TCP on Large Scale Network Topologies: Performance Analysis and Issues on Real Networks' Topology Design

Konstantinos Paximadis, Vassilis Triantafillou
Anna Galanopoulou, and Pavlos Kalpakioris
Computer & Informatics Engineering Dept., Western Greece University of Applied Sciences, Antirio, 30020, Greece
email: kpaximadis@gmail.com

***Abstract* – The Transmission Control Protocol (TCP) is a traffic carrier protocol and the only one that counts for reliability. TCP incorporates a sliding window mechanism which controls the traffic flow from source to destination. The behavior of the window is of critical importance to TCP's performance. TCP operates on an end-to-end basis, based on routes provided by a routing algorithm. Reliability issues dictate the need of alternate routes serving either as backup routes or as load balancing routes. The exact role of alternate routes is defined by the network manager. We point out that large scale topologies are more close to real networks in many aspects, and so they deserve more attention. We study the TCP window's behavior for two major TCP versions and we address design issues, constraints and tradeoffs for large scale, similar to real, network topologies.**

Keywords-Transmission Control Protocol (TCP); Congestion avoidance and control; Sliding Window mechanism; Network Simulator NS2; Network topology.

I. INTRODUCTION

Nowadays, billions of people are connected to each other, usually via internet. There are two common used traffic carriers used for carrying all this traffic. The Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). TCP main goal is to reliably carry users' traffic from source to destination. As no one can guarantee links' integrity and limit possible losses or packet delays, TCP's reliability feature lies on a mechanism that can check the received data for errors and/or missing packets. TCP checks the received packets, informs the source and, if needed, requests retransmission of specific data packets.

TCP not only merits for reliable data transmission but also tries to control the flow of data in order to avoid congestion, by incorporating a sliding window mechanism. TCP's sliding window mechanism tries to serve as many users with as much traffic as possible. Sometimes the total amount of traffic posed by users may exceed networks' capacity. If this happens, long delays and high packet losses naturally occur, slowing down the network and degrading its performance.

So, there is a trade off between throughput and delay-losses which has to be judged carefully.

Last but not least, TCP counts for fairness, meaning that it treats all users the same, thus ensuring that all users get a fair share of bandwidth.

TCP operates on an end-to-end basis, transmitting on routes provided by a routing algorithm. Generally, a routing algorithm finds the best route from a source node to a destination one. However, in an Internet provider's core network serving millions of clients, reliability issues dictate the need of alternate routes. Alternate routes may be standing by, to be used in case of a failure, or can be simultaneously used with the best ones for load balancing. This is due to the network manager to decide.

Alternate routes have to be defined prior to transmission, because, when serving millions of connections and a link fails, you do not have the luxury (of time) to wait for the routing algorithm to calculate new routes. So, alternate routes must be known before the network starts transmission and must be used accordingly to the manager's decisions. Other issues to be taken care of are how to define alternate routes, how many should they be and how different from the basic route and from other alternate routes should, or can, be.

This work is organized as follows: Section II deals with the congestion avoidance basics used in TCP networks. Sections III presents some of the existing TCP versions. Section IV describes the network topology. Section V comments and discusses issues on topology design. Section VI presents the simulation results. Finally in Section VII, we present the conclusion and thoughts for future work.

II. CONGESTION AVOIDANCE BASICS

If users enter a network in an uncontrolled manner, the amount of traffic to be carried may exceed the total network capacity. In this case, the effective throughput (thus the number of packets that manage to reach their destinations with success) decreases and may approach zero. The phenomenon during which throughput declines towards zero is called congestion collapse [7].

The main goal of congestion avoidance and congestion control algorithms is to prevent a network from congestion collapse. The main mechanism evolved towards this goal is the sliding window mechanism.

As defined in [7], the congestion window is a TCP sender's estimate of the number of data packets which the network can manage to transmit towards destination, without causing congestion. In this case, we must note that flow control aims to prevent the destination's buffer from overflow and uses the so-called receiver window. Since usually the end (receiving) systems can process the delivered packets faster than the network can transmit them, it is assumed that the congestion window (and not the receiver window) is the main network load limiting factor. So, one easily can understand that the

 ISBN: 978-1-61208-596-8

behavior of the congestion window is of critical importance to TCP's performance.

III. TCP VERSIONS

TCP as an end-to-end protocol relies on information gathered at the two network ends. So the communication subnet is viewed like a black box [7]. As mentioned earlier TCP tries to avoid congestion in order to avoid congestion collapse. Another TCP main objective is to maintain fairness that is to equally divide the available network capacity among the bandwidth competing users.

A table giving the main features of the various TCP variants can be found in [7], where we also can found an evolutionary graph of various TCP versions.

TCP versions can be categorized [7] as Reactive ones, which base their decisions on detection of losses, and Proactive ones, which base their decisions on delay measurements.

A. TCP Tahoe

Proposed by V. Jacobson in [1], TCP Tahoe is based on the original TCP specification RFC 793 [8]. It consists of two mechanisms the Slow Start and the Congestion Avoidance.

The window's increase policy is triggered by the in-time reception of an ACK (acknowledgment) which probably means that the network is coping well with the current traffic, and so, naturally, traffic can be increased.

This congestion avoidance algorithm was found quite effective [1] [7]. Its only drawback is its relatively slow discovery and use of the network's capacity due to the conservative nature of the additive increase policy. Also, the combination of Slow Start and Congestion Avoidance mechanisms result in good behaviour regarding fairness [7].

B. TCP Reno

TCP Tahoe sets the congestion window equal to one upon a packet loss, why? Because it "smells" congestion and feels that the session must limit the amount of data that poses into the network. However, this policy is rather strict and can sometimes lead to major throughput degradation, punishing the users for the lost packet they experienced.

So Jacobson et al. [8] renew the Slow Start and Congestion Avoidance mechanisms to count for different congestion states of the network.

A major congestion network state is defined as the state where the network can hardly deliver any packets. In this case a decrease mechanism should be strict in an effort to quickly deal with this unwanted state.

A minor congestion network state is defined as the state where the network can and does deliver some packets. This case is triggered by a supposed loss packet, based on duplicate ACKs evidence.

In this case we prefer a less strict decrease mechanism.

This less strict mechanism is used in TCP Reno and is called Fast Recovery [7] [9].

Incorporating Slow Start, Congestion Avoidance, Fast recovery and Fast retransmit TCP Reno shows a significantly better performance and achieves higher throughputs. TCP slow start approach is also discussed in [2].

C. TCP Vegas

TCP Vegas was presented by Brakmo and Peterson in [6].

As noted in [6], TCP Vegas aims to measure and accurately control a "right" amount of extra traffic in the network. Extra traffic would not otherwise have been accepted in the network.

TCP Vegas uses a proactive mechanism in order to replace the reactive Congestion Avoidance algorithm.

As shown in [7], TCP Vegas has the advantage of achieving rate stabilization in a steady state. Rate stabilization together with the absence of unwanted oscillations of the window size, can lead to higher values of throughput.

Other TCP versions are TCP Africa [3] which is a delay-sensitive congestion avoidance approach incorporated in networks of high bandwidth delay product (BDP) and Compound TCP (CTCP) [10] which a synergy of delay-based and loss-based approach.

IV. LARGE SCALE NETWORK TOPOLOGY: WHY AND WHERE?

TCP variants have been extensively simulated. However, in many studies [3] [4] [6] [7] [10] [13], the topologies tested are somewhat small (and sometimes trivial). In [4], a four-node backbone network topology with numerous nodes attached to the four nodes is used. In [11], a larger topology is tested but with a relatively small network core, and in [13], a fat-tree network topology is used with the question of how it can be applied in data center networks.

As mentioned in [3] several modern applications such as supercomputer grids and large biological simulations often have the need to transfer data between different continents. So, naturally, data need to travel through several nodes, and, apparently, large topologies.

Also, large telecommunication and internet regional providers operate on their private networks which are usually expanded all over the country. Other large institutions with Wide Area Private Networks (such as banks) also operate over a large area.

For all the above reasons we decided to simulate TCP versions over a large scale network topology, which was first introduced in [5] and also used in other studies [12]. By large we mean a network that is close to a regional network provider's backbone network, for example, a provider operating in a medium sized country (i.e. in Europe, not China!) who can use these 19 nodes to locate a router (or a Layer-3 switch) in each of the major country's cities.

Large scale networks provide a realistic simulation environment as they are close to real ones. They have real world's characteristics and if suitably modified can match exactly real networks of providers and/or banks. Figure 1

 ISBN: 978-1-61208-596-8

shows the topology of the network used and Figure 2 shows the 32 sessions simulated.

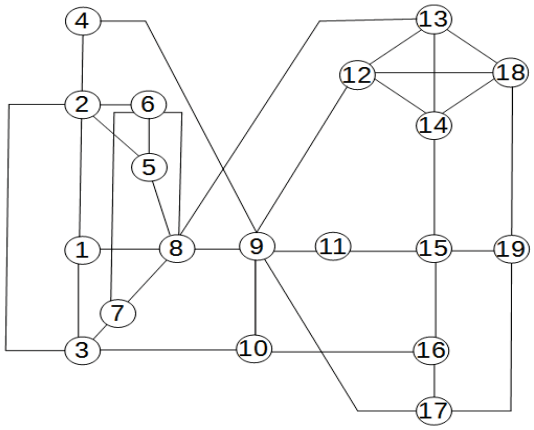

Fig. 1: The large scale network topology used

Traffic Matrix 32 sessions Source/Destination	
From node	To nodes
n1	n9, n19
n2	n18
n3	n4, n11, n17
n4	n3, n10, n13
n5	n12
n6	n14
n7	n16
n8	n15
n9	n1, n19
n10	n4, n13
n11	n3
n12	n5
n13	n4, n10, n17
n14	n6
n15	n8
n16	n7, n18
n17	n3, n13
n18	n2, n16
n19	n1, n9

Fig. 2: Network's traffic matrix

More specifically, if the topology is to be used for simulating a real network, the sessions must be modified to reflect the common case in such networks, where a central node is always the capital of the country, and all other nodes have a connection (direct or via other nodes) to it. Network reliability issues dictate the need of alternative routes for every communicating node pair.

So, a problem of defining the alternative routes arises. This problem is rather complicated as the network manager need not only to define the alternate routes but also guarantee that the network will continue to transmit in the case of more than one link failure. Moreover, the alternate routes must be chosen in a manner that, in case of a failure, they will not pose more traffic in certain links that are already heavy loaded.

V. ISSUES ON LARGE SCALE NETWORK TOPOLOGY DESIGN

A network manager in a major communications provider expanded over a whole country faces many problems. As in real life, a manager cannot start designing from a white sheet, because various constraints usually pre-exist and make the design more complex.

For example, in a network expanded in a whole country (i.e., internet provider network, bank network, universities network), usually (if not always) the major cities are hosting the major network nodes. Moreover, the country's capital usually hosts the central network node (or nodes). Although nobody suggests not do, a manager usually do not pose a network node in the middle of nowhere. So? So, there exists a somewhat predefined network topology and the only design freedom is designing the connections between the "existing" cities-nodes.

One of the first questions to be answered is which type of topology to use. To answer, one must first answer another question: What exactly a network manager expects from a topology?

As serving thousands or millions of customers and/or connections, what one mostly wants from a topology is fault tolerance. Speed and availability may wait a little. So, a large scale network topology for a major communications provider must have alternative routes assuring connectivity even in the event of a failure. By "must have" we mean that the alternative routes have to be specified and standing-by as the network operates. Why? Because in case of a failure, with millions of connections being served, the network does not have time to waste and it would be a small disaster to wait for a routing algorithm to run (again) and calculate new routes.

Another question is how many failures must the topology be able to successively overcome?

As noted before, a common practice is that the network central core node (or nodes) are usually placed in the country's capital, i.e., Athens for Greece, Rome for Italy and so on, following the well known "All roads lead to Rome".

So, usually the central network node is predefined and there exist network nodes which have to connect to the central node. How will the manager do that?

A. Geographical constraints

A known rule in backbone networks is that each network node must have at least two connections, each one connected to a different core node, obviously for reliability reasons. However, sometimes this is not easily achievable. Suppose a city is located high in the mountains or in a somehow isolated island (i.e., Kastelorizo in Greece), having only one major city in nearby distance and all others either far away or geographically difficult to reach and connect.

An alternative to this difficult situation is to use two separate links from city to city. These separate links must preferably follow a totally different route. Why? Because if placed side-by-side they are both vulnerable to the same failure if something goes wrong at their common route. So,

 ISBN: 978-1-61208-596-8

one have to connect cities which cannot (due to geographical or economical restraints) have duplicate links with other cities/nodes, with dual or triple links with the same city/node, providing that these links follow different routes. How different? Surface morphology, distance, accessibility and cost will decide.

This technique can "save lives", meaning that can make the ring topology tolerant to single, double or even triple link failures, as it acts like a whole backup network.

B. *Ring Topology*

Classical ring topology offers a reliability feature over a single failure. If a single link failure occurs, communication between all nodes is still achievable through the remaining active part of the ring. Figure 3 notes that, although for clarity reasons the ring network is shown as a rectangular one.

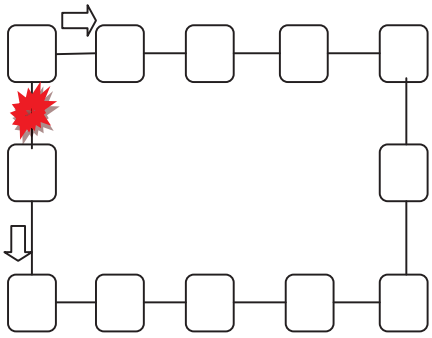

Fig. 3: A ring network experiencing a single link failure can still transmit packets.

But what if two link failures occur? Then you have a big problem, as your ring is separated in two parts, as in Figure 4.

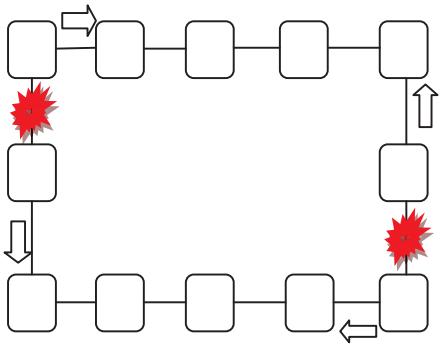

Fig. 4: A ring network experiencing a double link failure is separated in two parts.

This problem can be solved by adding a network node in the middle of the ring. This node, if connected by two other nodes, as shown in Figure 5, technically divides the original ring into two sub-rings. This topology can tolerate two simultaneous network link failures, if each one of them is in different sub-ring. If the two link failures occur in the same sub-ring, the nodes between failures are isolated from the other nodes.

Also, if one (of the two) link failure occurs on the common part of the two sub-rings, communication from every node to every other node will be still achievable.

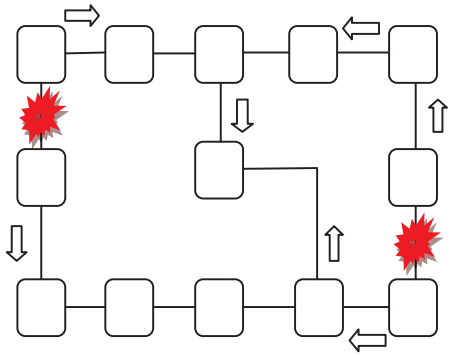

Fig. 5: A ring network divided in two internal rings.

C. *Star topology*

Naturally, capital's central node is the center of a star topology and all other cities-nodes are directly connected with it. Again, the only freedom one has is to choose the link connections. Fault tolerance reasons dictate the need of at least two links connecting each node with the central node. Also, as noted earlier in the ring topology analysis, it is a safer technique (although obviously more expensive) to use different routes for the links connecting each city-node to the central node.

So, one has to install two or more links between peripheral cities-nodes and the central node/nodes, each one of them preferably following different physical route.

D. *Final decisions*

So, finally, the topology is chosen and network links are installed. After that, the routing algorithm specifies the optimal and the alternative routes from all sources to all destinations. One big dilemma is whether the alternative routes should be used concurrently with the optimal ones, or should be used only as backup, thus stand-by and be used only in case of a failure.

The concept of using multiple routes to split traffic more efficiently over the available network capacity has been extensively explored by many researchers. One of the most analytical design called mTCP, is presented by Zhang et al in [14] and efficiently faces all problems raised.

Nowadays networks' response and speed are not a major problem. However, if it becomes (not temporarily, but for a long time), adding some high bandwidth links is generally a fair and cheap, in the long run, solution. A network manager except from failures (the frequency of which have to be examined), speed and availability must also merit for simplicity of protocols and algorithms used. So, an easy and "clean" solution would be to choose the second alternative to the big dilemma, which is to let the alternative routes standing by and send traffic to them only in case of a failure.

In this case it would helpful the alternative routes to be as "different" as possible from the optimal ones and, at the same

 ISBN: 978-1-61208-596-8

time, do not use links used by other active routes. An interesting and relatively simple approach for finding non-overlapping and disjointed routes is presented at [14].

VI. SIMULATION RESULTS

We conducted various simulations to observe and verify the behavior of the congestion window, the packet loss probability and the total throughput. We used NS2 and the 19 node network topology described in section 4. All links was set to 20 Mbps, links' propagation time was set to 10ms, and drop-tail type of queue was used. Standard NS2 packet size of 1000 bytes was used. As a means to avoid starting the window size from one in a lightly loaded network and the algorithm being slow in gaining bandwidth, we set the initial window size advertised by the receiver to various predefined values at the range from 10 to 100 packets.

Figures 6 and 7 show the window size versus time for TCP Reno and TCP Vegas respectively for initial window size of twenty (20) packets, for six (6), (2, 14, 18, 19, 24, 32) more representative of the topology, sessions. We observe that for TCP Reno the congestion window can reach large values in some flows. Which are these flows? Obviously these are flows that are not using same links with others, or in other words flows operating at a less crowded part of the network.

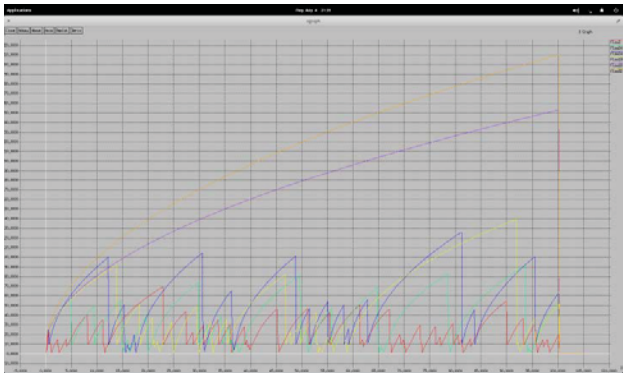

Fig. 6: The congestion window in time, 6/32 flows (TCP Reno)

Let's now focus on Flow 14 (blue colour at Figures 6, 7) which has mode n9 as source node and node n1 as destination. Thus Flow 14 transmits from a "central" node to a peripheral one. If it was a real internet provider's network, node n9 as the central node, would represent the capital of the country, and node n1 a nearby city. As Flow 14 is close to the central node, uses common routes with other sessions, and so, it cannot reach high window values. Why? Because TCP counts for fairness, and so, it divides equally the available bandwidth to the competing users on Flow's 14 links.

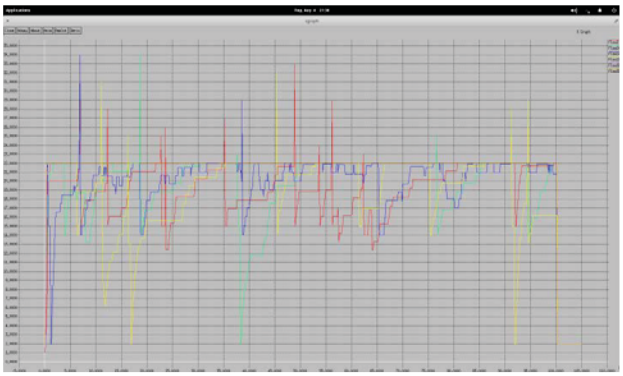

Fig. 7: The congestion window in time, 6/32 flows (TCP Vegas)

Total network statistics for TCP Reno and TCP Vegas, for all tested initial advertised window sizes follow, Tables 1 & 2.

Both TCP Reno and Vegas manage to send approximately the same amount of data packets all over the network. TCP Vegas due to its higher sensitivity manages to keep packet losses lower and so results to lower packet loss probabilities, as shown in Figure 8. One special characteristic of TCP Vegas is that it is hardly affected by the initial window size advertised by the receiver as because of its delay based mechanism it immediately senses network's capacity and adjusts the window size accordingly. TCP Vegas ends with lower packet loss probabilities for the same throughput, and, so, seems a better choice.

We observe that rising the initial advertised (by the receiver) window size beyond a mid-range value (at about 50-60) does not bring any benefits, because in this case we practically pose big loads at the beginning of network's operation. TCP congestion control reacts to these big loads and limits them quickly. So, we end up with higher packet loss probabilities, while the total gain in throughput is marginal.

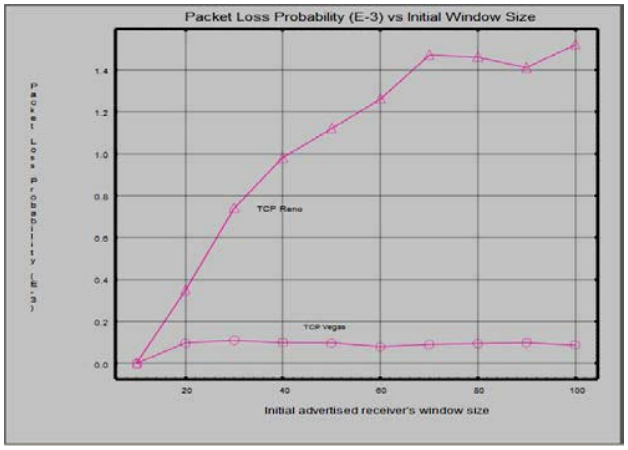

Fig. 8: Packet Loss Probability vs Initial window size for TCP Reno and TCP Vegas.

 ISBN: 978-1-61208-596-8

TABLE 1: TOTAL NETWORK STATISTICS, TCP RENO

TCP Reno					
Init WS	Sent packets	Received_ packets	Lost packets	Packet Loss Prob*10^{-3}	Average Throughput[kbps]
10	3114834	3114834	0	0	124585
20	5635813	5633828	1985	0,352212	225473
30	6904954	6899838	5116	0,740917	276289
40	7801538	7793820	7718	0,989292	311092
50	8368958	8359581	9377	1,12045	333815
60	8810732	8799548	11184	1,269361	351547
70	9016960	9003685	13275	1,472226	359587
80	9264323	9250711	13612	1,469292	370031
90	9304494	9291314	13180	1,41652	371485
100	9291425	9277248	14177	1,525815	371886

TABLE 2: TOTAL NETWORK STATISTICS, TCP VEGAS

TCP Vegas					
Init WS	Sent packets	Received_ packets	Lost packets	Packet Loss Prob*10^{-3}	Average Throughput[kbps]
10	3116546	3116546	0	0	124610
20	5730241	5729683	558	0,097378	229124
30	7177305	7176492	813	0,113274	287016
40	8093051	8092230	821	0,101445	323621
50	8717150	8716294	856	0,098197	348557
60	8883430	8882725	705	0,079361	355232
70	9070137	9069304	833	0,09184	362268
80	9237489	9236605	884	0,095697	369345
90	9220422	9219505	917	0,099453	368156
100	9240714	9239887	827	0,089495	369496

Generally speaking, it is not a good idea to let critical factors take large values in order to gain some benefits in a specific performance measure (with Higher is Better relationship). This policy, due to trade-offs, will probably lead to an equal (if not higher) degradation of another contradicting performance measure and the final result will be worse. This is a "rule" that comes true not only in computer networks but also in many aspects of engineering and life.

VII. CONCLUSION AND FUTURE WORK

We studied TCP Reno and TCP Vegas in a large scale network topology. Both versions prevented severe congestion, while TCP Vegas showed better performance.

Also, we commented on large scale network topologies. A big dilemma faced is whether to use the alternative routes concurrently with the optimal ones, or let them standing-by and use them only in case of a failure. Both options are well used in Greece major telecommunication networks. In either case, algorithms for defining the alternative routes, like the one proposed in [14], must be investigated.

If one chooses to concurrently transmit on alternative-backup routes except from the obvious overhead added, will also, most probably, face packet re-ordering problems. Packet re-ordering in TCP results in major performance degradation [13], and is another area of interest.

We are currently working on the problem of efficiently choosing, using and administrating the alternative non-overlapping backup routes in a large, close to real, network topology.

REFERENCES

[1] V. Jacobson and M. Karels, "Congestion avoidance and control," ACM SIGCOMM 88, 1988.

[2] K. Oyeyinka, et al. "TCP Window Based Congestion Control -Slow-Start Approach," Communications and Network, Vol. 3 No.2, pp 85-98, 2011.

[3] R. King, R. Baraniuk, and R. Riedi, "TCP-Africa: An Adaptive and Fair Rapid Increase Rule for Scalable TCP", Proc. IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 13-17 March 2005.

[4] H. Jamal and K. Sultan, "Performance Analysis of TCP Congestion Control Algorithms", International Journal of Computer and Communicaitons, Issue 1, Volume 2, 2008.

[5] G. Thaker and J. Cain, "Interactions Between Routing and Flow Control Algorithm" , IEEE Transactions on Communications, March 1986.

[6] L.S.Brakmo and L.L.Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet", IEEE Journal on Selected Areas in Communications, col.13, no 8, Oct. 1995.

[7] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-Host Congestion Control for TCP", IEEE Communications Surveys & Tutotrials, vol.12, no 3, 3rd quarter 2010.

[8] V. Jacobson, "Modified TCP congestion avoidance algorithm", email to the end2end list, April 1990.

[9] M. Allman, V. Paxson, and W. Stevens, " RFC2581 – TCP congestion control", RFC, 1999.

[10] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A Compound TCP Approach for High-speed and long Distance Networks", the 4th International Workshop on Protocols for Fast Long-Distance Networks (PFLDNet), 2006.

[11] H.Jamal and K. Sultan, "Performance Analysis of TCP Congestion Control Algorithms", Int. Journal of Computers and Communications, Issue. 1, vol.2, 2008.

[12] K. Paximadis and A. Vasilakos, "A Dynamic Congestion Avoidance Scheme Incorporating A-priori Information for Computer Networks", in Proc. of the 10th International Conference on Computer Communication, ICCC'90, New Delhi, India, 1990.

[13] N. Farrington, "Multipath TCP under Massive packet reording", Technical Report, University of California, San Diego.

M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, "A transport Layer Approach for Improving End-to-End Performance and Robustness Using Redyndant Paths", in ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference, pp.8-8, Berkely, CA, USA, 2004 Usenix Association.

 ISBN: 978-1-61208-596-8