TCP Optimized for Short Flows

Nitin Kartik, Department of Electrical Engineering, Stanford University

***Abstract**: TCP has been used very successfully for reliable communication for various data flows. One of the issues facing networks that support these data flows is Congestion. Current TCP Congestion Control mechanisms such as Reno and Tahoe focus on congestion avoidance based on network feedback. This is too conservative for short flows which require less than one round-trip time to communicate, and results in underutilization of available network bandwidth due to the TCP slow start problem. This paper proposes a technique to improve TCP's Congestion Control algorithms' ability to overcome the slow-start problem for short flows of data while retaining TCP Congestion Control semantics for longer flows. This proposed algorithm is analyzed theoretically and yields significantly lower latency for short flows which approaches that with no Congestion Control overhead.*

Table of Contents

1. Introduction

Modern day Internet traffic features a variety of data transfers including World-Wide Web (HTTP), File Transfer (FTP), and Remote Login (Telnet) among others. Studies dating back to Jacobson'88 [1] have concluded that the ever increasing Internet traffic inevitably causes congestion at routers, and this results in poor throughput performance of TCP, due to bandwidth wasted in packet retransmissions. In response, several schemes of TCP Congestion Control have been proposed, as summarized in Lai'01 [18] and in Fig 1.

Although the recent proliferation of file sharing services like Kazaa have gained momentum on the Internet, general data flows are dominated by HTTP traffic by sheer

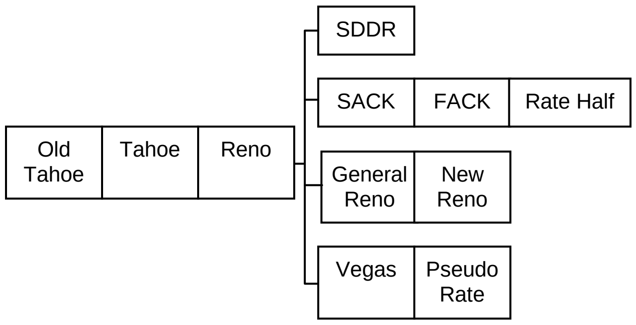

Fig 1. TCP Congestion Control Schemes, Lai'01 [18]

volume and the immense user base. The reliability of TCP upon which HTTP traffic is sent is well worth the drop in performance when compared with UDP. The average size of the typical HTTP packet including the TCP/IP header is approximately 1 KB. This is further bolstered by Lucas'97 [8] and Thompson'97 [9] who show frequency distribution of packet sizes, depicting peaks around 50 B, 500 B, and 1,500 B.

Current TCP Congestion Control algorithms include Old Tahoe (RFC793), Tahoe (RFC1122), Reno (RFC2001 [7]), SDDR (Wang'98 [11] , Wang'00 [17]), SACK (RFC2018), FACK, Rate Halving, General Reno (RFC2581), New Reno (RFC2582), Vegas (Brakmo'95 [4]), and Pseudo-Rate (Chen'99 [12]). There have been some proposals of modified TCP Congestion Control algorithms to cater for short flows, as explained in Chang'93 [3] , Kamik'00 [14] , Pradhan'00 [16] , and Xu'02 [21] .

Despite the positive aspects of these schemes, they are all based on network feedback to grow or shrink their TCP congestion control windows. They all start conservatively, and take at least one roundtrip time to grow their window sizes. This is referred to as the TCP *slow start* problem where available network bandwidth is underutilized as nodes gingerly ramp up their transmission speeds for fear of flooding the network. Ironically by the time a host uses these existing techniques to ramp up to optimal transmission speed, the entire data transfer could have completed if they did not use TCP congestion control at all.

In the following, we expose the shortcomings of existing TCP Congestion Control algorithms for TCP Short Flows, recommend a creative technique to cater for TCP Short Flows, analyze this algorithm theoretically, and draw relevant conclusions.

2. Shortcomings

Current TCP Congestion Control schemes work well for *congestion avoidance*, however they tend to underutilize the immense bandwidth that modern day networks typically have. To understand this better, we start off by defining Short Flows, then we explain why current TCP Congestion Control schemes fail to accommodate Short Flows.

2.1 Short Flows Defined

Short Flows are TCP flows which would last less than one round-trip time (RTT) without the overhead of TCP congestion control. With feedback based TCP congestion control techniques, the slow start period lasts a few RTTs and therefore results in tremendous underutilization of available (and continuously improving) network bandwidth.

Of all the data flows on the Internet, a large portion of these are comprised of data transfers of size approximately 1 KB, as explained earlier. A good example of these are web servers which service requests from various clients, and each connection lasts for the duration of one HTML web-page transfer. The HTTP protocol stipulates one TCP connection for each object (text section or inline image). Another good example is a security certificate authority which issues short-term certificates to various clients. With the average certificate being 128 bits (= 16 B) in size, the typical certificate data transfer is of the order of 0.5 KB. Yet another example of short flows is a Temporary Mobile Subscriber Identity (TMSI) authority in a GSM cellular system, which allocates 4 Byte TMSIs to various cellular phone subscriber units. The size of these data transfers is of the order of 0.25 KB including headers. These examples are summarized in Fig 2.

Flow Type	Flow Size
Web-page	1 KB
Certificate	0.5 KB
TMSI	0.25 KB

Fig 2. Typical Internet Flow Sizes

A study of the trends of Flow Sizes, Network Bandwidth, Round-Trip Times (RTTs), and Number of Flows per RTT elucidates the problem at hand. The Flow Sizes in Fig 2 are likely to remain approximately the same over time. Network Bandwidth between adjacent nodes over time is always increasing. Even though this is the case, long-distance RTTs are not likely to decrease very rapidly, since this is primarily determined by the number of hops along the route. As a result, the Number of Flows that could be sent per RTT over the Internet is always increasing over time. These trends are illustrated in Fig 3.

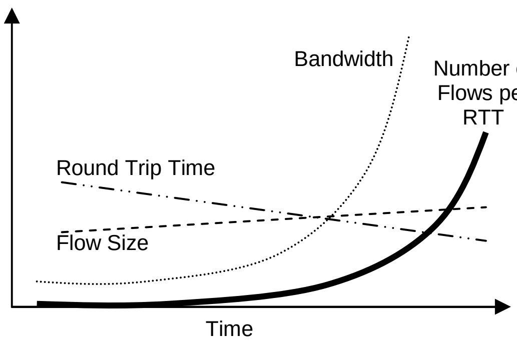

Fig 3. Trends of Bandwidth, Flow Size, and RTT over Time

Formally, we define Short Data Flows as those with a Flow Size that satisfies the following property (1).

$$\Phi = \frac{\beta.\tau}{\varphi} \geq 1 \qquad (1)$$

Where: Φ = *Num Flows per RTT*
φ = *Flow Size*
β = *Link Bandwidth*
τ = *RTT*

2.2 Canonical Example

The limitations of current TCP Congestion Control schemes are best studied by establishing a canonical example of how a typical scenario is poorly handled by existing congestion control algorithms.

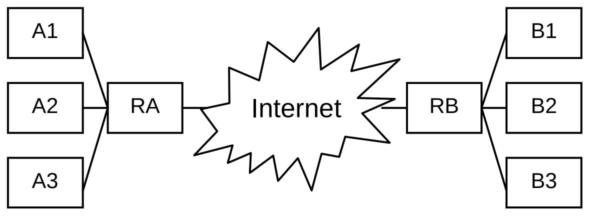

Fig 4. Canonical Data Transfer Example

Fig 4 illustrates a typical network setup. We consider the case where host A1 wants to send an HTML page over HTTP to host B2. Along the path from host A1 to B2, are routers RA and RB, as well as any other hops over the wide area Internet. For illustrative purposes, let's consider the HTML page in question to contain the following data:

- *Text Frame 1 (500 bytes)*
- *Inline Graphic 1 (300 bytes)*
- *Inline Graphic 2 (200 bytes)*
- *Text Frame 2 (100 bytes)*

The HTML page containing the above data involves four TCP flows, one for each transfer. We use the following values to evaluate this transfer:

- *200 byte TCP packets*
- *100 Kbps Link Bandwidth*
- *0.25 second RTT*

With the above values, the transfer times for the above with TCP Reno Congestion Control and with no congestion control at all are tabulated in Fig 5.

Flow	TCP-Reno	None
Text 1 (500 B)	0.5 sec (2 RTT)	0.05 sec
Text 2 (300 B)	0.5 sec (2 RTT)	0.03 sec
Graphic 1 (200 B)	0.25 sec (1 RTT)	0.02 sec
Graphic 2 (100 B)	0.25 sec (1 RTT)	0.01 sec

Fig 5. Comparing Transfer Times

These results from Fig 5 are graphed in Fig 6.

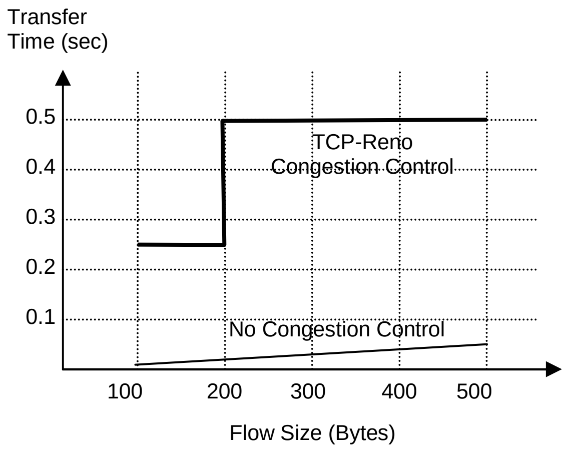

Fig 6. Shortcomings of Congestion Control for Short Flows

Therefore the problem at hand is evident – _the order of magnitude difference between the transfer times of short flows using traditional TCP Congestion schemes, and without any TCP Congestion Control overhead_. The difference in performance between feedback-based TCP Congestion Control algorithms and using no congestion control at all is evident from Fig 6. The main problem with using no congestion control however is that when the network is congested, there is no corrective action taken, and packet loss is rampant in the system.

Our proposed scheme optimizes for short flows, as well as provides a congestion control mechanism for when the network is genuinely congested.

3. Proposed Scheme

The proposed scheme improves on the Aggregate TCP (ATCP) mechanism proposed in Pradhan'00 [16] . The main idea is to start TCP flows optimistically for an initial amount of time T_{opt} (in milliseconds), and then resort back to traditional TCP Congestion Control mechanisms if the flow continues beyond T_{opt} time. We first consider the criteria that the new proposal must satisfy, and then explain the mechanism in context of the entire system, followed by the details of the handshaking between the clients and the Router, and also explain how the Router shares TCP state across connections it subtends.

3.1 Criteria

When considering a new proposal for a TCP Congestion Control mechanism, the following criteria must be satisfied by the new proposal:

- _Fairness_: The new proposal must not be so aggressive that it causes other clients running traditional forms of TCP Congestion Control to be forced to reduce their window sizes below their fair share of the flow.
- _Performance_: For the affected flows, the new proposal must provide performance no worse than traditional TCP Congestion Control schemes.
- _Interoperability_: The new proposal must be transparent to client applications so it can be deployed with ease in a progressive fashion.

3.2 Intuition Behind the Design

After considering the shortcomings of traditional TCP Congestion Control mechanisms, and understanding the criteria important for any proposed scheme, it is in order to explain the intuition and rationale behind the proposed scheme.

As illustrated in Fig 6, there exists an order-of-magnitude difference in the transfer times of short flows when traditional TCP Congestion Control is used, and when no TCP Congestion Control is used at all. That is to say that for short flows, we are better of not using any TCP Congestion Control at all. When it comes to long flows however, namely the ones that multiple RTTs to transfer, not using any form of TCP Congestion Control introduces the risk of flooding the network with this flow, and starving other well-behaved flows of their fair share of network bandwidth.

If we had ideal knowledge of the entire system, we would first determine if a flow were a short flow or not, and then use NO TCP Congestion Control algorithms for short flows, and use regular TCP Congestion Control algorithms for the longer flows. In the real world however we do not have perfect knowledge of whether a flow is a short flow or not. Internet data flows follow a *Heavy Tailed* Distribution meaning that:

- the majority of flows in the Internet are short flows as explained at the outset, whereas
- the majority of the data that is communicated over the Internet is through long flows.

Since the majority of flows over the Internet are short flows, best option under practical considerations is to start off optimistically, in the expectation that every flow is a short flow, and start off using no TCP Congestion Control overhead. If a flow turns out to be a long flow, then the system must resume regular TCP Congestion Control semantics to avoid congestion in the network.

This is the fundamental idea behind this design: Start off optimistically and not use TCP Congestion Control for the initial part of a flow. If a flow is prolonged, it is considered to be a longer flow, and regular TCP Congestion Control semantics are restored.

3.3 System Overview

The optimized TCP Congestion Control scheme is designed to reside in a router within the network that all clients use to access external hosts. This could be a corporate firewall, Internet Service Provider's (ISP) gateway, or even a small scale WAN router. This is illustrated in Fig 7.

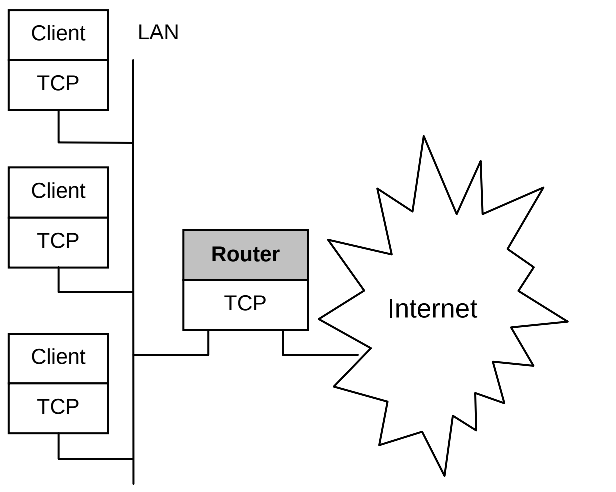

Fig 7. System Overview of Optimized Scheme

This router intercepts all TCP packets bound from the local LAN interface to the external interfaces. The router carries out the necessary handshaking so the clients are completely unaware that the optimized congestion control scheme is in place.

3.4 Handshaking

When a client initiates a TCP flow, the router starts sending *Phantom* TCP Acknowledgements back to the local client so the client quickly increases its TCP window size without waiting for multiple RTTs. After the T_{opt} time passes if this flow is still open, the Router stops sending *Phantom* Acks back to the local client, and lets the standard TCP Congestion Control mechanisms control the window size using the timing of the real Acks.

The Router keeps track of the following information on a per-flow basis:

- T_{opt} timer
- Flow Start time
- Outbound Queue
- Inbound Queue
- Real Ack Received Flag

With this optimized scheme, there is a concern that the initial optimistic approach of sending packets without waiting for real Acks could cause congestion in the network due to clients sending too aggressively. Therefore if the Router detects that an Outbound Queue overflow is imminent, it deliberately drops packets sent out by the client so the client does not speed up too fast.

Once the Router starts receiving real Acks from a remote host destined for a local client, it checks the transaction id. If this transaction id has already been *Phantom* Ack'd back to the local client, the Router discards this inbound real Ack. If the transaction id has not yet been *Phantom* Ack'd back to the local client, the Router forwards this inbound real Ack to the local client.

If the router starts receiving real acknowledgements before a flow's T_{opt} timer expires, it immediately stops sending *Phantom* Acks to the local client. It then checks every real Ack's transaction id, and as outlined above selectively forwards these back to the local client.

3.5 Autocorrection of the T_{opt} Timer

The T_{opt} timer is initialized to 500 milliseconds. Depending on the network characteristics however, this value should adapt to optimize performance. This value is autococorrected by the Router using a long-term running average (LTRA) mechanism. The length of the LTRA is L_{opt}, which is set to 10,000. The following describes how this value is updated.

Whenever a local client closes a TCP connection through the Router, the Router determines if it was a Short Flow or not. This is determined by checking if the flow received a real Ack before the local client closed the connection. If the flow was closed by the local client before receiving a real Ack, then it is considered a Short Flow, and its duration is factored into the LTRA T_{opt} calculation using Equation (2):

$$T_{opt}^{new} = \frac{\sum_{j=1}^{L_{opt}} T_{opt}(j)}{L_{opt}} \qquad (2)$$

where: $T_{opt}(j)$ = jth sample of T_{opt}

After the above calculation is performed and T_{opt} is updated, $T_{opt}(1)$ is purged, and the Topt(j) are shifed as follows:

$$T_{opt}(j-1) \leftarrow T_{opt}(j) \quad \text{for j = 2 .. } L_{opt} \qquad (3)$$

$$T_{opt}(L_{opt}) \leftarrow T_{opt}^{new} \qquad (4)$$

This ensures that T_{opt} is sensitive to changes in network dynamics.

3.6 Shared TCP State

The optimization is extended to another level by sharing TCP state across connections subtended by the Router, as explained below.

When the router terminates a TCP connection with a remote host, it records the optimal bandwidth achieved between its local subnet and the remote subnet. When a new connection is requested of the router with this same remote subnet, it already knows the optimal bandwidth, and can pace its Phantom Acks responsibly without becoming too aggressive. This further ensures fairness in the network.

3.7 Router Complexity

Considering all the above functionality in the router, its design must carefully take into account the various forms of book-keeping it performs.

In particular, the router must keep track of the following on a per-flow basis:

- Input and Output queues

- Source and Destination addresses
- Connection Start Time
- Real Ack Received Flag

In addition, the router must keep track of the following information on a global (for all flows) basis:

- T_{opt} Values 1 .. L_{opt}
- Optimal bandwidth for each destination subnet, in terms of frequency of Phantom Acks per second.

Although this arguably makes the router design more complicated, it simplifies the task of deployment, since existing TCP clients can be accommodated with this scheme. Alternate schemes that involve modification to the TCP clients are viable in the long term, but are not considered herein due to the obstacles in deployment they will face. Even then, they will only cater for modified clients with the new protocol, and not older legacy clients.

3.8 Multiple Paths to Destination

One possible scenario that deserves special treatment under this design approach is the situation where a local client has multiple paths to the destination, not necessarily involving this router. In this case, the system would not be able to take advantage of this proposed scheme. The more these routers are proliferated however, the smaller the chance that this local host will find a route to its destination bypassing all such routers along its path.

This is not a cause for concern, since this problem is one of initial penetration of the system in the Internet. While the system is still being deployed and not very prevalent in the Internet, many short flows would not be able to take advantage of these optimizations. This is the same problem that a client-based solution would face, so this problem of penetration is expected.

4. Evaluation

Based on the initial evaluation criteria laid out at the outset (Fairness, Performance, and Interoperability), the proposed scheme fares very well.

Fairness is ensured by the proposed scheme by two factors. Firstly if the local router detects that queue overflow for a particular local TCP connection is imminent, it deliberately drops packets, causing retransmissions and decrease of the window size. Secondly the router shares TCP state across different connections, so when multiple local hosts connect to a particular destination subnet (such as a popular website), the router immediately attains the optimal communication bandwidth rather than completely bypass TCP slow start and irresponsibly flood the network causing congestion.

Performance is improved since under the proposed scheme, the local hosts rapidly send the first few packets for T_{opt} milliseconds without delay, and then resume regular TCP Congestion Control semantics. Therefore throughput for short flows is increased. For longer flows, after the T_{opt} timer expires, traditional TCP Congestion Control resumes, so in this case the performance is no worse.

Interoperability is facilitated since the proposed solution relies only on modifications at the gateway access routers in a network, and not the clients themselves. Additionally, the proposed scheme works perfectly well even if only one network has this scheme, while the remote network it is communicating with does not possess this scheme.

5. Simulation

In order for the proposed scheme to work correctly, the choice of the T_{opt} timer is very important. If this timer were too long, then this would cause unfairness in the network. This is because the local host would continue to transmit beyond one round-trip time, and unnecessarily starve other nodes of their fair share of bandwidth. If the timer were too short, then short flows would not fully take advantage of available network bandwidth.

In order to choose the T_{opt} timer wisely, we analyze simulation results of queue sizes with network traffic characterized by varying T_{opt} timer values.

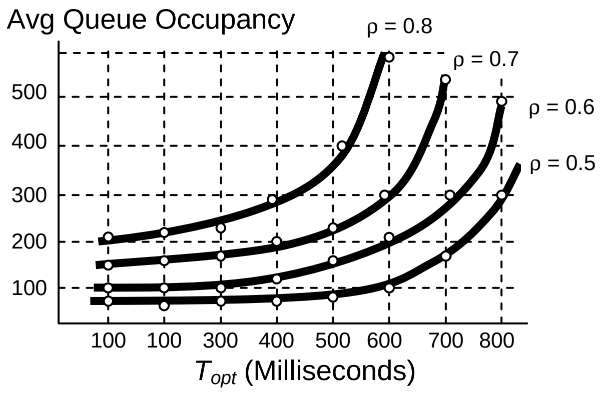

Fig 8. Simulation Results to select T_{opt}

As can be seen from Fig 8, for different values of

$$\rho = \lambda / \mu,$$

where λ = arrival rate

μ = service rate

the choice of 500 milliseconds is an appropriate choice, since before that queue occupancy does not grow significantly. After that value, the queue occupancy rises sharply.

6. Conclusions

The design of the proposed scheme has advantages as well as disadvantages.

The fact that the design involves no changes to existing TCP clients means that deployment is easy, especially in comparison with schemes that propose changes to the entire TCP layer. While these alternative schemes may arguably provide a more complete solution, the hurdles they face in deployment will result in those proposals being shelved just like the scores of other TCP congestion control proposals.

One disadvantage of this proposal is that it is not very universal. With the emphasis on the router, this scheme will only work when a local area network of computer nodes is communicating with the outside world through a well-defined network interface router. In many cases we just have individual nodes connected to the wide area network, and then this scheme is not directly applicable. Additionally if the gateway node were to experience an outage, all local hosts would be rendered incapable of taking advantage of this proposal.

Yet another disadvantage of this proposal is that it relies on a highly sophisticated router design. Although deployment is facilitated by having zero changes to the clients, the tremendous cost and complexity in designing the router software could prove to be prohibitive.

Future research in this line should explore the possibility of eliminating the T_{opt} timer altogether, and consider sending Phantom Acks to the local client indefinitely until real Acks are received by the remote host. This could result in additional network congestion, but if properly implemented, it could simplify the T_{opt} book-keeping. Additionally, a client-based approach of using this T_{opt} timer functionality can be studied. Although this will be harder to initially deploy, TCP clients will ultimately change, and for this reason it is viable to study such a long-term solution.

7. References

[1] *V. Jacobson*; "Congestion Avoidance and Control"; ACM SIGCOMM '88 pp 273-288; **1988**

[2] *R. Fox*; "TCP Big Window and Nak Options"; RFC 1106; **1989**

[3] *R. Chang, L. Huynh, J Gray*; "Adaptive Rate-Based Congestion Control Versus TCP-SS"; Proceedings 1993 International Conference on Network Protocols pp 186-197; **1993**

[4] *L.S. Brakmo, L.L. Peterson*; "TCP Vegas: End-to-End Congestion Avoidance on a Global Internet"; IEEE Journal on Selected Areas in Communications Vol. 13 Issue 8 pp 1465 – 1480; **1995**

[5] *S. Floyd*; "TCP and Successive Fast Retransmits"; ftp://ftp.ee.lbl.gov/papers/fastretrans.ps; **1995**

[6] *V. Jacobson*; "Modified TCP Congestion Avoidance Algorithm"; mailing-list, end2end-interest; **1990**

[7] *W.* Stevens; "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms"; RFC 2001; **1997**

[8] *M.T. Lucas, D.E. Wrege, B.J. Dempsey, A.C. Weaver*; "Statistical Characterization of Wide Area IP Traffic"; Proceedings 6th International Conference on Computer Communications and Networks pp 442-447; **1997**

[9] *K. Thompson, G.J. Miller, R. Wilder*; "Wide Area Internet Traffic Patterns and Characteristics"; IEEE Network Vol. 11 Issue 6 pp 10-23; **1997**

[10] *R. Wade, M. Kara, P.M. Dew*; "Proposed Modifications to TCP Congestion Control for High Bandwidth and Local Area Networks"; 6th IEE Conference on Telecommunications pp 151-155; **1998**

[11] *H. Wang, C. Williamson*; "A New Scheme for TCP Congestion Control: Smooth Start and Dynamic Recovery"; Proceedings 6th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems pp 69-76; **1998**

[12] *J.R. Chen, Y.C. Chen*; "Pseudo-Rate TCP: A Congestion Avoidance Scheme with Nearly Optimized Fairness and Throughput"; Computer Communications pp 1493-1501; **1999**

[13] *M. Allman*; "TCP Congestion Control"; RFC 2581; **1999**

[14] *A. Kamik, A. Kumar*; "Performance of TCP Congestion Control with Explicit Rate Feedback: Rate Adaptive TCP (RATCP)"; IEEE GLOBECOM '00 Global Telecommunications Conference; **2000**

[15] *F. Peng, S. Cheng, J. Ma*; "An Effective Way to Improve TCP Performance in Wireless / Mobile Networks"; EUROCOMM '00, Information Systems for Enhanced Public Safety and Security; **2000**

[16] *P. Pradhan, T. Chiueh, A. Neogi*; "Aggregate TCP Congestion Control Using Multiple Network Probing"; Proceedings 20th International Conference on Distributed Computing Systems; **2000**

[17] *H. Wang, H. Xin, D.S. Reeves, K.G.Shin*; "A Simple Refinement of Slow Start of TCP Congestion Control"; Proceedings 5th IEEE Symposium on Computers and Communications; **2000**

[18] *Y. Lai, C. Yao*; "TCP Congestion Control Algorithms and a Performance Comparison"; Proceedings 10th International Conference on Computer Communications and Networks; **2001**

[19] *M. Albuquerque, J.H. Kim, S. Roy*; "Effect of Packet Size on TCP-Reno Performance Over Lossy, Congested Links"; Military Communications Conference 2001 Vol. 1 pp 705-710; **2001**

[20] *F. Hu, N.K. Sharma*; "The Quantitative Analysis of TCP Congestion Control Algorithms in Third Generation Cellular Networks Based on FSMC Loss Model and Its Performance Enhancement"; Proceedings IEEE INFOCOM 2002 Vol. 1 pp 407-416; **2002**

[21] *W. Xu, A.G. Qureshi, K.W. Sarkies*; "Novel TCP Congestion Control Scheme and Its Performance Evaluation"; Proceedings IEE Communications Vol. 149 Issue 4 pp 217-222; **2002**