Journal, Advances in Mathematical & Computational Sciences
Vol. 9 No. 1, 2021
www.mathematics-computationaljournal.info

Journal of Advances in Mathematical & Computational Sciences
An International Pan-African Multidisciplinary Journal of the SMART Research Group
International Centre for IT & Development (ICITD) USA

Available online at https://www.isteams.net/ mathematics-computationaljournal.info
DOI: dx.doi.org/10.22624/AIMS/MATHS/V9N1P4
CrossREF Member Listing - https://www.crossref.org/06members/50go-live.html

Improving TCP Performance in Heterogeneous Computing Networks

[1]Mmue, Y.E, [2]Ogbozor Charles O. & [3]Umoh, K.J
School of Science Education
Department of Computer Science Education
Enugu State College of Education (Technical)
Abakaliki Rd, GRA 400102, Enugu, Nigeria
E-mails: [1]barifirste@gmail.com; [3]kenjumbo@yahoo.com

ABSTRACT

Cloud computing option such as data centres provide resources for a broad range of services, such as web search, email, web sides etc. The internet entirely provides platform for rapid and timely information exchange among disparate array of clients and server [3]. The default method of transport in the internet is the Transmission Control Protocol (TCP), and Internet Protocol (IP)TCP and IP are separately designed and closely tied protocols that define the rules of communication between end hosts, and are the most commonly used protocol suite for data transfer in the internet [2]. TCP is considered the most important protocols underlining internet operation. An important mechanism which controls TCP sending rate and makes TCP reacts to congestion signals is the congestion control mechanism in TCP [5]. In heterogeneous cloud computing networks (HCCNs), TCP may work with links that have lossy nature (wireless networks). In this wireless cloud environment, TCP treat all packet loss as if they were due to congestion, therefore when use in networks that have lossy link, TCP reduces sending rate aggressively when there are transmission errors (non–congestion) in an uncongested network. In this paper, we present all the problems TCP exhibits in the HCCNs environments, wireless TCP applications area in HCCN, approaches to enhancing TCP performance in HCCNs are all presented.

Keyword: HCCN, Hybrid, TCP, IP, Wired, Wireless Computing.

Mmue, Y.E, Ogbozor Charles O. & Umoh, K.J (2021): Improving TCP Performance in Heterogeneous Computing Networks. Journal of Advances in Mathematical & Computational Science. Vol. 9, No. 1. Pp 39-48. DOI: dx.doi.org/10.22624/AIMS/MATHS/V9N1P4
Available online at www.isteams.net/mathematics-computationaljournal.

1. INTRODUCTION

Communication using wired, wireless or the combination of wired –wireless, networks (here, called Heterogeneous cloud Computing Networks), is the lifeblood of academic research, Enterprises, Technology, Government and individual, or group of individuals. The usage of communication technologies cut across all human endeavors omitting nothing. This has motivated aggressive researches into these communication technologies.

The first Network, the APANET that use packet snitching technology was created by the U.S department of defense. APARTNET was the precursor for the modern day internet with its backbone rules, TCP and IP drafted [3]. The communication of TCP/IP dominates today's communication in various networks from the wired backbone to the heterogeneous. Networks due to it remarkable simplicity and reliability. TCP has become the de facto standard used in most applications ranging from interactive session s such as Telnet and HTTP, to build data transfer like FTP, TCP was originally designed primarily for wired networks, in a wired network random bit error rate, a characteristic usually more pronounced in the wireless networks, is negligible, and congestion is the main cause of pocket drop. Based on the assumption that packet losses are signals of network congestion, the additive increase multiplicative decrease congestion control reaches the steady state, which reflects the protocol's efficiency in terms of throughput and link utilization. However, this assumption dose not hold when the end- to-end path also includes wireless links. Factors such as high bit error rate (BER), unstable channel characteristics, and user mobility may all contribute to packet losses.

Many research studies have shown that the traditional standard TCP performs pooly in a wireless environment due to it inability to distinguish packet losses caused by network congestion from those attributed to transmission errors. TCP has three critical features in the traditional model, which include:

a) TCP is a reliable protocol that guarantees delivery of individual packets from the source to the destination. This is accomplished through the use of an acknowledgement (ACK) system where the receiving systems (receiver) informs the sender that it has received the sent packet successfully.
b) (b)TCP provides error checking: Each packet contains a checksum that the recipient uses to ensure that the data was not corrupted while in transit. If the checksum does not match the data, receiver ask the sender to retransmit the packet but on the security perspective TCP error- checking functionality does not provide any security against malicious tempering , it merely ensures that the data was not corrupted accidentally while in transit.
c) TCP is connection oriented: TCP uses a session establishment and teardown algorithm that creates dedicated channels of communication between two processes.

TCP packet is associated with a sequence number and only successfully receive in-order packet is acknowledge to the sender by the receiver, by sending corresponding packet ACK with sequence NUM of the next expected packet. On the other hand, packet loss or reception of out-of- order packet indicates failure. To eradicates such failures, TCP implements flow control and congestion control algorithms based on the sliding window and Additive Increase Multiplicative Decrease (AIMD) [2] algorithms

Computing networks have advanced greatly in the past few decades. Packet switching computing technologies have merged the traditional voice and data networks together into converge and integrated multimedia networks. The all IP wired and wireless hyboid network is to a large extended become a reality. In the scenario of the present day, TCP/IP need to depart from it's traditionally wired network oriented design and evolve to handle the challenges introduced by the wireless link of the HCCN network. Many challenges introduced by TCP used in HCCN need solution. Some of these problems are presented below.

2. CHALLENGES IN HCCN TCP USE

Advances in wireless and hybrid technologies and ever increasing user's demands, the IP protocol suit need to extends it capacity to encompass other network option. At present, almost all networks are heterogeneous by connection, implying that the communication path from link to link (sender –to-receiver ends) are both wired and wireless links. And that back bone-TCP that constitute communication path exhibit weakness in HCCN environment these weaknesses, emanate the characteristics of the wireless links as compared with the wired links. These problems show up in more of these ways: inefficiency in network resource utilization, throughput degradation and excessive interruption of data transfer [3,4].

For reasons of ease of discussion, the challenges of TCP used in HCCN can be grouped as frequent route changes and long propagation delay, ***ACK*** compression, Wireless transmission medium, current TCP design assumption that all packet losses are due to congestion and link asymmetry.

2.1 Frequent Route Changes and Long Propagation Delay

In different types of wireless networks this pose many typical challenges peculiar to their nature. Handoff is an inevitable process in mobile wireless communications due to user mobility as well as limited radio frequency coverage of wireless device, either base stations or terminals. In HCCN ad hoc network because of the structure less link and high mode mobility of the network, frequent route changes and network, partitions also present a great challenge to TCP performance. In a satellite network, long propagation delay makes long range transmission inefficient.

2.2 ACK Compression:

This problem further increase the challenge of link asymmetry that will be discussed latter. TCP congestion control is self- clocking. In essence, arrivals of ACKs at the sender trigger the sending of new packets and advancing of the congestion window. However, the queuing in the reverse path of a TCP flow can cause an instantaneous arrival of successive ACKs at the sender end. This is often referred to as ACK compression. ACK compression do breaks TCP'S self – clocking [3], and cause long bursts of packet Transmission in the forward direction, and hence possible congestive packet losses. Suppose several ACKs have been lost during two successfully received ACKs, the sender will send out the number of packets indicated by the newly received ACK back to back. This burdens the forward path with an instant load and exacerbates the forward path condition. This can be devastating to some TCP schemes, which control the sending rate based on inter – arrival intervals of the returning ACKS. Usually, such schemes assume that short intervals between successive ACKs at the sender end imply large network capacity, since it is assumed that the ACK stream on the reverse path preserves the inter packet depression on the forward paths, an assumption that only holds for ideal reverse paths. The TCP self – clocking mechanism would therefore be affected, the sender would make incorrect judgment of the forward path.

2.3 Wireless Transmission Medium of HCCN Links

Fiber optic is the backbone of the wired links and the copper cable of wired access network, the wireless links use the open air as the transmission medium and are subject to many uncontrollable quality affecting factors such as weather conditions, urban obstacles, multipath, interferences, large moving objects and mobility of wireless end devices. Consequence upon this, wireless links exhibit much higher BER than wired links in addition, limitations of radio coverage and user mobility facilitate frequent hand offs leading to temporal disconnections and re- connections between the communicating end hosts during a communication session. It is vital to note here that a short disconnection can stall the TCP transmission for much longer period. This effect is demonstrated in [3].

2.4 Current TCP Design Assumption that all Packet Losses are due to Congestion

The traditional standard TCP like Reno cannot handle the high BER and frequent disconnections effectively, since all packet losses are inferred to be the outcome of network congestion in standard TCP, random packet loss caused by the high BER of the wireless link would mistakenly trigger the TCP sender to reduce its sending rate unnecessarily. The fast retransmit and fast recovery algorithms introduced by TCP Reno can recover from sporadic random packet losses fairly quickly such losses only occur once within a round trip time (RTT). However, noises and other factors in the wireless environment usually cause random bit errors to occur in short bursts, hence resulting to a higher probability of multiple random packet losses within one RTT. Also, multiple unsuccessful retransmissions within one RTT would trigger the retransmission timer to exponentially back off. Therefore, for instance, two random packet losses within an RTT would cause the TCP sender to stall its transmission for a period of about 1s. the inability of the standard TCP protocol to distinguish between congestive and random packets loss contributes to a drastic decrease of TCP throughput.

3. TECHNIQUES TO ENHANCING TCP PERFORMANCE IN HCCN ENVIRONMENTS

From the implementation point of view a combined design oriented TCP algorithms (wired- wireless TCP) suitable for heterogeneous environment, can be designed either in Link layer, End –to- End, and or as proactive, and reachive congestion control. The understanding is necessitated based on the significant differences in characteristics between the wired and wireless links in HCCN environments. Also, it will be proper to note that approaches exist, that are basically wired approaches or wireless approaches not subject for discussion in this paper, for more information on those ones, see [5,6].

3.1 Link Layer:

The aim of this approach is to completely hide the errors that occur in the link so TCP will be unaware of them and hence it will not reduce its transmission rate as a reaction to those errors. Link layer approaches are used for HCCN environments and can be located at the base- station, which connects the heterogeneous network with the wireless links just before the receiver. They monitor the packets that pass the base station from one end to another, keep records, and may sometimes keep copies of the packets sent and a transmission timeout for each packet. When the wireless link drops a packet, a timeout will either occur or duplicate acknowledgements (DUPACK) will be received at the base station. The base station then resends the lost packets and suppresses the DUPACK at the base station so TCP does not notice the drop and hence will not need to reduce its transmission rate [5,6].

Merit of the Approach

The approach has a good feature, as it preserves the end – to – end semantics of TCP since it does not break the connection.

Problem of the Approach

Sometimes this method cannot completely hide errors from the TCP sender. For example when a mechanism like snoop dropped packet but the packet is dropped again due to high error rates and then the TCP Timeout for this packet occurs before snoop has a chance to resend it again. According to [5] this could happen because of the mismatch between the TCP and snoop retransmission timeout mechanisms (RTO). In principle, the snoop RTO should be shorter than the TCP RTO but this is not aways the case. Moreover, sometimes snoop's aggressive retransmission may cause congestion at the base station which may reduce the link utilization [7,6]. Example of this approach category, the M-TCP and I-TCP [9,8].

3.2 **End –to –End**

Generally, the end-to-end scheme try to deal with the problem at the end points of the connection (sender and receiver ends) and do not expect help from the network so they look at the network as a black box [5]. In the end-to-end approach, only the end hosts participate in flow control. The receiver provides feedback reflecting the network condition and the sender makes decision for congestion control. In the end-to-end approach, the ability to accurately probe for the available bandwidth is the key to better performance, which is still a great challenge [3].

Merit of the approach

The main advantage is that it does not add overhead to the network.

Demerit of the approach:

Most often, some proposed scheme from this category use some sort of indirect feedback from the network.

3.3 Proactive Congestion Control

In Proactive Congestion Control, the sender attempts to adjust the congestion window proactively to an optimal rate according to the information collected via feedback which can be translated to an indication of the network condition. By so doing, the sender reacts intelligently to the network condition or the cause of the packet loss, due to either congestion or random errors and therefore prevent the network from entering an undesired state [10,11]. Different strategies can be employed in the design to provide the sender with the explicit network condition.

3.3.1 TCP–*Vegas*

Estimates backlogged packets in the buffer bottleneck link. Vegas select the minimal RTT as a reference to derive the optimal throughput the network can accommodate. It also records the actual sending rate at the sender during transmission to derive the actual throughput. The difference between the optimal throughput and the actual sending rate can be used to derive the amount of backlogged data in the network [10,3]. It then sets two thresholds corresponding to two network stages, one of which represents two little backlogged data. For the former stage, *Vegas* increase the congestion window linearly. In doing so, *Vegas* tries to stabilize the network congestion state around the optimal point by proactively adjusting the congestion window without a dramatic change in the congestion window. TCP-*Vegas* [12,13] is a modification to the congestion control mechanism in standard. It aims to reduce the congestion losses and to increase TCP throughput by predicting the available capacity on the link and trying not to exceed it. According to the Vegas authors [11], Vegas has increase the throughput of TCP up to 70 percent more than older implementations of TCP. [TCP–Taloc and TCP-Reno]. Also Vegas has reduced the losses in the link up to 50 percent [11].

TCP-Vegas introduces changes to TCP in four areas [5] as follows:

Timeout Computation: The authors of Vegas have noticed from experiment over the internet that the timing mechanism used in previous implementions of TCP is not accurate and that computing round trip time (RTT) using current timing mechanism has given higher RTT estimations. This makes TCP take up to three times longer to recover from losses. A new mechanism has been introduced based on using a time stamp for each packet and computing the RTT by comparing packet's time stamp with its ACK time stamp. This way, a more accurate retransmission timeout can be computed.

Retransmission of Lost Packets: TCP –Vegas introduces a new retransmission mechanism by changing the way TCP responds to DUPACK. TCP needs to receive three DUPACKS before it retransmits the Lost Packet. However, when *Vegas* receives the first DUPACK for a segment, it compares the time stamp with current time. If the difference is more than the computed timeout then it triggers retransmission without waiting for more DUPACK to come.

The other area in which TCP-*Vegas* provides changes is in avoidance: TCP-Vegas has made dramatic changes to the congestion avoidance mechanism used in TCP by making TCP increase/decrease the sending rate, not based on packet drops as in TCP but based on prediction of available link bandwidth.

∴ Vegas estimates an expected throughout and an actual throughput for the connection. The expected throughput is computed using the current window size and the minimum RTT seen so far. The actual throughput is computed using the current window size and last RTT reading.

Then Vegas compares the expected throughput and the actual throughput and update the sender window according to the comparison result as follows: if the actual throughput is less than the expected one then TCP is unable to utilize the link bacause there is congestion and hence it should decrease the window size. On the other hand, if the actual throughput becomes closer to the expected throughput then it is save to increase the window size. The increase and decrease in the window size is linear unlike TCP which uses JACOBSON'S AIMD [14] mechanism (additive increase multiplicative decrease) to update the congestion window. The *Vegas* algorithm is expected to prevent congestion from occurring, and hence reduce congestion drops drastically.

Slow Start: In *Vegas*, the congestion predictor explained above is added to the slow start mechanism. Another modification *Vegas* make to slow start is that the update of window size during slow start is not done every RTT; instead it takes two RTTs before increasing the window size. This is done to give the algorithm chance to measure the actual throughput between updating window size. The review in [15] shows the modifications Vegas did to TCP. Their results show that the new retransmission technique has improved the performance noticeably because it was able to avoid timeouts during multiple packet drops from the same window. It does this by performing retransmission when it new timeout mechnism expires even before receiving DUPACK.

3.3.2 TCP-*Veno*:

This applies changes to Vegas congestion predictor in order to differentiate between congestive states [16] and non congestive state of the connection. If a packet drop occurs during a congestive state then it is considered congestion drop otherwise it is considered transmission drop.

TCP-*Veno* estimates the number of packets buffered in the network and if this number exceeds a predefined threshold (3 in this case) then the system enters congestive state. It uses *Vegas* congestion predictors to estimate buffered packets, and instead of updating the congestion window based on this information like Vegas, it uses it to differentiate between errors and uses TCP AIMD to update the congestion window. The other changes **TCP-*Veno*** proposes is to reduce the rate at which the congestion window increases during the congestive state. So instead of increasing the congestion window every RTT, the window is increased every other RTT if the system is in the congestive state [16].
Merit: It importantly cuts the congestion window even for transmission errors by a fixed factor of 4/5 [16] which may reduce the effect of poor discrimination ability. Also, the authors in [16] indicated noticeable improvement (up to 80%) for TCP-Veno over TCP-Reno in different scenario.

TCP-*Westwood* [17]: it is a rate-based end–to-end approach in which the sender estimates the available network bandwidth dynamically by measuring and averaging the rate of returning ***ACKs***. Under the assumption of an ideal error-free and congestion-free reverse path, the inter –ACK gap reflects the available network resource. Westwood deploys an available bandwidth measurement module at the sender side based on the interval of returning ACKs. It calculates the explicit available bandwidth and uses it to guide the sending rate. When TCP-westwood determines that the link is congested after having 3 DUPACKS, it sets the slow start threshold to reflect its estimated bandwidth- delay product. TCP-*westwood* claims improved performance over **TCP-*Reno*** and *SACK* while achieving fairness and friendliness. The end-to-end approach maintains the network layer structure and requires minimum modification at end hosts, and in some cases also the routers.

Merit: The authors of TCP-westwood [20] reported an improvement in TCP performance, especially over networks suffering from transmission errors like heterogeneous networks. This improvement has been confirmed by [21]. Also, the experinmental results in [20] and [21] showed that TCP-westwood has maintained fair sharing of bandwidth.

TCP – *Jersey*: The authors [17] had suggested using the estimated bandwidth instead of errors to tell TCP when to decrease sender window size, which is an idea similar to Westwood but with a different implementation. The available bandwidth is estimated based on the rate of arrival ACKs. High ACK arrival rate means packets can get to the other end fast and hence high network capacity. Moreover, in this approach the nodes in the middle should be able to mark packets when congestion is expected in order to notify sender. Hence, this method is a combination of TCP-Westwood and Explicit congestion notification (ECN) error discriminator except that it differs in some implementation details in both cases.

Problems with Proactive Congestion Control Schemes;
Various schemes in proactive congestion control suffer from one problem or the other.

TCP – *Vegas*: Results in [15] shows that **TCP – *Vegas*** suffers from performance degradation when it coexists with versions of TCP that use AIMD mechanism like TCP Reno. This is because the AIMD mechanism is more aggressive in grabbing the link bandwidth as it keeps increasing the window size until an error occurs while Vegas tries to prevent causing drops and hence it keeps smaller window size. This indicates that the congestion predictor in Vegas sometimes has a negative impact on the performance.

TCP-*Veno*: **TCP-*Veno*** suffers from the bad performance of Vegas predictor that may lead to classify errors wrongly.

TCP-*Westwood*: This could not estimate the link capacity correctly when a non-TCP traffic exists in the reverse path [18]. This can be explained since bandwidth estimation in TCP-Westwood is based on taking the average rate of receive ACKs and since the added traffic in the reverse path could add additional delay to the received ACKs, TCP-Westwood will underestimate the available bandwidth.

TCP-*Jersey*: Like **TCP-*Westwood*** **TCP-*Jersey*** may suffer from performance degradation when coexisting with non-TCP traffic on the reverse link because it cannot estimate the link capacity correctly as the added traffic in the reverse path can delay the *ACKs*, so it will underestimate the available bandwidth.

3.4 Reactive Congestion Control

The reactive congestion control schemes include TCP-Reno and the slow start mechanism.

3.4.1 TCP-*Reno*:

The standard Reno scheme employs reactive flow control. The congestion window is adjusted based on the collective feeback of ACKs and DUPACKs generated at the receiver. TCP probes for the available bandwidth by continuously increasing the congestion window gradually until the network reaches the congestion state. In this sense, congestion is inevitable. TCP will then fall back to a much slower transmission rate, which may be unnecessary for wireless random loss. Many TCP schemes have been proposed as an enhancement to the standard Reno scheme in this reactive manner.

The fast recovery algorithm of Reno takes care of a single packet drop within one window. After one lost packet is recovered, Reno terminates the fast recovery mechanism. Due to the nature of wireless networks, correlated errors may induce multiple packet drops. Therefore, the Reno scheme would be forced to invoke multiple fast recovery procedure back and forth, slowing down recovery of the lost packet. New Reno [19] modifies the fast recovery mechanism of Reno to cope with multiple loses from a single window, this is one of the characteristics of wireless networks, where a fading channel may cause contiguous packet loss. In new Reno, the fast recovery mechanism does not terminate until multiple losses, indicated by the reception of partial *ACKs*, from one window are all received.

3.4.2 TCP SACK

This is a selective *ACK* (SACK) option for TCP, targeting the same problem Reno tries to tackle. While the feedback of Reno and new Reno is based on cumulative *ACKs*, SACK employs a selective repeat retransmission policy, it indicates a block of data that has been successfully received and queued at the receiver when packet loss occurs instead of sending partial ACK as in new Reno. Hence, the sender has better knowledge about the exact number of packets that has been lost rather than limited knowledge about the loss at the left of the window edge only, as in the standard *ACK* scheme, *SACK* requires modifications at the sender and receiver sides. *SACK* blocks are encoded in the TCP option field, which inherently limits the number of *SACK* blocks one *ACK* can carry.

Problems with TCP reactive scheme

TCP-Reno: TCP-new Reno cannot distinguish the cause of packet loss making the implementation an effective fast recovery algorithm impossible. TCP-SACK: As in Reno, SACK relatively responds to packet losses and hence has limited ability for congestion avoidance.

Table 1: Comparison of Various TCP Modification Schemes In HCCN

	Schemes	Need intermediate	TCP Semantic	Support for mobility	Modification requirement	Targeted application
Reactive approach	TCP-New Reno TCP-SACK	No No	End-to-end End-to-end	Low Low	Sender side Sender side	Heterogeneous Heterogeneous
Proactive approach	TCP-Vegas TCP-Veno TCP-Westwood TCP Jersey	No No No No	End-to-end End-to-end End-to-end End-to-end	Low Low High High	Sender side Sender side Sender side Router end Sender side	Heterogeneous Heterogeneous Heterogeneous Heterogeneous

5. CONCLUSION

In this paper, we focus on those schemes that improve TCP Performance in heterogeneous network environment such as End-to-End scheme. Proactive TCP schemes have attracted much attention recently. Generally, these schemes address the issue of the heterogeneous computing Network including that of the frequent random errors in the wireless links. More importantly, such schemes are able to manage and utilize the available bandwidth in HCCN more efficiently. Also, it may be noticed that all the schemes for TCP performance improvement exist with their problems. We therefore wish that a more in-depth study be taken to see a more effective way(s) of minimizing and or completely eliminating these problems with the various enhancement schemes.

REFERENCES

1. M. Chowdury, M. Zaharia, and Et al; 'Managing Data transfers in computer clusters with orchestra', SIGCOMM,2011.
2. D. Chiu and R. Jam, 'Analysis of the increase/decrease algorithms for congestion avoidance in computer networks', J.comp.net;vol.17 no.1, June 1989, pp.1-14.
3. Ye Tia, Kai Xu, and Nirwan Ansari, 'TCP in Wireless Environment: Problems and Solutions', IEEE Radio communications, March, 2005.
4. F. Lefevre and G. Vivier, "Understanding TCP's Behaviour over Wireless Links" Proc. Commun. And Vehic. Tech; 2000 scvt-200,2000, Pp.123-30.
5. Mohammed A. Alnuem, "an Extended Review of Techniques for Enhancing TCP Performance:, J. King Saud University Vo. 22 Comp. and Info. Sci; Pp. 45-61, Ruyadh (1431 h/2010).
6. H. Elaagrag; "Improving TCP performance over Mobile Networks. ACM Computing Surveys (CSUR), 34(3); 357-374, 2002.
7. Hari Balakrishnan, Venkata M; Padmanabhan, Srinivasan Seshan, and Randy H. Katz; "A Comparison of Mechanism for Improving TCP Performance Over wireless Links. "In Conference Proceeding on Applications, Technilogies, Architectures and Protocols for Computer Communications, Palo Alto, California, USA, 1996.
8. K. Brown and S. Singh. "M=TCP: TCP for Mobile Cellular Networks". ACM SIG-COMM Computer Communication Review, 27(5): 19-43, 1997.
9. A.V. Bakre and B.R. Badrinath. "Implementation and Performance Evaluation of Indirect TCP" IEEE Transaction on Computers, 46(3): 260-278, 1997.

10. L. Brakmo and L. Peterson, "TCP Vegas: End-to-end Congestion Avoidance on a Global Internet", IEEE JSAC, Vol. 13, no. 8, Oct. 1995, Pp. 1465-80.
11. C.P. Fu and S.C. Liew, "TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks", IEEE JSAC, Vol. 21, No. 2 Feb; 2004, Pp. 216-28.
12. L.S. Brakmo, S.W. Omalley, and L.L. Peterson. "TCP Vegas: New Techniques for congestion Detection and Avoidance. In Proc. Of Conference on Comm. Architectures Protocols and Applications, Pp. 24-35, London, U.K; 1994.
13. S. Biaz and N. Validya. "Distinguishing Congestion Losses from Wireless Transmission on Losses, a negative Result". In Proc. Of Seventh Internal Conf. on Comp. Comm. And Networks, Pp. 722-731,1998.
14. V. Jacobson. "Congestion Avoidance and Control". In Symposium of Proceeding on Communications Architecture and Protocols, Pp. 314-329, Standford, California, USA, 1988.
15. U. Henegartner, J. Bolliger, and T.Gross. "TCP Vegas Revisited in Proc. 19th Annual Joint Conf. IEEE Comp. and Comm. Societies, INFOCOM 2000, Vol. 3, Pp.1546-1555.
16. Fu Cheng Peng and S.C. Liew. "TCP Veno: TCP Enhancement for transmission over Wireless Access Networks". IEEE Journal on Selected areas in Communications, 21(2): 216-228, 2003.
17. K.XU, Y. Tian, and N. Ansari, "Improving TCP Performance in Integrated Wireless Communications Networks. Computer Networks: The International Journal of Computer and Telecommunications Networking, 2005.
18. S. Biaz and M. Vaidya. "Derandomizing Congestion Losses To Improve TCP Performance over Wired-Wireless Networks. IEEE/ACM Transaction in Networking, 13(3): 596-608, 2005.
19. S. Floyd and T. Henderson, "The New Reno Modification to TCP's Fast Recovery Algorithm", RFC 2582, Apr. 1999.
20. S. Mascolo, C.Casette; M.Cerla; M.Y. Sanadid and R. Wang "TCP Westwood Bandwidth Estimation for Enhance Transport over Wireless Links" In Proceedings of the 7th Annual International Conference on Mobile Computing and Networking; Rome/Italy 2001.
21. L. Grieco and S. Mascolo "Performance Evaluation and Comparison of Westwood, New Reno and Vegas TCP Congestion Control", ACM SIGCOM Computer Communication Review; 34 (2), 2004.