TCP Enhancement: Fast Active Queue Management Scalable Transmission Control Protocol

TABASSAM NAWAZ, MUHAMMAD SALEEM MIAN, HAFIZ ADNAN HABIB
Telecommunication & Information Engineering Department
University of Engineering & Technology
Taxila, Punjab
PAKISTAN

Abstract: - FAST TCP is Fast Active queue management Scalable Transmission Control Protocol. FAST TCP is an alternative congestion control algorithm in TCP. It is designed for high speed data transfers over large distance. FAST TCP is assumed to be the "successor" of TCP. Lot of people already worked on this subject. Their research work and results yield a new dimension for the entertainment and scientist eras. This research paper is the comprehensive study on FAST TCP.

Key-Words: - TCP**,** Fast TCP, Congestion Control, Fast TCP review, token based F-TCP.

1. Introduction:

Computer systems worldwide use TCP/IP protocols to communicate because TCP/IP provides the highest degree of interoperability, and runs over more network technologies than any other protocol suite. The robustness of TCP is the main reason for its large-scale deployment. Besides, TCP is one of a few transport protocols that have congestion control mechanisms. With acknowledgments and time-out based congestion control mechanism, the performance of TCP is inherently related to the bandwidth delay product of the connection. For a TCP connection, congestion might occur in those nodes (e.g., routers, IP/ATM access nodes) along the traversing paths, which mostly results in packet loss [6].

TCP breaks down large files into small packets of about 1500 bytes, each carrying the address of the sender and the recipient. The sending computer transmits a packet, waits for a signal from the recipient that acknowledges its safe arrival, and then sends the next packet. If no receipt comes back, the sender transmits the same packet at half the speed of the previous one, and repeats the process, getting slower each time, until it succeeds. This means that even minor glitches on the line can make a connection very sluggish. Because FAST TCP uses the same packet sizes as regular TCP, the hardware that carries messages around the net will still work. The difference is in software and hardware on the sending computer, which continually measures the time it takes for sent packets to arrive, and how long acknowledgements take to come back. This reveals the delays on the line, giving early warnings of likely packet losses. The FAST TCP software uses this to predict the highest data rate the connection can support without losing data [10].

Dr. Jian Ma originally proposed F-TCP in NRC, and it has become an ATM Forum draft. Its basic idea is to avoid congestion in intermediate nodes by effectively controlling acknowledgement (ACK) flow, that is, delay the ACKs to inform the source that the network will be congested [6].

1.1 Basic principle of the F-TCP flow control:

The present simple FAST TCP (F-TCP) flow control relates to end-to-end flow control in packet network where Transmission Control Protocol (TCP) is used as transport layer protocol. The most critical problem today in the Internet is the long control time of the TCP flow control which results buffer oscillation, low link utilization and low throughput. The main objectives of the F-TCP flow control are to remedy these problems by early informing TCP source that the network will be congested, and to direct the TCP source to slowdown its output rates. The basic idea of the scheme is to delay the ACKs being transferred from the destination towards the sender. This can be done at the same network point where congestion has been detected, or, alternatively, a network point detecting overload or congestion can direct another network point to delay the ACKs[1].

1.2 Simple FAST- TCP:

The basic idea of the F-TCP flow control is to delay the ACKs being transferred from the destination towards the source to inform the source that the network will be congested, and to direct the TCP source to slowdown its output rates. This can be done at the same network point where congestion has been detected or a network point detecting overload can direct another network point to delay the ACK. When detecting overload, F-TCP delays ACK on the backward path instead of discarding packets on the forward path to inform the TCP source [3].

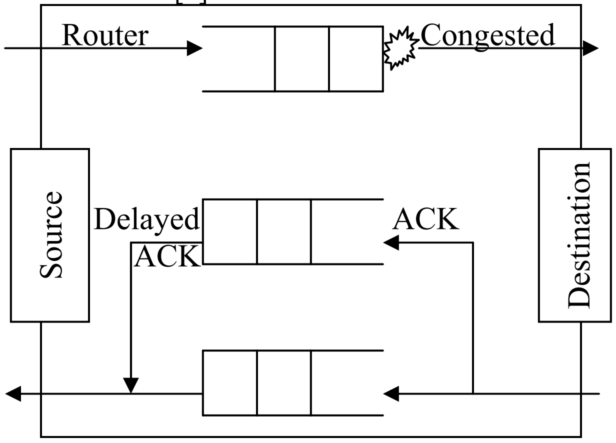


Fig 1: Prototype of Simple FAST TCP[3]

1.3 Comparison of Standard TCP with FAST TCP:

We will take a look on some properties comparison between standard TCP and FAST TCP.

1.3.1 Infrastructure problem

TCP: Does not solve infrastructure problem

FAST: Does not solve infrastructure problem

1.3.2 To get high throughput

TCP: The efficiency of the (congestion control algorithm in the) current TCP implementation drops steadily, and the protocol eventually becomes a performance bottleneck itself, as the network infrastructure scales up in capacity [10].

FAST: FAST TCP aims to remove this bottleneck: it is scalable to networks with large bandwidth-delay product [10].

1.3.3 Performance improvement in low speed networks

TCP: If the bottleneck in the end-to-end path is the 10Mbps or 100Mbps Ethernet card, we expect the current TCP implementation to be quite efficient, so there is not much to improve [10].

FAST: If the performance of the current TCP implementation is poor even at such speeds, then FAST TCP may or may not provide significant improvement depending on the reason for the poor performance [10].

1.3.4 High speed networks and wireless networks.

TCP: The current TCP performs poorly in two types of networks [10].

FAST: FAST TCP is optimized for the former and believe it can be tailored to provide significant benefit in wireless networks as well [10].

1.3.5 Delay-based congestion control

TCP: Delay-based congestion control has been proposed since the late 80s by Jain and many others, notably Brakmo and Peterson in TCP Vegas. We believe its advantage over loss-based approach is small at low speed, but decisive at high speed [10].

FAST: This does not mean that it is futile to use delay as a measure of congestion, but rather, that using a delay-based algorithm to predict loss in the hope of helping a loss-based algorithm adjust its window is the wrong approach to address problems at large windows. Instead, a different approach that fully exploits delay as a congestion measure, augmented with loss information, is needed [10].

1.3.6 Difficulties of the current TCP at large windows

TCP: Four difficulties contribute to the poor performance of current TCP implementation in networks with large bandwidth-delay product: [10]

- At the packet level, linear increase by one packet per Round-Trip Time (RTT) is too slow, and multiplicative decrease per loss event is too drastic.
- At the flow level, maintaining large average congestion windows requires an extremely small equilibrium loss probability that is hard to achieve in practice.
- At the packet level, oscillation is unavoidable because of the binary nature of the congestion signal (packet loss).
- At the flow level, the dynamics is unstable, leading to severe oscillations that can only be reduced by the accurate estimation of packet loss probability and a stable design of the flow dynamics [10].

FAST :

- FAST TCP is equation-based, hence avoiding packet level oscillation,
- FAST TCP has stable flow dynamics,
- FAST TCP uses queueing delay, rather than loss probability, as the main measure of congestion [10].

1.4 Token based F-TCP:

Fig 1 shows the prototype of F-TCP exploited in routers. The mechanism of FTCP could be divided into three parts: congestion detection, ACK's identification and delaying ACKs. A fixed threshold for the forward buffer occupancy is set, so that congestion is notified once the buffer occupancy exceeds the threshold. ACKs flows are delayed according to the state (CONGESTION or NON-CONGESTION), that is, when no congestion occurs, ACKs leak by a normal rate, otherwise, by a fraction of the normal rate. We set normal rate the same as the rate of data packet in the forward path. The fraction is set to half that the rate is halved when congestion is detected. Besides, determining the rate of Delaying ACKs is also a hazard problem. For F-TCP, ACKs should be delayed according to the network traffic conditions, while traffic in real network changes so quickly and frequently that it is difficult to grasp. So a scheme is developed called token-base F-TCP [6].

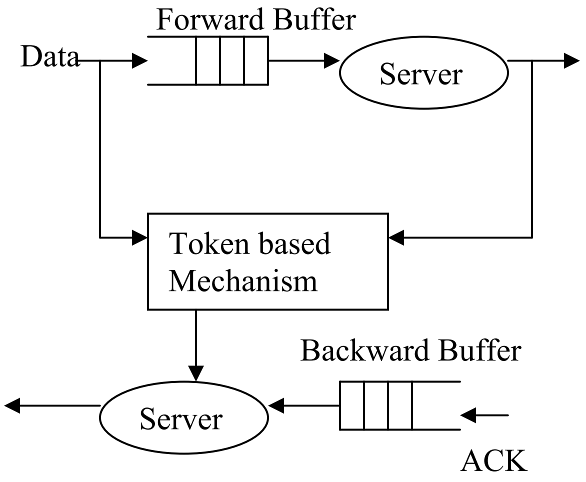


Fig: 2 Token Based FAST TCP

The prototype of the token-based FTCP is shown in Fig.3. When data packets arrive or leave and the resource condition changes, this mechanism collects associated information and recalculates the number of tokens. This mechanism has many potential advantages as following. Since this scheme is one type of F-TCP, it also possesses the merits of F-TCP, such as fully avoiding packet loss, shortening buffer capacity, smoothening traffic etc. It does not need to determine the threshold or the rate of delaying ACKs. ACKs are constrained closely related to current network resource (i.e., spare buffer occupancy) not the characterization of data traffic. As a result, this scheme is significantly robust. This scheme is also very simple and easy to algorithm. In TCP, delayed ACKs allows data receivers to refrain from sending ACK for every incoming data packet. Although delayed ACKs can reduce the number of packets sent by the receivers, excessive delays on ACKs will disturb the round-trip timing and inherent self-clocking of TCP.

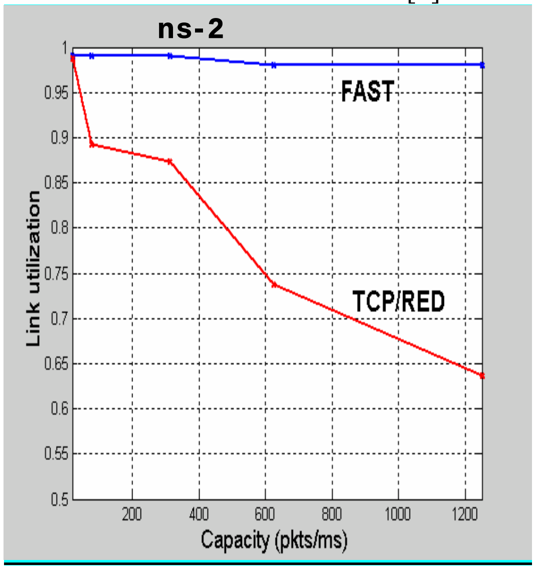
1.5 In ATM Networks:

The essential idea of FAST TCP (F-TCP) is to delay TCP acknowledgment (ACK) traveling towards its TCP source through a node where its forward channel is congested. It can be seen that: 1. F-TCP smoothes the peak of the TCP flow, consequently F-TCP reduce the requirement of ATM buffer;
2. With the same size of ATM buffer, F-TCP reduces the probability of overflow and as a result, improve the TCP throughput;
3. F-TCP reduces ATM buffer oscillation, since in most time the TCP is in congestion avoidance phase after short period of slow start phase, the flow is fairly smooth and the ATM buffer utilization is improved [1].

2. METHODOLOGY

2.1 WAN in LAB at netlab.CALTECH.edu

They have described the development of FAST TCP, from background theory to actual implementation and its first demonstration. Unlike TCP Reno and its variants, FAST TCP is delay-based. This allows it to achieve high utilization without having to fill the buffer and incur large queuing delay, as loss-based algorithms often do. It achieves proportional fairness and does not penalize flows with large RTTs. Whether FAST TCP can converge rapidly, yet stably, to a fair allocation in a dynamic environment where flows of heavy-tailed sizes join and depart in a random fashion, and in the presence of current TCP flows needs a lot more evaluation [9].



The congestion control mechanism of TCP separated into four components in fig 4. These four components are functionally independent so that they can be designed separately and upgraded asynchronously [2].

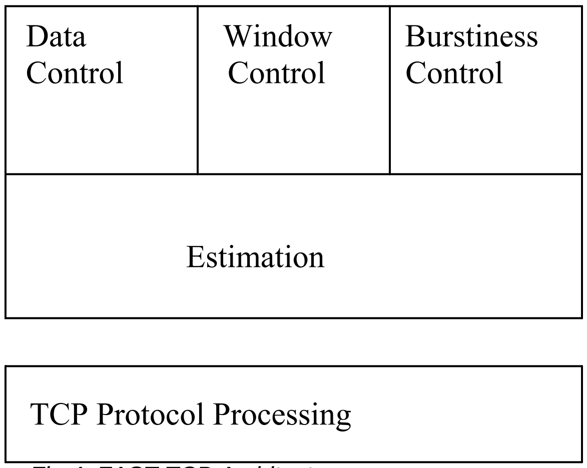
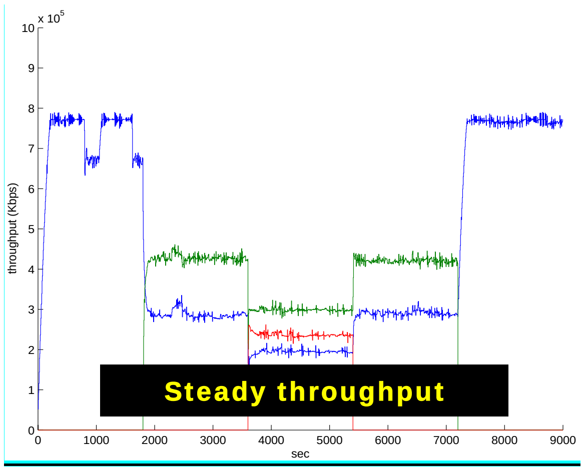


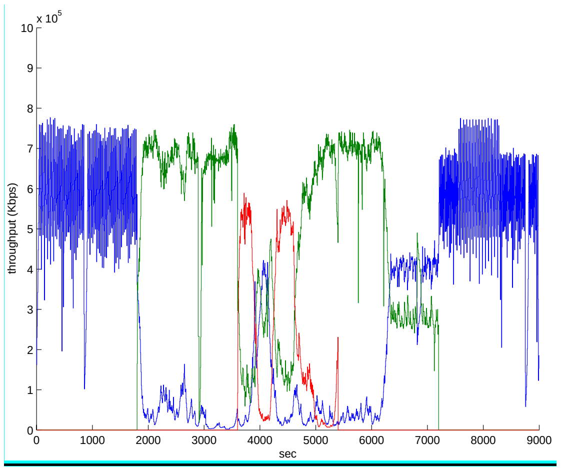
Fig 4: FAST TCP Architecture

They presented experimental results of their first Linux prototype and compared its performance with TCP Reno, HSTCP and STCP. They have evaluated these algorithms not only in static experiments, but also dynamic environments where flow comes and go; not only in terms of end to end throughput, but also queue behavior in the network [2].

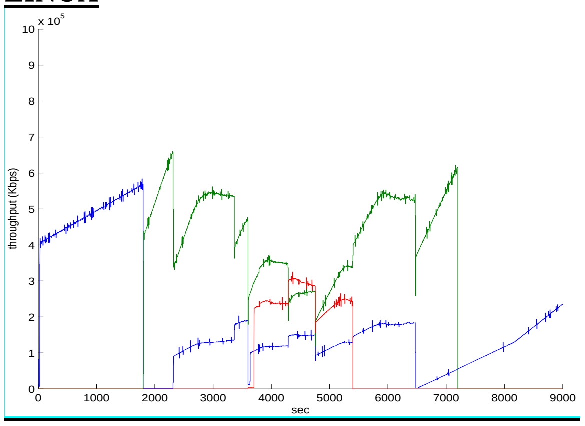
FAST



LINUX



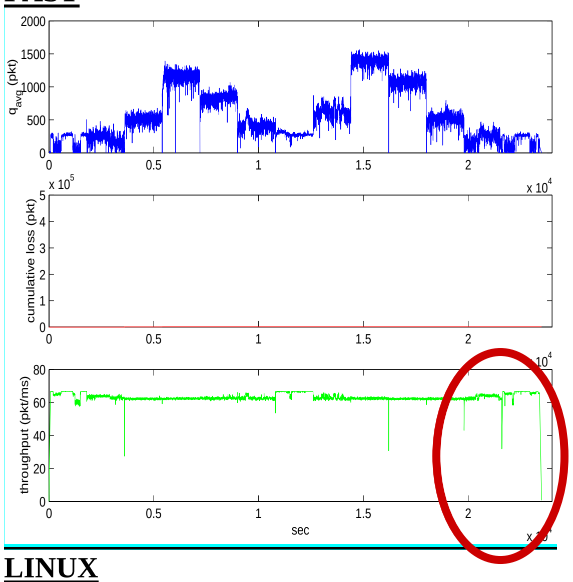
FAST



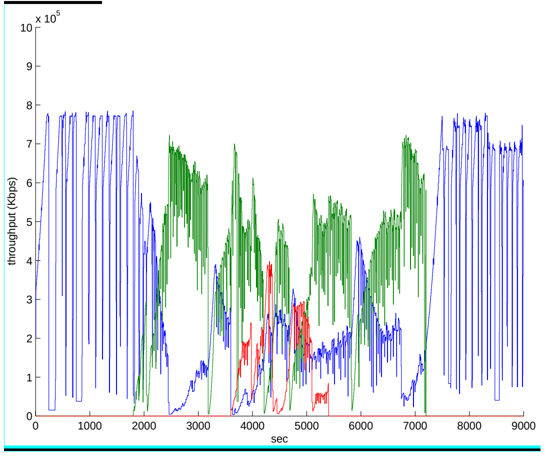
Dynamic sharing on Dummynet

- capacity = 800Mbps
- delay=120ms
- 3 flows
- iperf throughput
- Linux 2.4.x (HSTCP: UCL)

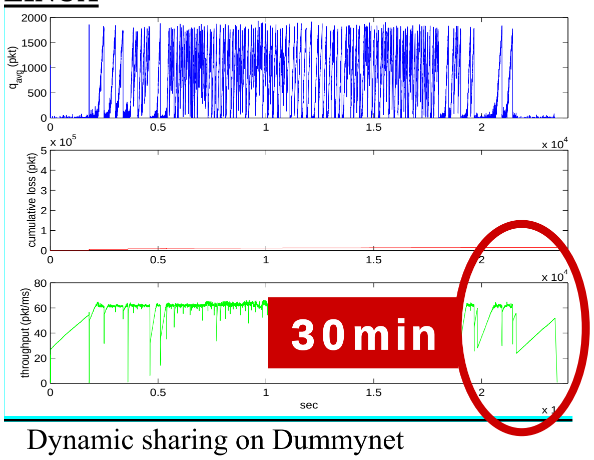
HSTCP



LINUX



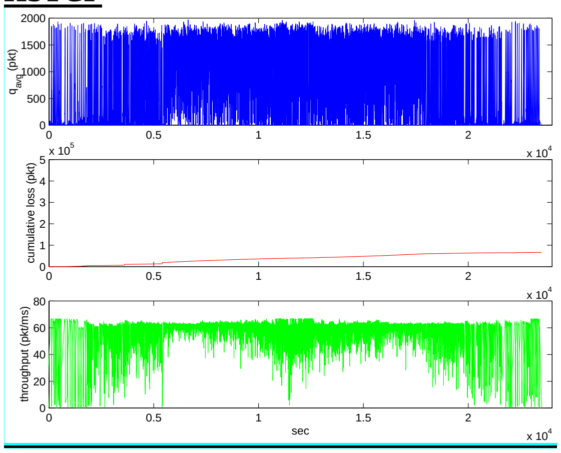
STCP



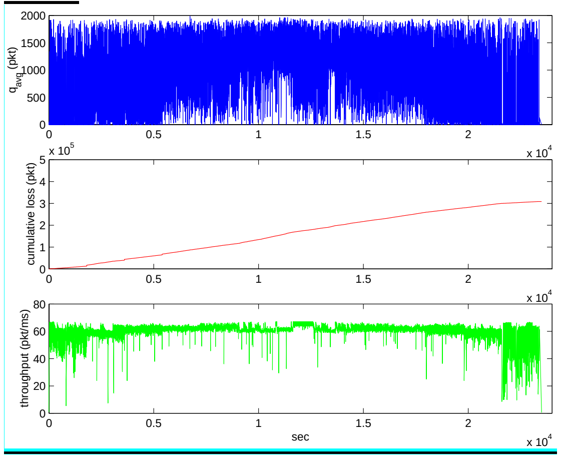
Dynamic sharing on Dummynet

- capacity = 800Mbps
- delay=120ms
- 14 flows
- iperf throughput
- Linux 2.4.x (HSTCP: UCL)

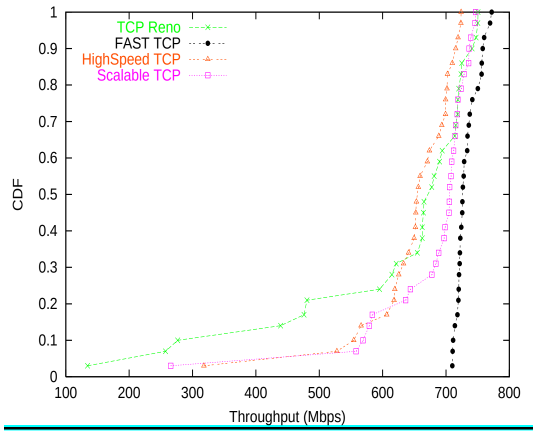
HSTCP



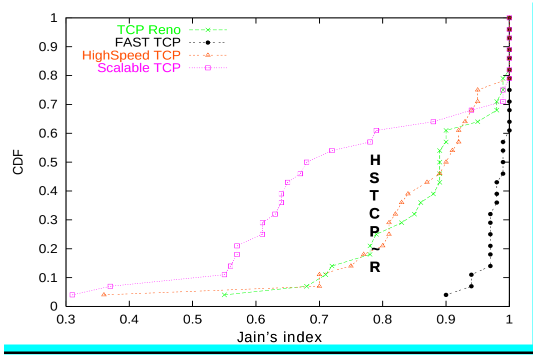
STCP



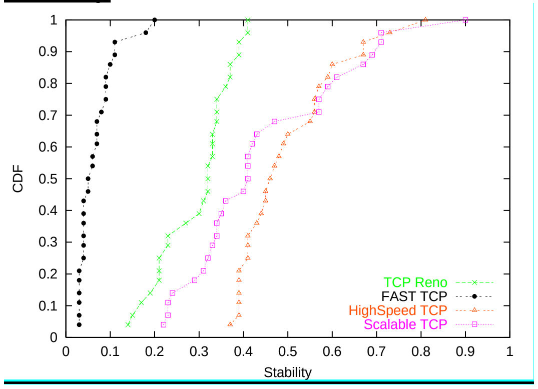
Aggregate throughput



Fairness



Stability



2.2 Token Based F-TCP

For the long control loop problems, a mechanism named FAST-TCP is proposed to avoid congestion in the intermediate nodes by effectively controlling ACKs flow traverse the same node as its forward packets. The algorithm contains three parameters: token buffer capacity: B_T; ACKs buffer capacity: B_A; number of tokens: N_T. When data packets arrive or leave, the resource condition changes, this mechanism collects the information and calculate number of tokens. Some temporary variables are needed: average packet length: L_p; data packet counts: P_c. The procedure of its implementation can be depicted with the pseudo-code:

```
Initialization: Packet count=0; $B_T$ = $N_T$ =Forward buffer capacity/Default packet length, then,
If a data packet arrives at the forward buffer,
1. Recalculate the average packet length
Lp and $B_T$
packet count = packet count+l ;
Lp=(Lp+input packet length)/packet count
$B_T$ =Forward buffer capacity/ Lp
```

2. Calculate N_T
N_T =Spare Forward buffer capacity/Lp
N_T = min(B_T, N_T); End
If a data packet leaves the forward buffer,
- Calculate N_T
N_T =Spare Forward buffer capacity/ L_p
N_T = min(B_T, N_T); End
If N_T >0 an ACK is served,
N_T =N_T --l; End [4]

2.3 In ATM Networks:

The implementation of F-TCP in the third layer and add some new statistics in the ATM switches to monitor the utilization and occupancy of the ATM buffer. They use a very simple example to show the effectiveness of F-TCP. Both the client and the server in the network use F-TCP. The client downloads one big file from the server. The following is some important parameters of the network [1].

2.3.1 Client and Server

ATM buffer capacity: 500 cells
IP Forwarding Kate: 10,000,000 packet/sec.
TCP Initial RTO (Retransmition Timeout): 0.5 sec.
TCP Maximum ACK Delay: 0.0 sec. (This parameter is the maximum time the TCP waits after receiving a segment before sending an acknowledgment.)
TCP Maximum KTO: 10 sec.
TCP Maximum Segment Size: 536 bytes
TCP Minimum RTO: 0.25 sec.
TCP Receive Buffer Capacity: 2,000,000 bytes
Trigger of Increase of Backward ACK Delay Time if F-TCP is Enabled: ATM buffer occupancy is greater than or equal to 472 cells
Increasement of Backward ACK Delay Time: 7.634E-5 sec [1].

2.3.2 ATM Switches (SW 1 and SW 2)

ATM Buffer Capacity: 1000 cells
ATM Switch Fabric Delay: 0.0 sec [1].

2.3.3 Links

Data Kate: 155,520,000 bit/sec.
Delay: from Client to SW I is 5E-6 sec., from SW 1 to SW 2 is 0.12 sec., from SW 2 to Server is 5E-6 sec [1].

2.3.4 Results when F-TCP is enabled

It can be clearly seen that after carefully setting the trigger condition of delay time increase for F-TCP and backward ACK delay time, ATM switch buffer will not overflow even though the buffer of the ATM switch is relative small (500 cells). In this simple example, the overflow of the buffer of the ATM switch can be completely avoided. The reason of this conclusion will be analyzed below. For 536 bytes of the packet of data from TCP layer, the ATM switch receives: 536 bytes of data + 20 bytes of TCP header + 20 bytes of IP header + 8 bytes of LLC header + 8 bytes of AAL5 trailer = 592 bytes. These are padded to produce 13 ATM cells. Since the receiver waits no time after receiving a segment before sending an ACK, each ACK will acknowledge 536 bytes data in the TCP layer. And for each ACK received by the sender, at most twice mount of data will be sent out. In another words, at most 2* 13 = 26 ATM cells will arrive at the buffer from the forward link and 2 ATM cells from backward link which are segmented from a ACK packet before the increasing of the delay time of backward ACKs by F-TCP takes effect. When the buffer occupancy reaches as high as 500 - 26 - 2 = 472 cells, additional specified delay time is triggered to be added to the consecutive backward ACKs. Therefore the consecutive ACKs stay more time in the 1P layer of the sender node before they are passing to the TCP layer. By carefully selecting the delay time, the buffer occupancy returns below 472 cells as a result of the cells being transferred by ATM switch [1].

2.3.5 Results when F-TCP is disabled

It can be seen that without F-TCP, the buffer of ATM switch faces to overflow. As a result, the throughput is very low. With the same network configuration and the parameters of the network elements, when F-TCP is enabled, more than 2M bytes of data are sent , while when FTCP is disabled, only less than 400K bytes of data are sent in the same period (10 sec.) One time of ATM buffer overflow will result in the loss of one TCP segment and consequently reducing the

TCP throughput significantly in plain TCP [1].

3. FAST TCP: Benefits, Achievements and Future Planning:

3.1 Advantages & Disadvantages of FAST TCP:

Advantages of FAST TCP

- FAST TCP is just like TCP in the sense that any application, such as FTP, using TCP will use FAST TCP once the patch is installed. You don't need any special programs to use FAST TCP.
- In principle FAST TCP can be transported over IPV6 or IPV4. Unfortunately the Linux implementation splits the TCP source into an IPV6 and IPV4 part and we only have an IPV4 implementation presently. An IPV6 implementation is on the roadmap [10].

Disadvantages of FAST TCP

- Reverse path congestion reduces throughput.
- Many FAST sources cause buffer overflow and packet loss.
- Route change may reduce throughput [10].

3.2 Avoid typical Internet congestion:

The transmission control protocol (TCP) is seen as the dominant transport protocol. The current stability of the Internet depends on the end-to-end congestion control of TCP. TCP does not perform well in high-speed wide area networks. To achieve a steady-state throughput of 7.2Gbps with 1500 byte packets and a 100 ms round trip time (RTT), for example, the packet loss rate must be less than $4.17 \times 10-10$. This is beyond the limits of achievable fiber error rates. In addition, TCP requires 40,000 RTTs, or almost 70 minutes, to recover from a single packet loss. This means that TCP cannot fully utilize the available bandwidth. The remarkable thing about FAST TCP is that it uses the existing Internet. The secret is in software at the sending point that parses the data into network-compatible packets that avoid typical Internet congestion as they weave their way to their ultimate destination [10].

3.3 Use standard packet size:

The protocol is called FAST, standing for Fast Active queue management Scalable Transmission Control Protocol (TCP). The FAST protocol sustained the speed using standard packet size, stably over an extended period on shared networks in the presence of background traffic, making it adaptable for deployment on the world's high-speed production networks. The ability to demonstrate efficient high performance throughput using commercial off the shelf hardware and applications, standard Internet packet sizes supported throughput today's networks, and requiring modifications to the ubiquitous TCP protocol only at the data sender, is an important achievement. The problem today is that this algorithm cannot scale to anticipated future needs, when the networks will be compelled to carry millions of uncompressed voice calls on a single path or support major science experiments that require the on-demand rapid transport of gigabyte to terabyte data sets drawn from multi-petabyte data stores. This protocol problem has prompted several interim remedies, such as using nonstandard packet sizes or aggressive algorithms that can monopolize network resources to the detriment of other users. Despite years of effort, these measures have proved to be ineffective or difficult to deploy. Using standard packet size that is supported throughout today's networks, the current TCP typically achieves an average throughput of 266 Mbps, averaged over an hour, with a single TCP/IP flow between Sunnyvale near SLAC and CERN in Geneva, over a distance of 10,037 kilometers [10].

3.4 The planned future improvements:

- Measurement of Backward Queuing delay to avoid reverse path congestion affecting throughput.
- Detection of route change.
- Tuning of Socket Buffers.
- Reducing number of ACKs processed to improve CPU utilization.
- Introduce 'TCP Friendliness' parameter to control sender aggressiveness in lossy environments.
- SACK processing optimization [10].

4. Conclusion

FAST TCP is an alternative congestion control algorithm in TCP. Lot of people already worked on this subject. Their research work and results yield a new dimension for the entertainment and scientist eras. TCP flow control mechanism can only indirectly detect congestion by keeping track of how many packets are lost, congestion control has to be initiated after packet losses due to congestion have already happened. Therefore, if TCP control time can not be speed up, TCP will cause major overloads and outages on long haul networks Furthermore, the maximum window size allowed in current systems is not large enough to catch up with Bandwidth-Delay Products. We studied Fast Active queue management Scalable Transmission Control Protocol comprehensively. FAST TCP uses the same packet sizes as regular TCP, the hardware that carries messages around the net will still work. The difference is in software and hardware on the sending computer, which continually measures the time it takes for sent packets to arrive, and how long acknowledgements take to come back. This reveals the delays on the line, giving early warnings of likely packet losses. The FAST TCP software uses this to predict the highest data rate the connection can support without losing data. We discussed the issues related to congestion control in the current TCP. We compare TCP with FAST TCP and also discussed Token-based FAST TCP. We also check its performance in ATM Networks. Steven Low and Co. did a fantastic job in this research area. We studied their work as well. Also some benefits, achievements and some planned future improvements are also described. And the bandwidth-hungry entertainment industry is also looking at Fast TCP.

References:

[1] Jing WU, Peng ZHANG, Tao DU, Jian MA and Shiduan CHENG, “Improving TCP Performance in ATM Network by the Fast TCP Flow Control,” International Conference on Communication Technology ICCT’98, October 22-24: 1998 Beijing, China

[2] C. Jin, D. X. Wei, and S. H. Low, “TCP FAST: Motivation, Architecture, Algorithms, Performance,” Proc. IEEE INFOCOM, Mar. 2004, http://netlab.caltech.edu

[3] Qian Wang, Jing Wu, Shiduan Cheng, Jian Ma2, “Fast TCP Flow Control with Differentiated Services”

[4] F. Peng, B. Wei and Y. Ma, “Delay performance analysis of token-based fast TCP in window limit systems”, Proc. 9th International Conference on Computer Communications and Networks, 2000.

[5] [13 Peng Zhang, Jian Ma, “ Token-based Fast-TCP ”, Invention Report, 1999

[6] H. Wu, Jing Wu, Keping Long, Shiduan Cheng, Jian Ma2, “TCP Enhancement: Token based Fast-TCP delay algorithm”

[7] W. Qian, W. Jing, C. Shiduan and Ma Jian, “Differentiated Service Fast-TCP Policy for Resource Management”

[8] F. Peng, C.M. Leung, “Performance Analysis of Token-based Fast TCP in Systems Supporting Large Windows”

[9] C. Jin, D. Wei, S. H. Low, “FAST TCP: From Theory to Experiments” http://netlab.caltech.edu/FAST/ December 6, 2003

[10] http://netlab.caltech.edu/FAST

[11] http://newscientist.com

[12] http://pr.caltech.edu/media