IJCST Vol. 4, Issue 2, April - June 2013 ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

Implementation of TCP Fast Open (TFO) and Proportional Rate Reduction (PRR) Mechanism by Google in TCP for Faster Web

[1]M. Vijayalakshmi, [2]G.Parthasarathi
[1,2]Dept. of Computer Science and Engineering, Velammal Engineering College, Anna University, India

Abstract
In the current cyber world, Google maintains the fastest web service as of date and they are planning to make it even better and faster in the future. Their main objective is to tackle challenges that make the Web slow and prevent it from delivering its full potential. All efforts made by Google, could be boosted by forthcoming protocols like HTML 5.0.Google's research shows that the network latency with respect to web transfers can be reduced by over 10%, by increasing the initial congestion window from 3 packets to 10 packets. They also propose to reduce the initial timeout from 3 seconds to 1 second as the present Internet requires a timeout lesser than 3 seconds [3]. Thus Google proposes TCP Fast Open (TFO) protocol, which permits data transfer during TCP's initial handshake.

Keywords
Fast Open, Latency, TCP, Transmission, PRR

I. Introduction
For thirty three percent of all HTTP requests, one full Round Trip Time (RTT) is required to establish a TCP connection with a remote entity. TFO overcomes this problem by including the data in the initial TCP SYN packet and delivers the data while the three way handshake is in progress. This reduces the delay of short TCP transfers, by minimizing the latency in the network by1RTT [3]. Google also proposes the use of Proportional Rate Reduction (PRR) for TCP.Packet loss increases latency for Web users. Faster recovery is a key mechanism for TCP to recover from packet losses.PRR is a design used to control transmission in fast recovery.PRR was implemented in Linux kernel and now it is implemented in TCP. With respect to TCP, it is a loss recovery algorithm which operates smoothly to recover from packet losses during congestion [3]. In this paper we discuss in detail about the new technologies namely Proportional Rate Reduction (PRR) and TCP Fast Open (TFO) that will help in improving the speed of internet thereby increasing the usage of it. This paper contains the following sections. Section II introduces the new fast recovery algorithm (PRR) and its key properties and discusses about the design overview of TFO and its security considerations and deployment status. Section III discusses about the status of the YOUTUBE application using various fast recovery algorithms. Section IV includes conclusion and future work.

II. PRR and TFO

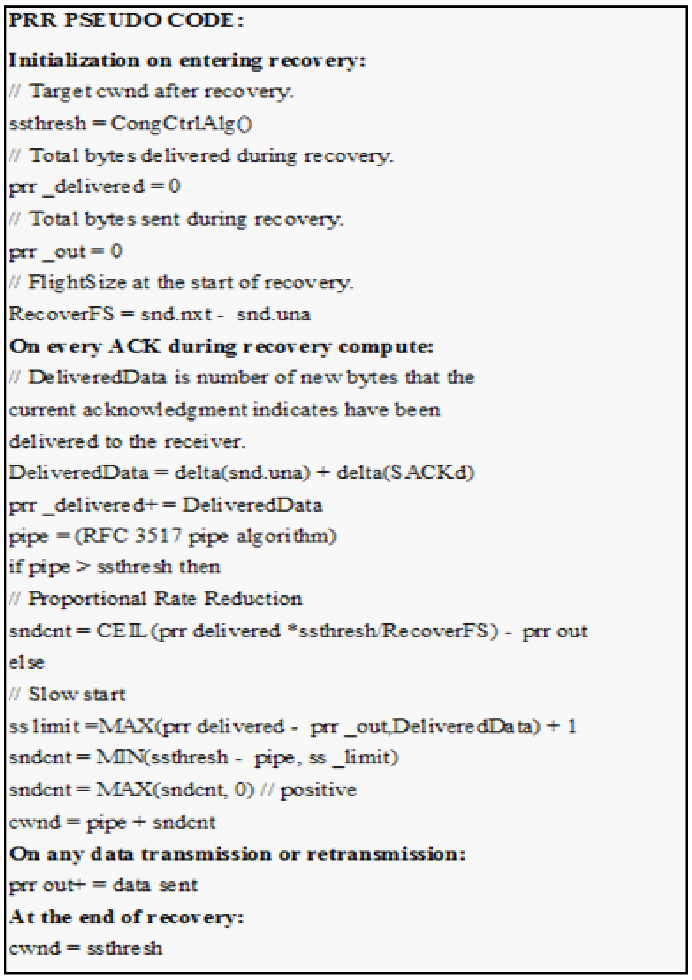
A. Goal of TFO and PRR
The present TCP specification allows a client to include data in its SYN packet when initiating connections to servers, but does not allow the server from delivering the data to application until the 3 way handshake is complete. The proposed TCP fast open is designed in such a way that it enable each end of a TCP connection to safely transmit and process any received data while the 3 way handshake is still in progress [1]. The main intension of using PRR is to reduce latency for TCP connections experiencing packet losses. PRR operates smoothly when losses are heavy and it recovers quickly for short flows and it operates accurately when acknowledgments are re-ordered or lost [2].

B. Implementation of PRR Algorithm in TCP
PRR is a new fast recovery algorithm which is inspired by the rate halving algorithm i.e. finding the correct window size following a packet loss. This algorithm determines the number of segments to be transmitted per ACK during recovery phase. It is used to satisfy two goals namely a speedy and smooth recovery from losses and the window size will remain close to ssthresh(i.e. window size set by the congestion control algorithm) at the end of the recovery. Using PRR Algorithm as shown in figure 1, we determine the number of segments transmitted per acknowledgment during recovery phase [2]. The algorithm depends on three new state variables namely prr_delivered (total no. of bytes delivered to receiver during recovery), prr_out (total no. of bytes transmitted during recovery) and RecoverFS is the Flight Size (amount of outstanding data in the network) at the start of recovery. In addition PRR maintains two local variables namely Delivered Data (the total number of bytes that the current ACK indicates that have been delivered to the receiver) and sndcnt (total no. of bytes to be sent in response to each acknowledgment). At the beginning of recovery the PRR state is initialized. The modern congestion algorithm CongCtrlAlg(), might set ssthreshto something other than FlightSize/2. The variables prr _delivered and prr _out during start of recovery are initialized to zero. In every TCP connection, variables are stored in a connection record known as transmission control block (TCB). This record contains 'send sequence' variables namely snd.nxt (send next) and snd.una (send unacknowledged). The difference between these two variables gives the FlightSize at the start of recovery. For every acknowledgment received during the recovery phase the following is computed. First, SACKd (i.e. total number of bytes that have been delivered to the receiver after receiving SACK) can be computed by scanning the TCP scoreboard (data structure that stores per packet state) and counting the total number of bytes covered by all SACK (Selective acknowledgment) blocks. The change in previous send unacknowledged value (snd.una) plus the change in previous selective acknowledged value (SACKd) gives the current delivered data. The above algorithm is described as modifications to RFC 5681 TCP Congestion Control, using concepts drawn from the pipe algorithm (RFC3517-loss recovery algorithm based on SACK for TCP) [2].In pipe algorithm, pipe refers to total number of bytes outstanding in the network as per sender's estimation. Pipe is used to determine which of the two algorithm modes namely PRR or Slow Start is used to compute the number of segments to be sent per acknowledgment. If the pipe (number of bytes outstanding in the network) is greater than ssthresh then sndcnt (i.e. total number of bytes sent in response to each acknowledgement) is calculated using PRR.If the pipe is less

than ssthresh then slow start is performed instead of PRR.Slow start limit is obtained by taking maximum of difference between the number of bytes sent and delivered during recovery and the latest Delivered Data and then increment the obtained value by one. Later the sndcnt is computed using slow start. If sndcnt values attained are positive, it indicates number of bytes sent in response to each acknowledgment. If the values attained are negative then no bytes can be sent. If data is transmitted successfully using the PRR algorithm (which is only for recovery mode) then the sum of all the data sent will give the total bytes sent during recovery (prr_out).At the end of recovery, congestion window is initialized to ssthresh [2].

C. Properties of PRR

The key properties of PRR are ACK clocking; Convergence to ssthresh and Robustness of Delivered Data [2]. ACK clocking property (i.e. setting transmission rate with respect to bandwidth and delay of the network) is maintained by PRR even for large burst segment losses. The second property states that for small number of losses the window size is exactly the window chosen by the congestion control algorithm, ssthresh. For burst losses, window size reduction can be avoided by sending extra segments in response to acknowledgments arriving later during recovery. As long as few window size reductions are undone, the final value will be same as ssthresh chosen by congestion control algorithm. The PRR provides robustness of Delivered Data by allowing the TCP sender to learn about the precise number of segments arrived at the receiver end [2].



```
PRR PSEUDO CODE:
Initialization on entering recovery:
// Target cwnd after recovery.
ssthresh = CongCtrlAlg()
// Total bytes delivered during recovery.
prr _delivered = 0
// Total bytes sent during recovery.
prr _out = 0
// FlightSize at the start of recovery.
RecoverFS = snd.nxt -  snd.una
On every ACK during recovery compute:
// DeliveredData is number of new bytes that the
current acknowledgment indicates have been
delivered to the receiver.
DeliveredData = delta(snd.una) + delta(SACKd)
prr _delivered+ = DeliveredData
pipe = (RFC 3517 pipe algorithm)
if pipe > ssthresh then
// Proportional Rate Reduction
sndcnt = CEIL(prr delivered *ssthresh/RecoverFS) -  prr out
else
// Slow start
ss limit =MAX(prr delivered -  prr _out,DeliveredData) + 1
sndcnt = MIN(ssthresh -  pipe, ss _limit)
sndcnt = MAX(sndcnt, 0) // positive
cwnd = pipe + sndcnt
On any data transmission or retransmission:
prr out+ = data sent
At the end of recovery:
cwnd = ssthresh
```

Fig. 1:

III. Design Overview of TFO in TCP

Figure 2 depicts the overview of TFO connection [1]. The client requests for TFO cookie (encrypted data string used for validating the IP ownership of the client) by sending aSYN packet with a Fast Open Cookie request option to the server. The server generates a cookie and transmits it through the Fast Open Cookie option of SYN-ACK packet. The client caches the cookie in first TCP connection and uses it for future TCP connections to exchange data during 3-way handshake (3-WHS). TCP Fast Open operates as follows: The client sends a SYN with Fast Open cookie option and data. The server performs validation of cookie. If cookie is valid, the server sends SYN-ACK packet i.e. it acknowledges both SYN and data. The server later delivers the data to the application. If not, the server sends a SYN-ACK packet i.e. it acknowledges only the SYN sequence number and drops the data. If the server accepts the data in SYN packet then it sends the response data within the completion of three way handshake. The client sends an ACK (acknowledging SYN+server data) to the server. If the client data is not acknowledged, the client retransmits data in ACK packet. The rest of connection proceeds like normal TCP connection [1].

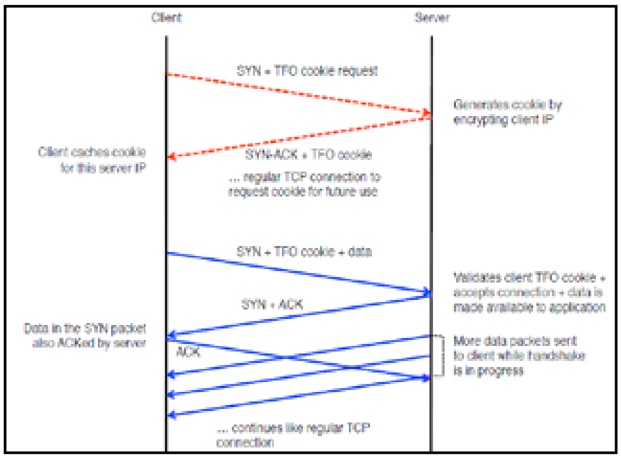

Client
Server
SYN + TFO cookie request
Generates cookie by encrypting client IP
Client caches cookie for this server IP
SYN-ACK + TFO cookie
... regular TCP connection to request cookie for future use
SYN + TFO cookie + data
Validates client TFO cookie + accepts connection + data is made available to application
SYN + ACK
Data in the SYN packet also ACKed by server
ACK
More data packets sent to client while handshake is in progress
... continues like regular TCP connection

Fig. 2: TFO Connection Overview [1]

A. Security Considerations in TFO Design

The main goal of designing TFO is to prevent source address spoofing attack. Standard TCP suffers from the SYN flood attack. The server's queue can be filled up easily with attacker's SYN packets with spoofed source IP addresses causing the service port to be blocked completely until timeouts. [4] The faked SYN packets cause memory wastage. TFO design allows the TCP server to validate the cookie sent by the attacker. If the cookie is invalid, the server drops the data. There might be some security malfunctioning if the cookies presented by attackers are valid, the server might undergo resource exhaustion as the connections are exhausted. To overcome this type of problem, TFO server maintains the total number of pending TFO connection requests. When the total number of pending connections exceeds the threshold value, the server disables the TFO temporarily. Thus limiting the overflow of attacker's SYN packets across the network [1].

B. Current Status of TFO in Deployability

A study found that network or end-hosts may drop packets with unknown TCP options. Another study shows that six percent of the probed paths on the Internet drop SYN packets with data. This problem is dealt by TFO protocol by retransmitting SYN without data. In doing so, TFO acts as normal TCP 3-WHS and the server connectivity is not lost [1].

C. Potential Application: YouTube

Two Fast Recovery Algorithms namely PRR, RFC 3517 are evaluated in India YouTube video servers to observe the performance with TCP transfers as shown in Table 1 [2]. YouTube services in India are built on the HTTP where one TCP connection is used to deliver entire video. The first twenty seconds of the video are sent to the user and then the transfer rate is limited based on video encoding rate in order to minimize the wasted network capacity. Considering video downloading in India, the difference between throttled and less throttled data rates is less because the network capacity is not that great above the video encoding rate. In such cases, TCP is mostly in bulk transfer mode. A sample of 0.8 million TCP connections in 96 hours was taken. Video downloads served with different fast recovery algorithms have more or less same average transfer size (2.3MB per connection). In India 43 to 46% of network transmit time (total time per connection when there is unacknowledged data in the TCP write queue) is used for recovering from losses, whereas RFC3517 spends more time to retransmit as well as sends more data during re-transmission compared to PRR. This is because the pipe drops below ssthresh for about 40% of events, causing RFC3517 to send as much data necessary to keep the pipe at ssthresh. Thus fast and aggressive retransmission causes 16.4% of the RFC 3517 fast retransmit to drop, while PRR lose less than 5% of fast retransmit. On the whole RFC3517 retransmits faster than PRR but PRR achieves good performance just with slight increase of retransmission rate. PRR reduces time spent in recovery compared to RFC 3517 [2].

Table 1:

	RFC 3517	PRR
Network Transmit Time (s)	83.3%	84.8%
% Time in Loss Recovery	46.3%	44.9%
Retransmission Rate %	6.6%	5.6%
% Bytes Sent in FR	12%	10%
% Fast-retransmit Lost	16.4%	4.8%
Slow-start after FR	1%	0%

VI Conclusion and Future Enhancement

TCP fast open enables data to be exchanged safely during the TCP's handshake and enables the application to decrease the latency by one RTT. TFO's cookies (8-byte) are used as a defensive mechanism for attacks. The main advantage of TFO is its simple design and its defence towards DOS attacks [1]. TFO is more efficient than the present TCP implementation. When it comes to PRR, it improves the fast recovery and reduces the latency of short web transfers by 3-10%. PRR proved to be the smooth recovery for video traffic and a default fast recovery algorithm [2].By increasing the initial window size using TFO may cause buffer overflow and packet loss in routers with small buffers. This leads to re-transmission of packets. In future, the technique must be enhanced to overcome this type of problem.

References

[1] "TCP Fast Open", Sivasankar Radhakrishnan, Yuchung Cheng, Jerry Chu, Arvind Jain, Barath Raghavan, In Proceedings of the 7th International Conference on emerging Networking Experiments and Technologies (CoNEXT), December 6-9, 2011.

[2] "Proportional Rate Reduction for TCP", Nandita Dukkipati, Matt Mathis, Yuchung Cheng, Monia Ghobadi, In Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement 2011, November 2-4, 2011.

[3] Paul Krill (2012),"Google looks to speed up Internet [Online] Available: http://www.infoworld.com/d/networking/google-looks-speed-the-internet-184839

[4] Larry L. Peterson,"Computer Networks: A Systems Approach".

[5] Douglas Comer,"Internetworking with TCP/IP: Principles, protocols, and architecture".

M.Vijayalakshmi (First author) completed her Bachelors in technology in Information technology in velammal engineering college in the year 2004. and she did her Masters in Information technology from Sathyabama university in 2008. She is presently the Assistant Professor in department of computer science, Velammal engineering college-Anna university. She is a IEEE,CSI member and her research area include Data mining and Date warehousing. she has presented papers at International conferences in the area of data warehousing.

G.Parthasarathi is presently pursuing final year Bachelors in Engineering at velammal engineering college in Computer science and engineering. He is a IEEE, CSI student member. He has presented 2 papers in international conferences in the area of motion tracking and image processing. He was awarded the Young investigator award for the best research work. He is currently the editor of the newsletter from Computer society of India.