Traffic Control Method on TCP/IP Communication

A.Siles Balasingh\(^1\)  
Department of Computer of  
Computer Science Engineering  
St.Joseph University in  
Tanzania,East Africa  
singh_bala@yahoo.co.in

Omkar Nath Tiwary\(^2\)  
Department of Computer of  
Computer Science Engineering  
St.Joseph University in  
Tanzania,East Africa  
Omkar.tiwary@yahoo.co.in

N.Surenth\(^3\)  
Department of Computer of  
Computer Science Engineering  
St.Joseph University in  
Tanzania,East Africa  
suren.zin1@gmail.com

Abstract—In this paper we have cast the idea of two windows for measuring the real Traffic window and status of packet loss due to corruption in channel. Proposed algorithm changes real Traffic window based on degree of packet loss due corruption in channel to increase in throughput of network. It has been tested on the network simulator NS3. Results shows that Traffic widow get affected only in case of packet loss due to high load rather than packet loss due to corruption in channel. It enhances the network throughput significantly of such networks, which is susceptible to error, with respect to present TCP variant such as Tahoe, Reno and NewReno.

Keywords—Traffic control; two windows; styling; error-prone network.

I. INTRODUCTION

We know the TCP/IP network is reservation less, packet switched network. Its capacity is shared dynamically by a community of competing user. To control the abuse of this dynamically sharing there are Traffic control mechanisms [1], so that one can not jeopardize all network to reduce its performance. At transport layer different protocol like TCP, SCTP, DCCP et al are available to provide Traffic control mechanism. Among these protocols TCP has been most prominent due to its end to end reliability and broad use in different application. Rather than looking into new protocol, we opted to enhance the present TCP protocol.

A. Evolution of TCP Traffic Control Mechanism

The trend in TCP Traffic control have been by limiting the sending strength based on Traffic window, changes on occurrence of packet loss due to high load. In the beginning of Internet focus has been on wired network, so lots of TCP Traffic control mechanisms have been suggested and implement on wired technology. The change in TCP Traffic window uses initially the Additive Increase Multiplicative Decrease (AIMD) mechanism, halving the Traffic window for every window containing a packet loss, and increasing the Traffic window by roughly one segment per RTT otherwise along with it’s another derivative like MIAD or MIMD [2] tried. Just increasing the Traffic window is not sufficient, retransmission time was also one of major component of TCP Traffic control to decide when actually packet should be resent based on the Retransmit Timer, including the exponential back off of the retransmit timer when a retransmitted packet is itself dropped [3]. A third fundamental component is the Slow Start mechanism for the initial probing for available bandwidth, instead of initially sending at a high rate that might not be supported by the network. The fourth TCP Traffic control mechanism is ACK-clocking, where the arrival of acknowledgments at the sender is used to clock out the transmission of new data. For embedding the above component only single window was maintained based on packet loss due to high loss [2].

Later on fast retransmission algorithm been added to avoid waiting for the retransmission timer to expire following every packet loss. These techniques provided stability but have been very conservative. To rid off from this fast recovery algorithm has therefore been introduced [5].

B. Problem Identification

Unfortunately above mechanism could not give expected result due to packet loss due to corruption in channel. We know internet technology is changing fast and wireless network is becoming pervasive all around. And packet lost in wireless network may not imply Traffic in network because it may happen due to weather conditions, obstacles, and multipath interferences, mobility of wireless end-devices, signal attenuation and fading. So our assumption upon which network Traffic window has been changing in TCP Traffic control mechanism is not appropriate for wireless.
C. Proposed TCP Traffic Control Mechanism

This new situation is deteriorating the network performance. For wireless network different flavor of TCP Traffic control have been suggested and implemented but there is need of such algorithm which can be ported upon wired/wireless network universally. To improve the situation we need to use loss discrimination mechanism i.e. packet loss happens due to high load or corruption in channel [4]. In this improved algorithm we have used two window where one window measures Traffic window and another window measures degree of packet loss due corruption in channel.

Packet Loss Rate

One of the major characteristic, which differentiates between wired and wireless network is large error rates due to noise, fading, interference from other sources and mobile host movement. Before we go to design modified TCP, let’s define Bit Error Rate (BER).

Bit Error Rate (BER) and Packet Error Rate (PER) are important Quality of Service Parameters for Wireless network. Basically any event is defined as any divergence of the decoded path at the receiver from the initially followed path in the encoder. For long codes, the error probability of block codes and convolution codes is upper-bounded in terms of error exponent. The error exponent $E(R)$ is defined as

$$E(R) = -\frac{1}{N} \log Pe(N, R)$$

Where $Pe(N, R)$ is the error probability of a block code of length $N$ and rate $R$.

With the classical approach where it is supposed that errors are uniformly distributed in packets with a probability given by BER. With this hypothesis the packet error rate (PER), which is the number of incorrectly received data packets divided by the total number of received packets. A packet is declared incorrect if at least one bit is erroneous. The expectation value of the PER, for which a data packet length of $N$ bits can be expressed as [7]

$$PER = 1 - (1 - BER)^N$$

For example, normally the bit error rate of wireless BER=0.00001 and the length of data frame Length=1024bits, so the corruption loss rate PER=0.0101878; the PER=0.097336209 while BER=0.0001 and PER=0.641028521 while BER=0.001.

It is reasonable to assume that the possibility of packet lost by corruption can be obtained approximately from $Pe=m/n$, where $n$ is the number of total packets and $m$ is the sum of packets lost by corruption during the period of time $T$.

From the simulation point of view, if the corruption loss rate Pe is higher than the certain lower limit Pemin, the sending rate will be decreased. Many factors decide the value of corruption loss rate lower limit Pemin, mainly include: the kind of application; the length of data frame; the bit error rate of wireless link layer; the bandwidth and the transmission delay of wireless network etc. Normally we choose Pemin=0.4.

II. MODIFIED TCP

We designed improved TCP by modifying the base design of TCP NewReno [8] and improved it by embedding additional context. This improved TCP Traffic control mechanism is designed by changing into Slow-Start, Traffic Avoidance, and Fast Recovery part based on present corruption loss rate with respect to Pemin.

This modified TCP Traffic control has been improved for wireless network by keeping unaffected in case of wired network. It considers the influences to TCP sender’s packet sending rate by packet loss in Traffic and considers the degree of packet loss due to corruption in channel too.

During transmission of packet in network we measure the degree of loss of packet due to corruption in channel and accordingly keep the Traffic window as normal TCP Traffic window otherwise updated to second window, which we maintain in this Traffic control mechanism.

We use $ewnd$ (error Traffic window) as original Traffic window (cwnd) multiplied by corruption loss rate and $swnd(similar Traffic window)$ as original Traffic window subtract $ewnd$. It mean

$cwnd=swnd+ewnd$;

We initialize $swnd$ and $cwnd$ equal to one and $ewnd$ to zero. These three windows will change their value according to present corruption loss rate (Pe). In general if Pe is less than Pemin then $cwnd$ will similar to original NewReno $cwnd$, $ewnd$ is scale of corruption loss and $swnd$ is deviation of $ewnd$ from $cwnd$

III. ALGORITHM

A. Initial Window

The IW, the initial value of $cwnd$, MUST be less than or equal to $2*SMSS$ bytes and MUST NOT be more than 2 segments. It calculated as follow [9].

$Cwnd = \min (4*SMSS, \max (2*SMSS, 4380$ bytes))

B. Slow Start Algorithm

The slow start algorithm is used to start a connection of improved TCP like another TCP, and the periods after the value of retransmission timer exceed the RTO (retransmission timeout). In the start of improved TCP, the size of $cwnd$ will be initialized to 1. The slow start algorithm describes as below

Slow start algorithm is used to start a connection and the periods after the value of retransmission timer exceed the RTO (retransmission timeout). When the ACK is received, the $swnd$ is increased from one to two, and two segments can be sent. This provides an exponential growth. The slow start algorithm will be ended in two conditions.
First, if the Traffic window size reaches the slow start threshold size (ssthresh), the slow start will be ended and then traffic avoidance takes over. Second, if there lose any packet due to Traffic or high packet loss rate due to corruption, the slow start also will be ended and then fast recovery takes over.

```c
if (Receive ACKs && cwnd < ssthresh)
{
    cwnd++; // Increase congestion window
    swnd=cwnd; // Set slow start window
    ewnd=0; // Empty early start window
}
```

C. Congestion Avoidance Algorithm

If the Traffic window size (cwnd) is less than or equal to the slow start threshold size (ssthresh), improved TCP is in slow start; otherwise it is performing Traffic avoidance. The Traffic avoidance algorithm describes as below.

```c
if (Receive ACKs || (Receive Explicit Corruption Loss Notification && Corruption Loss Rate Pe<Pemin))
{
    if (cwnd > ssthresh)
        cwnd = cwnd + 1 / cwnd; // Increase congestion window
    else
        cwnd++; // Increase congestion window
    m++; // Increase total packets sent
    if (Receive Explicit Corruption Loss Notification)
        n++; // Increase lost packets due to corruption
        Pe=a* Pe + (1-a) * (n/m); // Update corruption loss rate
        ewnd=cwnd* Pe; // Set early start window
        swnd=cwnd – ewnd; // Set slow start window
    if (m>cwnd)
        m=n=0; // Reset counters
}
```

In the algorithm, cwnd denotes the Traffic window size; m denotes the total number of sending packets; n denotes the number of lost packets due to wireless link corruption; Pe denotes the corruption loss rate and use parameter “a” to add up the old values. The idea of “a” has been taken from Jacobson RTT calculation [10].

D. Fast Retransmission and Fast Recovery Algorithm

The TCP sender should use the "fast retransmit" algorithm to detect and repair loss, based on incoming duplicate ACKs. The fast retransmit algorithm uses the arrival of 3 duplicate ACKs (4 identical ACKs without the arrival of any other intervening packets) as an indication that a segment has been lost. After receiving 3 duplicate ACKs, TCP performs a retransmission of what appears to be the missing segment, without waiting for the retransmission timer to expire.

"Fast Recovery procedure" begins when three duplicate ACKs are received and ends when either a retransmission timeout occurs or an ACK arrives that acknowledge all of the data up to and including the data that was outstanding when the Fast Recovery procedure began [8].

In improved TCP, the fast recovery algorithm will be taken when network Traffic or heavy corruption occurs. the network Traffic, set ssthresh to one-half the flight size or double of MSS (maximum segment size) window. If the network has high corruption loss rate, set cwnd to c times its original size [12].

Here the network Traffic means the sender receives the same ACK 3 times or retransmission timer overtime. The heavy corruption means the corruption loss rate Pe not less than Pemin when sender receives explicit corruption loss notification.

```c
if (Traffic || Heavy Corruption)
{
    if (Receive Same ACK 3 Times || Retransmission Timer Overtime) /* Traffic */
    {
        ssthresh = cwnd / 2; // Decrease congestion window
        if (Retransmission Timer Overtime)
        {
            cwnd = 1; // Exit and call slow-start
        }
        else /* Receive Same ACK 3 Time */
            cwnd = cwnd / 2; // Decrease congestion window
    }
```
else if (Receive Explicit Corruption Loss Notification & Corruption Loss Rate Pe>=Pemin)
{
    ewnd=c* ewnd;
    cwnd=cwnd+ ewnd;
}

m++;  
if (Receive Explicit Loss Corruption Notification)
    n++;

Pe=a* Pe + (1-b)* (n/m);
ewnd=cwnd* Pe;
swnd=cwnd – ewnd;
if (m>2cwnd)
    m=n=0;

}   //” Traffic || Heavy Corruption close”

IV. COMPUTER SIMULATION

A. Simulation environment

To demonstrate effectiveness of our improved TCP Traffic control mechanism, we use network simulator version 3.10 (NS3) to study the transport scheme, the proposed model has been implemented and simulated in the network simulation tool NS3. It is open source and is a relatively new simulator. NS3 provides great flexibility while simulating various scenarios [11].

Figure 1 illustrates the network topology of the simulation. Channel may wire or wireless with error model, which add erroneous of channel. The bandwidth of link is 5Gbps and delay is 20ms.

Length of sending data frame Length=1024bits
For corruption in channel, we are using uniform random distribution on BER 0.001, 0.0001 and 0.00001.

We are sending 1000 packet for this simulation.

B. experimental result

To analyze the result we use TCPTRACE tool. TCPTRACE tool can give information related to Throughput by analyzing the Pcap file, generated by simulation. Table 1 shows the throughput of network at different BER. If we compare the BER and ratio of throughput of NewReno and Improved TCP, it shows that less the corruption more the resemblance. It means if there is no corruption both protocol will show same behavior otherwise Improved TCP shows better performance. This is also implied through Figure 2 and Figure 3. Figure 1 is illustration of Traffic window at BER 0.0001. Improved TCP finishes sending packet at 6 ms due to its higher throughput and Figure 2 shows similar Traffic window initially, even afterward it’s Traffic window remains higher with respect to NewReno. in Figure 2 NewReno Traffic window graph is hidden due to overlap by Improved TCP Traffic control.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Throughput</th>
<th>BER=.0010</th>
<th>BER=.0001</th>
<th>BER=.00001</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newreno</td>
<td>4629bps</td>
<td>17811bps</td>
<td>115990bps</td>
<td></td>
</tr>
<tr>
<td>Improved TCP</td>
<td>20099bps</td>
<td>37335bps</td>
<td>116712bps</td>
<td></td>
</tr>
<tr>
<td>Ratio</td>
<td>4.34</td>
<td>2.096</td>
<td>1.006</td>
<td></td>
</tr>
</tbody>
</table>

Throughput of both protocols at different BER. Last row shows ratio of NewReno by Improved TCP

Figure 2: Traffic window plot at BER 0.0001

TABLE I. THE PROTOCOL CAPABILITY WITH DIFFERENT BER
V. CONCLUSION

This article presents improved TCP Traffic control mechanism to implement on wire/wireless network. It considers the corruption of channel; differentiate the wire and wireless, major hurdle of traditional TCP to implement universally on wire and wireless. Proposed algorithm shows better performance by identifying the packet loss due to corruption in network rather than Traffic in network.

VI. REFERENCES


AUTHORS:

Mr. A.Siles Balasingh M.Tech (I.T) He is currently working as Lecturer in Department of Computer Science and Engineering in St. Joseph College of Engineering and Technology, Dar Es Salaam, Tanzania, East Africa. He has 12 International publications to his credit and he is editorial member for 6 International Journals. He has guided more than 35 projects to final year B.E/B.Tech students with good industry and teaching experience. He is His areas of interest in Computer Networks, Computer Networks and security, neural networks and Bioinformatics Computer Architecture and Ethical hacking.

Mr. N.Surenth M.Tech (I.T) He is currently working as Lecturer in of Department in Computer Science and Engineering in St. Joseph College of Engineering and Technology, Dar Es Salaam, Tanzania, East Africa. He has guided more than 15 projects to final year B.E/B.Tech students and his area of interest in Computer Networks and Artificial Intelligence.