Int. J. Communications, Network and System Sciences, 2011, 4, 256-265
doi:10.4236/ijcns.2011.44031 Published Online April 2011 (http://www.SciRP.org/journal/ijcns)

A New Transmission Control Protocol for Satellite Networks

Liang Yu, Gang Zhou
College of Computer Science, Sichuan University, Chengdu, China
E-mail: *yuliang@scu.edu.cn*
Received February 16, 2011; *revised March* 16, 2011; *accepted March* 22, 2011

Abstract

According to technical statistics, current TCP protocols with approximately 80% Internet applications run on perform very well on wired networks. However, due to the effects of long propagation delay, great bandwidth asymmetry, high sporadic Bit Error Rate (BER) and etc., TCP performance degrades obviously on the satellite communication networks. To avoid the problems, TP-S, a novel transport control protocol, is introduced for satellite IP networks. Firstly, in order to increase the increment speed of Congestion Window (cwnd) at the beginning of data transmission, the traditional Slow Start strategy is replaced by a new strategy, known as Super Start. Secondly, a new packet lost discriminated scheme based on IP packets alternately sending with different priority is used in the protocol to decouple congestion decision from errors. Thirdly, bandwidth asymmetry problem is avoided by adopting Modified NACK (M-NACK) in receiving ends, which is sent periodically. In addition, the sending strategy in routers is also modified along with other's changes to support the protocol. Finally, the simulation experiments show that the new protocol can not only significantly enhance throughput performance, but also reduce sharply bandwidth used in the reverse path as compared with traditional TCP protocols and those protocols, which are recently proposed for satellite IP networks.

Keywords: Satellite Communication, TCP, Long Propagation Delay, Bandwidth Asymmetry

1. Introduction

With the rapid development of Internet, TCP protocols has great adjusted and improved. Since Van Jacobson propose four algorithms [1]: Slow Start, Avoiding Congestion, Fast-retransmission and Fast-restoration, three TCP protocols TCP-Tahoe, TCP-Reno [2] and TCP-New Reno [3] have gradually become the current main version of Internet. These TCP protocols have their own traits. TCP-Tahoe protocol hasn't Fast-restoration algorithm that means it will directly access to the Slow Start initial state when Fast-retransmission with lost data. TCP-Reno protocol immediately access to Fast-restoration stage after Fast-retransmission rather than return to the Slow Start stage. TCP-New Reno protocol has modified the Fast-restoration algorithms of TCP-Reno protocol, and it avoids the Congestion Window and Slow Start (ssthresh) from halving problem, which caused by many lost data in Sending Window. Whether TCP-Tahoe or TCP-New Reno protocol, these protocols are designed to achieve that the Error Rate should be lower than 10^{-8} in wired ground network under the assumption that congestion is the only reason caused data loss and damage. Obviously, the assumption is not suit for the satellite network [4]. Satellite Channel has the following characteristics: long propagation delay (Synchronous satellite channel Return-Transmission Time (RTT) is about 550 ms), great bandwidth asymmetry and high sporadic Bit Error Rate (BER) [5,6] .All these traits have serious impact on the transmission performance of TCP protocols [7-9].

The paper proposes a novel transmission control protocol which is suit for satellite network based on the traits of the satellite network. It not only can increase the Congestion Window value rapidly after the connection set up, but also can distinguish the specific reasons for lost data. Thereby, it adopts related control strategy of Sending Window. Sending-Client applies cycle Sent-Response strategy in order to solve reverse link bandwidth asymmetry problem. According to simulation and analysis, the new protocol not only can increase the throughput for the forward link, but also greatly reduces

bandwidth occupancy rate of the reverse link, compared with the traditional TCP protocols and transmission control protocol which is proposed according to traits of satellite links in recent year.

One of the most interesting aspects of TCP is a mechanism for congestion control. Recall that in the Internet, delay or packet loss is more likely to be caused by congestion than a hardware failure, and that retransmission can exacerbate the problem of congestion by injecting additional copies of a packet. To avoid congestion collapse, TCP uses changes in delay as a measure of congestion, and responds to congestion by reducing the rate at which it retransmits data. TCP's four intertwined congestion control algorithms are slow start, congestion avoidance, fast retransmit, and fast recovery.

1.1. Slow Start and Congestion Avoidance

TCP uses a special congestion control mechanism when starting a new connection or when a message is lost. Instead of transmitting enough data to fill the receiver's buffer (*i.e.*, the receiver's window size), TCP begins by sending a single message containing data. If an acknowledgement arrives without additional loss, TCP doubles the amount of data being sent and sends two additional messages. If both acknowledgements arrive, TCP sends four messages, and so on. The exponential increase continues until TCP is sending half of the receiver's advertised window. When one-half of the original window size is reached, TCP slows the rate of increase, and increases the window size linearly as long as congestion does not occur. The approach is known as slow start.

TCP's congestion control mechanisms respond well to increases in traffic. By backing off quickly, TCP is able to alleviate congestion. In essence, TCP avoids adding retransmissions when the Internet becomes congested. More important, if all TCPs follow the standard, the congestion control scheme means that all senders back off when congestion occurs and congestion collapse is avoided.

When the load offered to any network is more than it can handle, congestion builds up. The Internet is no exception. In this section we will discuss algorithms that have been developed over the past quarter of a century to deal with congestion. Although the network layer also tries to manage congestion, most of the heavy lifting is done by TCP because the real solution to congestion is to slow down the data rate.

In theory, congestion can be dealt with by employing a principle borrowed from physics: the law of conservation of packets. The idea is to refrain from injecting a new packet into the network until an old one leaves (*i.e.*, is delivered). TCP attempts to achieve this goal by dynamically manipulating the window size. The first step in managing congestion is detecting it. In the old days, detecting congestion was difficult. A timeout caused by a lost packet could have been caused by either noise on a transmission line or packet discard at a congested router. Consequently, most transmission timeouts on the Internet are due to congestion. All the Internet TCP algorithms assume that timeouts are caused by congestion and monitor timeouts for signs of trouble the way miners watch their canaries.

The slow start and congestion avoidance algorithms must be used by a TCP sender to control the amount of outstanding data being injected into the network. To implement these algorithms, two variables are added to the TCP per-connection state. The congestion window (cwnd) is a sender-side limit on the amount of data the sender can transmit into the network before receiving an acknowledgment (ACK), while the receiver's advertised window (rwnd) is a receiver-side limit on the amount of outstanding data. The minimum of cwnd and rwnd governs data transmission.

Another state variable, the slow start threshold (ssthresh), is used to determine whether the slow start or congestion avoidance algorithm is used to control data transmission, as discussed below.

Beginning transmission into a network with unknown conditions requires TCP to slowly probe the network to determine the available capacity, in order to avoid congesting the network with an inappropriately large burst of data. The slow start algorithm is used for this purpose at the beginning of a transfer, or after repairing loss detected by the retransmission timer.

IW, the initial value of cwnd, must be less than or equal to 2*SMSS bytes and must not be more than 2 segments.

We note that a non-standard, experimental TCP extension allows that a TCP may use a larger initial window (IW), as defined in Equation (1):

$$IW = \min (4 \times SMSS, \max (2 \times SMSS, 4380 \text{ bytes})) \quad (1)$$

With this extension, a TCP sender may use a 3 or 4 segment initial window, provided the combined size of the segments does not exceed 4380 bytes. We do not allow this change as part of the standard defined by this document. However, we include discussion of (1) in the remainder of this document as a guideline for those experimenting with the change, rather than conforming to the present standards for TCP congestion control.

The initial value of ssthresh may be arbitrarily high (for example, some implementations use the size of the advertised window), but it may be reduced in response to congestion. The slow start algorithm is used when cwnd < ssthresh, while the congestion avoidance algorithm is

used when cwnd > ssthresh. When cwnd and ssthresh are equal the sender may use either slow start or congestion avoidance.

During slow start, a TCP increments cwnd by at most SMSS bytes for each ACK received that acknowledges new data. Slow start ends when cwnd exceeds ssthresh (or, optionally, when it reaches it, as noted above) or when congestion is observed.

During congestion avoidance, cwnd is incremented by 1 full-sized segment per round-trip time (RTT). Congestion avoidance continues until congestion is detected. One formula commonly used to update cwnd during congestion avoidance is given in Equation (2):

$$\text{cwnd} \mathrel{+}= \text{SMSS} \times \text{SMSS}/\text{cwnd} \quad (2)$$

This adjustment is executed on every incoming non-duplicate ACK. Equation (2) provides an acceptable approximation to the underlying principle of increasing cwnd by 1 full-sized segment per RTT. (Note that for a connection in which the receiver acknowledges every data segment, (2) proves slightly more aggressive than 1 segment per RTT, and for a receiver acknowledging every-other packet, (2) is less aggressive.)

Another acceptable way to increase cwnd during congestion avoidance is to count the number of bytes that have been acknowledged by ACKs for new data. (A drawback of this implementation is that it requires maintaining an additional state variable.) When the number of bytes acknowledged reaches cwnd, then cwnd can be incremented by up to SMSS bytes. Note that during congestion avoidance, cwnd must not be increased by more than the larger of either 1 full-sized segment per RTT, or the value computed using Equation (2).

When a TCP sender detects segment loss using the retransmission timer, the value of ssthresh must be set to no more than the value given in Equation (3):

$$\text{ssthresh} = \max(\text{FlightSize}/2,\ 2 \times \text{SMSS}) \quad (3)$$

As discussed above, FlightSize is the amount of outstanding data in the network.

Furthermore, upon a timeout cwnd must be set to no more than the loss window, LW, which equals 1 full-sized segment (regardless of the value of IW). Therefore, after retransmitting the dropped segment the TCP sender uses the slow start algorithm to increase the window from 1 full-sized segment to the new value of ssthresh, at which point congestion avoidance again takes over.

1.2. Fast Retransmit/Fast Recovery

To handle packet loss, transport protocols use positive acknowledgement with retransmission. Whenever a frame arrives intact, the receiving protocol software sends a small acknowledgement (ACK) message that reports successful reception, the sender takes responsibility for ensuring that each packet is transferred successfully. Whenever it sends a packet, the sending-side protocol software starts a timer. If an acknowledgement arrives before the timer expires, the software cancels the timer; if the timer expires before an acknowledgement arrives, the software sends another copy of the packet and starts the timer again. The action of sending a second copy is known as retransmitting, and the copy is commonly called a retransmission.

A TCP receiver should send an immediate duplicate ACK when an out-of-order segment arrives. The purpose of this ACK is to inform the sender that a segment was received out-of-order and which sequence number is expected. From the sender's perspective, duplicate ACKs can be caused by a number of network problems. First, they can be caused by dropped segments. In this case, all segments after the dropped segment will trigger duplicate ACKs. Second, duplicate ACKs can be caused by the re-ordering of data segments by the network (not a rare event along some network paths. Finally, duplicate ACKs can be caused by replication of ACK or data segments by the network. In addition, a TCP receiver should send an immediate ACK when the incoming segment fills in all or part of a gap in the sequence space. This will generate more timely information for a sender recovering from a loss through a retransmission timeout, a fast retransmit, or an experimental loss recovery algorithm, such as NewReno.

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.

After the fast retransmit algorithm sends what appears to be the missing segment, the "fast recovery" algorithm governs the transmission of new data until a non-duplicate ACK arrives. The reason for not performing slow start is that the receipt of the duplicate ACKs not only indicates that a segment has been lost, but also that segments are most likely leaving the network (although a massive segment duplication by the network can invalidate this conclusion). In other words, since the receiver can only generate a duplicate ACK when a segment has arrived, that segment has left the network and is in the receiver's buffer, so we know it is no longer consuming network resources. Furthermore, since the ACK "clock" is preserved, the TCP sender can continue to transmit new segments (although transmission must continue using a

reduced cwnd).

The fast retransmit and fast recovery algorithms are usually implemented together as follows.

1) When the third duplicate ACK is received, set ssthresh to no more than the value given in Equation (3).
2) Retransmit the lost segment and set cwnd to ssthresh plus 3*SMSS. This artificially "inflates" the congestion window by the number of segments (three) that have left the network and which the receiver has buffered.
3) For each additional duplicate ACK received, increment cwnd by SMSS. This artificially inflates the congestion window in order to reflect the additional segment that has left the network.
4) Transmit a segment, if allowed by the new value of cwnd and the receiver's advertised window.
5) When the next ACK arrives that acknowledges new data, set cwnd to ssthresh (the value set in step 1). This is termed "deflating" the window.

This ACK should be the acknowledgment elicited by the retransmission from step 1, one RTT after the retransmission (though it may arrive sooner in the presence of significant out-of-order delivery of data segments at the receiver).

Additionally, this ACK should acknowledge all the intermediate segments sent between the lost segment and the receipt of the third duplicate ACK, if none of these were lost.

How to improve the performance of TCP protocols in satellite network has become the focus. Through analyzing theoretical and experimental, the following are factors about TCP performance that impacted on by satellite channel:

Long propagation delay

- The most throughput of TCP is restricted by RTT and Winmax. When most sending window value is 64Kbytes and RTT is 550 ms, the most throughput of TCP is only 1 Mb/s.
- The TCP congestion window increases very slowly in slow start and congestion avoiding stage so that TCP's performance is not efficiency.

High Bit Error Rate

- Sporadic Bit Error Rate is prime formation in GEO channel environment. TCP judge the network is in congestion state when BER occur in channel. So it not only reduces the efficiency of TCP transmission but also waste bandwidth available in satellite channel.
- When the satellite signal is obscured or multi-path and eclipse become serious, it will occur unexpected Bit Error. All make TCP's performance unstable.

Link bandwidth asymmetry

- The ACKs congestion that caused by narrow bandwidth in reverse channel result in throughput of forward link decline.

The satellite and network working group in Internet Engineering Task Force (IETF) have established some Request for Comments (RFC) [10-12] in order to improve TCP protocols performance in satellite network. Many research organizations and institutions dedicate to this field, so a large number of new solutions are emerging. Now many TCP protocols solutions have been proposed that can be sort by TCP protocol modification program, other layers protocol modification program, agent design program and specialized transmission control protocol solutions program.

1) TCP protocol modification program adopts to expand the largest sending window, expand the initial sending window, TCP header compression, congestion instructions, T/TCP, multiple TCP connection, SACK and a NACK to adapt to satellite network data transmission.
2) Other layer protocols modification programs are concentrate to link layer include FEC and Auto-Retransmission mechanism. The principle of Auto-Retransmission mechanism is that it isolates TCP protocol duplication responses information when data loss and directly retransmits lost data in link layer. All of this to make TCP protocol believes satellite channel is without fault and the delay increased slightly. The shortage is redundant function between link layer and transport layer and the design of link layer is complex. In addition, it maybe results in waited overtime.
3) Agent solution program adopt agent manner (TCP-Spoofing, TCP-Splitting) [13-15] which refer to TCP-Spoofing, Snoop-TCP, TCP-Splitting and I-TCP. The core idea of agent manner is that it sends fake response information at satellite network gateway so that the users can not detect the long propagation delay in satellite link.
4) Special transmission control protocol solutions include:

Fast Start [16]

Fast start is instead of Slow Start strategy in connection beginning stage by adopting last link of TCP transmission window size and repeating the latest TCP transmission data rate. In order to avoid network congestion, the IP packet should be set with low priority, which is sent at beginning state.

TCP-Peach [17] and TCP-Peach+ [18]

TCP-Peach protocol adopts to send "Dummy" data segment of redundancy data information to detect the available bandwidth in start and Retransmission-Restoration stage. In order to reduce the bandwidth occupied in re-

verse link, STP protocol send a periodic "POLL" packet to inquire the receiver about receiving the data situation.

STP and STP+ [19]

In order to reduce the bandwidth occupied in reverse link, STP protocol send a periodic "POLL" packet to inquire the receiver about receiving the data situation. Through periodic response and NACK, it reduces the occupied resources in backward bandwidth

XCP [20]

It uses the congestion instruction method and adds the current congestion window value, the time for data return and back, the reservation bandwidth. In addition, it demands mid router to add algorithm and modify the reservation bandwidth information.

TCP-Westwood [21]

In order to improve the network bandwidth utilization, TCP Westwood protocol achieves to estimate end-to-end available bandwidth through continuous monitoring the response information.

2. Novel Protocol Module

The Transmission Control Protocol (TCP) is the major transport protocol in the TCP/IP protocol suite. TCP provides application programs with a reliable, flow controlled, full-duplex, stream transport service. After requesting TCP to establish a connection, an application program can use the connection to send or receive data; TCP guarantees to deliver the data in order without duplication. Finally, when the two applications finish using a connection, they request that the connection be terminated.

TCP on one computer communicates with TCP on another computer by exchanging messages. All messages from one TCP to another use the TCP segment format, including messages that carry data, acknowledgements, and window advertisements, as well as messages used to establish and terminate a connection. Each TCP segment travels in an IP datagram.

In general, transport protocols use a variety of mechanisms to insure reliable service. TCP has a particularly complex combination of techniques that have proven to be extremely successful. In addition to a checksum in each segment, TCP retransmits any message that is lost. To be useful in the Internet where delays vary over time, TCP's retransmission timeout is adaptive TCP measures the current round-trip delay separately for each connection, and uses a weighted average of the round- trip time to choose a timeout for retransmission.

Novel protocol is composed with Super Start of sending client, congestion avoiding, distinguishing lost, congestion restoration strategy, discarded data strategy of router, and periodic response strategy of receiving terminal.

2.1. Novel Protocol Module and Architecture

As shown in **Figure 1**, the sending terminal include following algorithm in novel protocol: super start, congestion avoiding, distinguishing lost, congestion restoration strategy and etc. The super start strategy has replaced the slow start strategy in TCP-Reno which is widely applied in current network and in TCP-New Reno. Distinguishing lost and congestion restoration are new strategies, but congestion avoiding is same as TCP-Reno, TCP-New Reno.

2.2 Super Start Strategy

Super start strategy is shown in **Figure 2**; the congestion window value of sending terminal is set half of the receiving window value of receiving terminal, *i.e*. cwnd = rwnd/2. Sending terminal send a TCP data segment every interval time τ and the priority of sending data segment are alternative switch. Interval time τ is calculated in accordance with the following formula:

$$\tau = 2 \cdot RTT/cwnd \quad (4)$$

The return time is estimated in link set up process. At this stage a priority bit ("pri") is stand for the priority of packet in IP header's TOS segment, besides a state bit ("start") is stand for the state of sending data. The IP packet set with start = 1 in super start stage and the sending terminal access to congestion avoiding state from ending super start strategy when receiving first M-NACK information. The later IP packet is sent with start = 0, in addition the receiver send an M-NACK every RTT.

The specific algorithm of super start strategy can be illustrated by following examples. Assuming that TCP segment begin to send data when t = 0, thereby the receiver receive first segment and begin to count time after

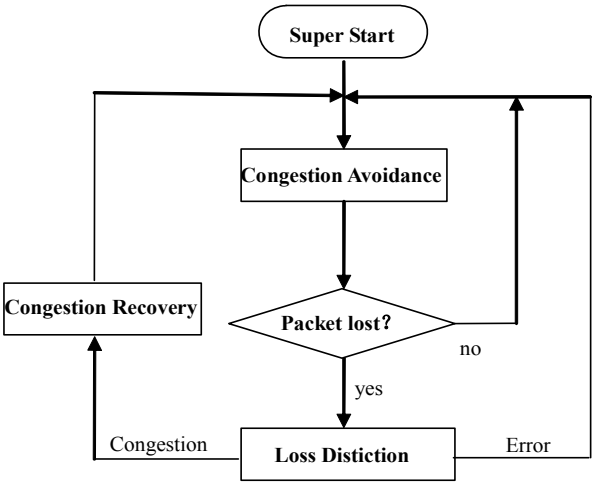

Figure 1. A novel protocol architecture.

about RTT/2. The receiving start to send the first M-NACK when t ≈ 3·RTT/2, *i.e.* after one RTT. After RTT/2, the receiver receives the M-NACK and increase by the number of cwnd data which are received by receiver. At the same time it ends the super start stage and enter to congestion avoiding state.

M-NACK include all state of receiver like hoping to get the maximum sequence value for segment, hoping to get the maximum sequence value of segment and packets lost list.

```
Super Start()
    {
      start = 1;
      cwnd = rwnd/2;
      τ = RTT/cwnd;
      for (i = 1 to rwnd)
      {
        if (pri = = 1)
          pri = 0;
          else
          pri = 1;
        send(Data_Segment);
        wait(τ);
      }
      start = 0;
      wait for ACK;
      if (ACK arrives)
      {
        cwnd = cwnd + num_received;
          τ = RTT/cwnd;
       }
    }
```

Figure 2. Super start strategy.

2.3. Distinguishing Lost Strategy

The IP packet sent by alternating high and low priority with start = 0 after accessing to congestion avoiding stage. The transmitter judge whether data are lost according to M-NACK information. As shown in **Figure 3**, when transmitter finds that data lost, it judges why data loss according to how many high and low priority packet lost.

When congestion occurs in network, the router firstly discards the low priority data in queue. Liking Formula (5), the number of lost low priority packets is same as all lost packets.

$$Low\_pri = lost\_num \tag{5}$$

When there is serious congestion in network and all low priority packets have been lost, the high priority packet will begin to discard. As shown in Formula (6), the number of lost packets is more than half of congestion window value.

```
Loss Distinction()
{
if(lost num = = 1)
{
pri = 1;
send(Missing_Data);
}
else
{
if((low-pri = = lost num) or (lost num > cwnd/2))
{
goto   Congestion Recovery();
}
else
{
while(lost num > 0)
{
send(missing_packet);
lost num = lost num－1;
wait(τ);
          }
        }
    }
}
```

Figure 3. Distinguishing lost strategy.

$$Lost\_num > \text{cwnd}/2 \tag{6}$$

When above two situation occur, the transmitter considers the congestion occurs in network, so it adopts congestion restoration strategy. As shown in **Figure 4**, congestion window value has halved. It recounts interval time τ and sends lost data in low and high priority interval time. After sending all lost data, it ends congestion restoration stage and accesses to congestion avoiding stage.

The transmitter considers that data lost due to bit error except above two situations. As shown in **Figure 3**, the transmitter immediately sends lost data and the sending window value should not be changed. It access to congestion avoiding stage and send new data after sending all lost data.

When the network congestion situation is not serious and only a single packet loss, the whole network performance will not obviously decrease. Therefore, when finding a single packet loss, the transmitter considers that lost data are caused by bit error and will not reduce congestion window value.

2.4. Router Packet Discarded Strategy

According to the priority ("pri") and state value ("start") of header, it can be sorted by four categories: 1) start = 1, pri = 0, 2) start = 1, pri = 1, 3) start = 0, pri = 0, 4) start = 0, pri = 1. When congestion occurs, router will


```
Congestion Recovery()
{
   if((now－last_time) > = 2*RTT)
   {
      cwnd = cwnd/2;
         last_time = now;
      τ = RTT/cwnd;
   }
   for(i = 1 to lost_num)
   {
      if(pri = = 1)
         pri = 0;
         else
         pri = 1;
      send(Missing_Data);
      wait(τ);
   }
}
```

Figure 4. Congestion restoration strategy.

```
Packet Drop()
{
   if (start = 1)
   {
   if (pri = 0)
      drop (packet);
   else
      drop (packet);
}
else
{
   if (pri = 0)
      drop (packet);
   else
      drop (packet);
}
```

Figure 5. Data discarded strategy.

apply following data discarded strategy like **Figure 5**.

1) Firstly discarding low priority with high beginning packets, *i.e*. start = 1, pri = 0;
2) When the above packets can not be found, high priority with high beginning packets will be discarded, *i.e*. start = 1, pri = 1;
3) When high beginning packets are not exist, low priority with low beginning packets will be discarded, *i.e*. start = 0, pri = 0;
4) When above all packets are not exist, high priority with low beginning packets will be discarded, *i.e*. start = 0, pri = 1;

Due to the IP data header sent at super start stage with state value start = 1, router adopts above data discarded strategy which can avoid from data loss in other link caused by network congestion when new connection set up.

3. Simulation and Analysis

In order to compare with TCP-Reno, TCP-New Reno, TCP-Peach, TCP Westwood, XCP and STP protocols. We take advantage of the current popular NS-2 network simulation software to do simulation according to simulation topology proposed in literature user connect to synchronous satellites through ground gateway which can aggregate N connections. The capacity of gateway cache is 50 packets; the bandwidth of ground connection is 10 Mbits/s, the forward bandwidth ($B_{forward}$)of satellite link is 10 Mbits/s, RTT is 550 ms, every TCP segment length is 1000 bytes, the receiving window value of receiver is 64 packets, simulation time is 1000 times than RTT, which is 550 s.

3.1. The Performance of Single-link Comparison

When only one link connects in network and reverse link bandwidth ($B_{backward}$) is 10 Mbits/s, it sends files with different size and data transmission time for every protocol. From **Figure 6**, we can see TP-S transmission time is shortest when file is less than 64 Kbytes and its transmission time is nearly same as TCP-Peach when file is more than 64 Kbytes. In the other protocols, STP transmission time is longest but TCP Westwood, TCP-NewReno and TCP-Reno all are same. Evidently, super start strategy of TP-S can quickly increase transmitter's congestion window, thus the smaller files can be transmitted with short time.

When only one link connects in network, all the average throughput of forward link for every protocol is shown in **Figure 7**. Bit error is the only reason result in

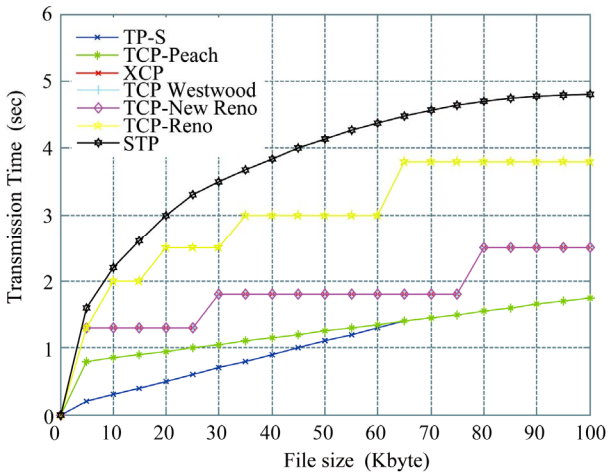

Figure 6. Compared performance of file transmission with single-link without bit error.

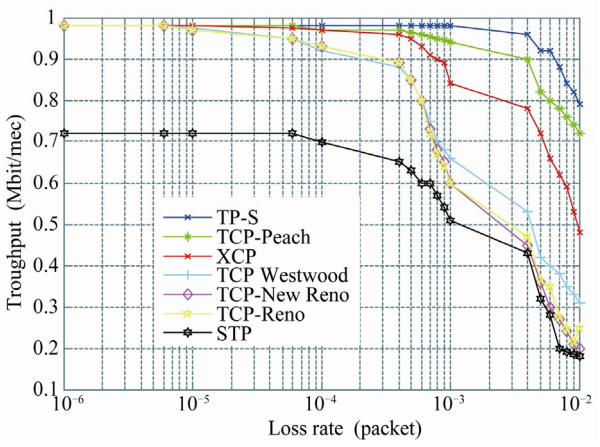

Figure 7. Compared throughput of forward link in single-link with random bit error.

data lost in such situation. It can be seen from graph that TP-S protocol average throughput is 9.5 Mbits/s or so which is similar with others when the packet lost rate is less than 10^{-4}. The throughput of STP protocol is lower. With packet lost rate is gradually increasing, TCP-Reno, TCP-NewReno, TCP-Peach, TCP Westwood, XCP and STP protocols' throughput decrease rapidly. When packet lost rate 10^{-2}, the throughput of TCP-Reno, TCP-New Reno and STP only are 0.2 Mbits/s, TCP Westwood is 0.3 Mbits/s, XCP is 0.5 Mbits/s，TCP-Peach is 0.72 Mbits/s. But the performance of TP-S degrade slightly, whose throughput begin to decrease until packet lost rate is more than 10^{-3}. When the packet lost rate as high as 10^{-2}, throughput still remain at 0.8 Mbits/s level.

We can see that the occupied backward bandwidth of TCP-Reno, TCP-NewReno, TCP-Peach, TCP Westwood and XCP protocols are more than 35 Kbits/s when the packet lost rate is less than 10^{-4}. With packet lost rate is gradually increasing, backward bandwidth decrease. All backward bandwidth are more than 5 Kbit/s when packet lost rate is as high as 10^{-2}. Due to adopt periodic sending response information strategy, STP and TP-S protocols backward bandwidth occupied mainly rely on sending periods. With packet lost rate growing, backward bandwidth of TP-S protocol increase slowly. When the packet lost rate is as high as 10^{-2}, the backward bandwidth is not higher than 1 Kbits/s.

3.2. The Performance of Multi-Link Comparison

When the number of connection $N > 20$, there is not only link error but also congestion in network. It caused by network data maximum input rate is more than satellite forward link bandwidth. How to calculate Maximum input rate ($In_{\max}$) is shown in the following formula:

$$In_{\max} = N \cdot \text{rwnd} \cdot 8/\text{RTT} \quad (7)$$

That can be deduced by above formula, when rwnd = 64，RTT = 550 ms and $N = 10$, forward Link is at the saturation state, that is the utilization of forward link bandwidth will be maximum without congestion in network. When the number of connection $N = 20$ and backward bandwidth $B_{backward} = 10$ Mbits/s, the average throughput of forward link for every protocol is shown in **Figure 8**. It shows that throughput of TP-S is not the highest which value is close to 9 Mbits/s but lower than the TCP Westwood, STP and XCP protocol when packet lost rate is less than 10^{-4}. But with increasing of packet lost rate, TP-S throughput has not declined. It has remained at close to 9 Mbits/s level, but the performance of other agreements are obviouly decreased. When packet lost rate is 10^{-2}, throughput of TCP-Reno, TCP-NewReno and STP is only about 3.5 Mbits/s. The throughput of XCP and TCP Westwood protocols are 4.5 Mbits/s and 5.5 Mbits/s, even if TCP-Peach protocol is only 7.5 Mbits/s.

The situation of backward bandwidth occupied when connection number $N = 20$ is shown in **Figure 9**. When packet lost rate is less than 10^{-3}, the occupied backward bandwidth of TCP-Reno, TCP-NewReno, TCP-Peach, TCP Westwood and XCP protocols are more than 300 Kbits/s. Although backward bandwidth of these protocols continuously decline with increasing of packet lost rate, backward bandwidth still more than 100 Kbit/s especially TCP-Peach protocol which backward bandwidth is nearly 300 Kbits/s, even if the packet lost rate is 10^{-2}. As a result of adopting periodic sending response information strategy, backward bandwidth of STP and TP-S protocol has not seriously changed with increasing of packet lost rate. Backward bandwidth of TP-S protocol occupoes very low with remained at 12 Kbits/s level.

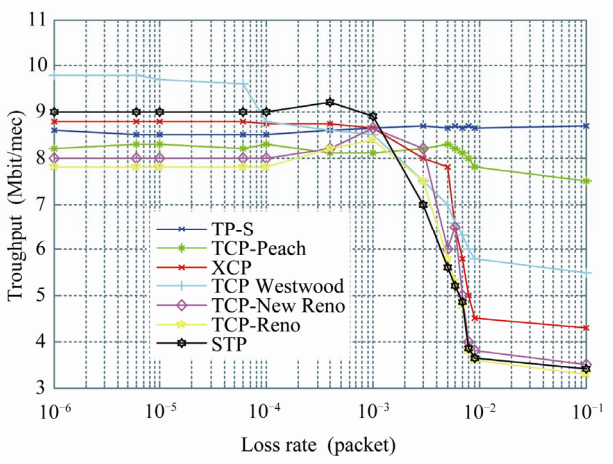

Figure 8. Compared with the performance of forward link throughput in 20 connections with random error.

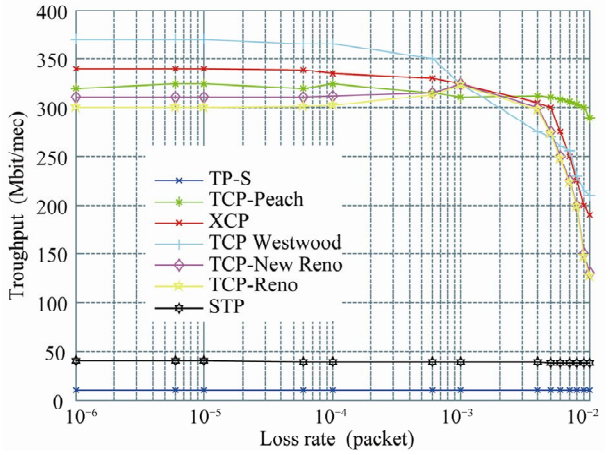

Figure 9. Compared with the performance of backward link throughput in 20 connections with random error.

3.3. The Performance of Backward Link Restricted Bandwidth Comparison

When network connection number N = 10, forward link is in saturation state and the average throughput of forward link for every protocols are shown in below. As can be seen from the graph, the performance of TCP-Reno, TCP-NewReno, TCP-Peach, TCP Westwood and XCP protocols are impacted by congestion of backward link response information. When backward link bandwidth is 200 Kbits/s, throughputs of these protocols are only about 8 Mbits/s. With backward link bandwidth is continuous declining; the congestion of response information become serious and throughput of forward link also decline. When backward link bandwidth is only 50 Kbits/s, performance of protocol drop to less than 6 Mbits/s. As backward link bandwidth which TP-S and STP need is much less than 50 Kbits/s, especially TP-S protocol only need 5Kbits/s, the performance of forward link will not be impacted by backward link bandwidth for TP-S and STP protocols, throughput of TP-S maintain at 9.5 Mbits/s or so. When network connection number N = 10, the situation of backward link bandwidth occupied for every protocols are shown in **Figure 10**.

4. Conclusions

In order to resolve the problem that TCP protocols has poor performance in satellite network, the paper proposed a novel satellite network transmission control protocol. The protocol adopts a novel window growth strategy to accelerate the speed of increasing congestion window after connection established. In order to distinguish specific reasons for data lost, the protocol adopts interval sending strategy for different priority IP packets and judges network performance according to data lost.

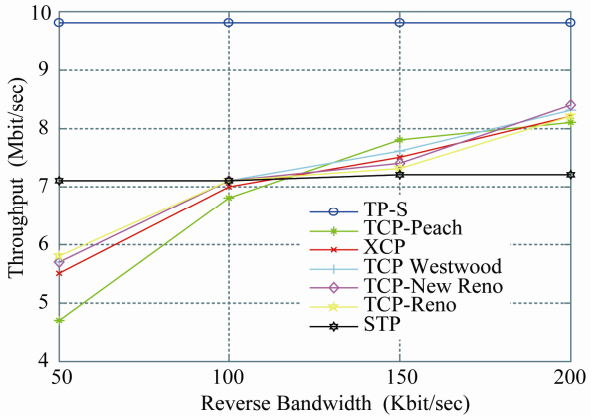

Figure 10. Compared with throughput of forward link under backward link bandwidth restricted.

In addition, periodic sending response information strategy is used in this protocol to solve bandwidth asymmetry problem in forward and backward satellite link. Compared with TCP protocols and some satellite network transmission control protocols proposed in recent year through simulation, the protocol not only can obviously enhance the throughput of forward link but also greatly reduces the bandwidth occupancy rate in backward link. Main feature of the protocol is that it adopts different priority data interleaving sending principle to send data. Super start can rapidly enhance sending rate at the beginning of connection. Data lost judgment and congestion restoration strategy can effectively distinguish the specific reasons for data lost and take related transmission rate control strategy. Periodic sending response strategy can reduce occupancy for backward link bandwidth to solve satellite link bandwidth asymmetry problem. Router adopts a simple and easy data discarded strategy with different priority. The protocol can rapidly enhance sending rate at the beginning of connection and effectively distinguish reasons for data lost. It maintains very high throughputs of forward link in channel environment with random and unexpected error. Backward link bandwidth occupied with rare resource can solve the asymmetry problem of satellite link bandwidth. Router algorithm is simple, low design requirements and facility realization in network.

5. References

[1] V. Jacobson, "Congestion Avoidance and Control," *Computer Communication Review*, Vol. 18, No. 4, 1988, pp. 314-329. doi:10.1145/52325.52356

[2] M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control," RFC 2581, 1999, pp. 1-5.

[3] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 2582, 1999.

[4] C. Partridge and T. Shepard, "TCP/IP Performance Over Satellite Links," *IEEE Network Magazine*, Vol. 11, No. 5, 1997, pp. 44-49. doi:10.1109/65.620521

[5] A. Jamalipour and T. Tung, "The Role of Satellites in Global IT: Trends and Implications," *Personal Communications, IEEE*, Vol. 8, No. 3, 2001. pp. 5-11.

[6] M. Allman, D. Glover and L. Sanchez, "Enhancing TCP Over Satellite Channels Using Standard Mechanisms," RFC 2488, 1999.

[7] T. V. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Transactions on Networking*, Vol. 5, No. 3, 1997, pp. 336-350. doi:10.1109/90.611099

[8] E. Lutz, *et al*., "The Land Mobile Satellite Communication Channel-Recording, Statistics, and Channel Model," *IEEE Transactions on Vehicular Technology*, Vol. 40, No. 2, 1991, pp. 375-386. doi:10.1109/25.289418

[9] H. Balakrishnan, V. Padmanabhan and R. Katz, "The Effects of Asymmetry on TCP Performance," *Proceedings of the* 3*rd ACM/IEEE Mobile Computing Conference*, Budapest, 26-30 September 1997, pp. 77-89.

[10] J. Border, M. Kojo and J. Griner, "Performance Enhancing Proxies Intended to Mitigate Link Related Degradations," RFC 3135, June 2001.

[11] M. Allman, S. Dawkins and D. Glover, "Ongoing TCP Research Related to Satellites," RFC 2760, 2000.

[12] M. Mathis, J. Mahdavi and S. Floyd, "TCP Selective Acknowledgment Options," RFC 2018, 1996.

[13] H. Balakrishnan, S. Seshan, E. Amir and R. H. Katz, "Improving TCP/IP Performance Over Wireless Networks," *Proceedings of ACM Mobile computing*, California, 13-15 November 1995, pp. 2-15.

[14] M. Mario, R. Michele and M. Giacomo, "PETRA: Performance Enchancing Transport Architecture for Satellite Communications," *IEEE Journal on Selected Areas in Communications*, Vol. 22, No. 2, 2004, pp. 320-332. doi:10.1109/JSAC.2003.819981

[15] M. Luglio, *et al*., "On-Board Satellite 'Split TCP' Proxy," *IEEE Journal on Selected Areas in Communications*, Vol. 22,No. 2, 2004, pp. 362-370.

[16] V. N. Padmanabhan and R. Katz, "TCP Fast Start: A Technique for Speeding Up Web Transfer," Proceedings of *IEEE GLOBECOM'98 Internet*,Sydney, November 1998, pp. 41-46.

[17] I. F. Akyildiz, M. Giacomo and P. Sergio, "TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks," *IEEE/ACM Transactions on Networking*, Vol. 9, No. 3, 2001, pp. 307-321.

[18] I. F. Akyildiz, X. Zhang, and J. Fang, "TCP-Peach+: Enhancement of TCPPeach for Satellite IP Networks," *IEEE Communications Letters*, Vol. 6, No. 7, 2002, pp. 303-305. doi:10.1109/LCOMM.2002.801317

[19] T. R. Henderson and R. H. Katz, "Transport Protocols for Internet Compatible Satellite Networks," *IEEE Journal on Selected Areas in Communications,* Vol. 17, No. 2, 1999, pp. 326-344. doi:10.1109/49.748815

[20] D. Katabi, M. Handley and C. Rohrs, "Congestion Control for High Bandwidth Delay Product Networks," *Proceedings of the ACM SIGCOMM Conference on Applications*, Pittsburgh, 19-23 August 2002, pp. 1-14. doi:10.1145/633025.633035

[21] C. Casetti, *et al*., "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Link," *Proceedings of Mobile computing*, Rome, 16-21 July 2001, pp. 287-297.

