol and to learn how to tune wired and wireless networks for optimal TCP performance, since its usage is ample spread.

8. REFRENCES

[1] Yi-Cheng Chan and Hon-Jie Lee, "A Hybrid Congestion Control for TCP over High Speed Networks", IEEE 2012 Sixth International Conference on Genetic and Evolutionary Computing, pp. 530-533, 2012.

[2] Marina Del Ray "Transmission Control Protocol, RFC-793" Defence Advanced Research Projects Agency, Arlington, Virginia, Sept-1981.

[3] Santosh Kumar, Sonam Rai, Survey on Transport Layer Protocols: TCP & UDP International Journal of Computer Applications (0975 – 8887),

[4] David Schinazi, Improving TCP latency with super-packets, Electrical Engineering and Computer Sciences, University of California at Berkeley, Technical Report No. UCB/EECS-2014-80, May 15, 2014.

[5] Ye Tian, Kai Xu, and Ansari N, "TCP in Wireless Environments: Problems and Solutions", IEEE Communications Magazine,

[6] Yi-Cheng Chan, Chia-Tai Chan, Yaw-Chung Chen and Cheng- Yuan Ho "Performance Improvement of Congestion Avoidance Mechanism for TCP Vegas", IEEE Proceedings of the 10th International Conference on Parallel and Distributed Systems (ICPADS'04), 2004.

[7] V.Jacobson, "Congestion Avoidance and Control", ACM **(**SIGCOMM) Symposium on Communication Architecture and protocols, pp-157-173, 1988.

[8] V.Jacobson, "Modified TCP Congestion Control and Avoidance Algorithms", Technical Report, 30 April 1990.

[9**]** D.D. Clark and J. Hoe.,"Start-up Dynamics of TCP's Congestion Control and Avoidance Scheme", Presentation on Internet End to End Research, Group, Technical Report, June 1995.

[10] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for enhanced transport over wireless links", Processing of the Seventh Annual International Conference on Mobile Computing and Networking, July 2001.

[11] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance", SIGCOMM, 1999.

[12] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanov, "TCP Selective Acknowledgement Options," RFC 2018, October 1996.

[13] Youssef Bassil, "TCP Congestion Control Scheme for Wireless Networks based on TCP Reserved Field and SNR Ratio" International Journal of Research and Reviews in Information Sciences (IJRRIS), ISSN: 2046-6439, Vol. 2, No. 2, June 2012.

[14] A Khurshid, M.H. Kabir, M.A.T Prodhan, "An improved TCP congestion control algorithm for wireless networks **",** IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim), pp. 382-387, 23-26 August 2011.

[15] Kun Tan, Qian Zhang, Xuemin Shen, " Congestion Comtrol in Multihop Wireless Network", IEEE Transaction on Vehicular Technology,

[16] Manjeet Kaur Bedi1, Kendar Pratap2 and Raj Kumari "A New Approach for Congestion Control In Wireless Network and analyze it for TCP" International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 1, Issue 2, October 2012.

[17] Yegui Cai, Shengming Jiang, Quansheng Guan, F Richard Yu, "Decoupling congestion control from TCP (semi-TCP) for multi-hop wireless networks", Springer EURASIP Journal on Wireless Communications and Networking, pp. 1-14, 2013.

[18] David Q. LIU, Williana Jean Baptiste, "On Approaches to Congestion Control over Wireless Networks",International Journal of Communications Network and System Sciences, pp. 169-247, 2009.

[19] Carlo Cainin,y and Rosario Firrincieli "TCP Hybla: a TCP enhancement for heterogeneous networks" International Journal of Satellite Communications And Networking, John Wiley & Sons, pp. 547–566, 2004.

[20] Vasilios A. Siris and Despina Triantafyllidou, "Seamless Congestion Control over Wired and Wireless IEEE 802.11 Networks", LNCS 3042.