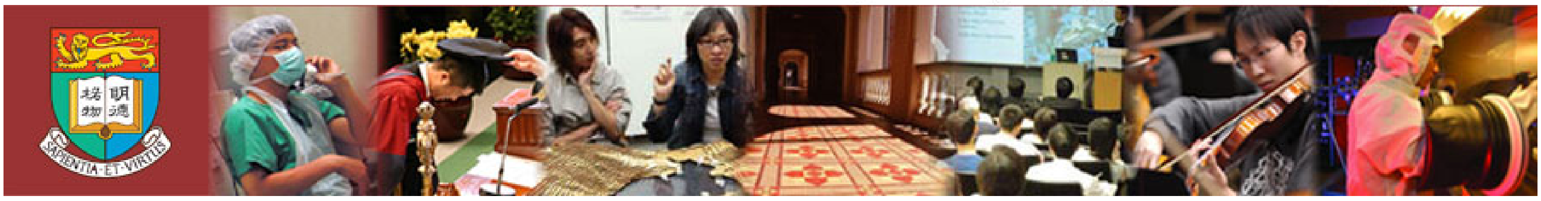

TCP-Swift: An End-Host Enhancement Scheme for TCP over Satellite IP Networks

Kui-Fai Leung and Kwan L. Yeung

Department of Electrical and Electronic Engineering
The University of Hong Kong
Hong Kong, PRC.
Tel: (852) 2857-8493 Fax: (852) 2559-8738

Abstract: *A new transport layer protocol called TCP-Swift is proposed for enhancing the TCP performance over satellite IP networks. TCP-Swift replaces the conventional TCP slow start and fast recovery algorithms by speedy start and speedy recovery. With speedy start, a TCP-Swift sender opens up its congestion window in only two round trip times. This significantly shortens the time needed in probing the network for equilibrium state. With speedy recovery, we can infer the cause of a packet loss by observing the ACK stream received at the sender. If the loss is due to wireless transmission error, the sender's congestion window can be re-opened up more aggressively to fully utilize the available satellite link bandwidth. We show that TCP-Swift outperforms existing TCP schemes by simulations.*

I. Introduction

The market need for Internet access via broadband satellite networks is increasing. Although TCP has been designed, improved and tuned to work efficiently in the wireline network, extending it to satellite networks is problematic. This is mainly due to the following satellite link characteristics.

Large bandwidth-delay product. TCP spends a long time in its slow start phase because of long latency of the satellite link. Current TCP applications, like HTTP, consist of many transfers of small files. Very likely such transfers will be completed within the slow start phase. In other words, TCP may not be able to fully utilize the available bandwidth of the network.

Transmission errors in satellite links. TCP interprets packet loss as a signal of network congestion, which then triggers the congestion control mechanism to reduce the transmission window size. Such interpretation is no longer valid when applied to satellite links. Due to factors like atmospheric conditions, RF interference, weak signal and so on, satellite links usually experience high bit error rate (BER).

Channel asymmetric. In many satellite systems, the bandwidths on the forward and backward channels are asymmetric. This is mainly due to cost limitation, increased transmitter power, and antenna size requirements for high bandwidth transmission. Using slower reverse channels can make more cost effective designs and save the scarce satellite bandwidth.

Many studies [1-2,4,6-8] have been carried out to enhance the performance of TCP over satellite IP networks. Recently an efficient scheme called TCP-Peach [4] was designed. It adopts two new algorithms, *sudden start* and *rapid recovery.* Dummy segments are used in these algorithms for fast probing the availability of the network resources.

In this paper, a new transport layer protocol called TCP-Swift is proposed based on TCP-Peach. TCP-Swift replaces the conventional TCP slow start and fast recovery algorithms by speedy start and speedy recovery. With speedy start, a TCP-Swift sender opens up its congestion window size in only two round trip times (RTTs). With speedy recovery, we can infer the cause of a packet loss by observing the ACK stream received at the sender. If a packet loss is due to wireless transmission error, the sender's congestion window can be re-opened up more aggressively to fully utilize the available bandwidth. In the next section, we first summarize our two major observations on TCP-Peach. Then TCP-Swift is proposed in Section III based on these observations. The performance of TCP-Swift is evaluated in Section IV. Finally we conclude the paper in Section V.

II. Observations on TCP-Peach

A. *Using Dummy Segment*

The dummy segments used in TCP-Peach [4] are low-priority segments generated by the sender as a copy of the *last* transmitted data segment. Efforts on transmitting dummy segments into the network can be treated as the network overhead since they do not carry any new information to the receiver (assuming the last transmitted data segment is correctly received). Dummy segments may be transmitted through the network but dropped in subsequent routers along the transmission path. Hence dummy segments may cause network inefficiency.

B. *Window Expanding Period (WEP)*

The information that conveyed by ACKs for data segments after rapid recovery [4] is not fully utilized. TCP-Peach sends a number of dummy segments to probe the availability of the network resources and expand its congestion window size according to the receipt of ACK for these dummy segments. Fig. 1 shows the corresponding window expanding period (WEP). WEP lasts for one RTT which is the period from the time the sender exiting/quitting the rapid recovery algorithm to receiving the ACK of the last data segment sent in the rapid recovery [4].

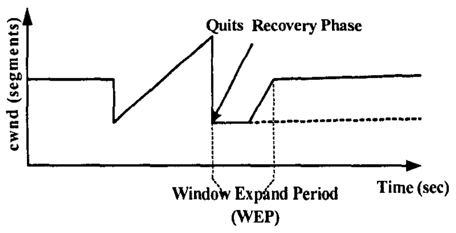

Fig. 1. Window Expanding Period

In rapid recovery, $cwnd_0/2$ data segments and $cwnd_0$ dummy segments are injected into the network. During the WEP, the TCP sender will receive dummy ACKs (i.e. ACKs for dummy segments) and data ACKs (i.e. ACKs for regular data segments) respectively. TCP-Peach starts increasing its congestion window upon receiving the $(cwnd_0/2+1)$-*th* dummy ACK. Similar to the implementation of TCP-Reno, TCP-Peach enters the congestion avoidance phase after exiting the rapid recovery. Thus the congestion window is increased by one for each RTT during the congestion avoidance phase.

TCP is designed to adapt its transmission rate to bandwidth along the sender-to-receiver's transmission path. The return of data ACKs (for those data segments transmitted during rapid recovery) during WEP does indicate to the sender that there are still resources available in the network. As such, the congestion window should be further expanded for each data ACK received during WEP.

III. Our Approach: TCP-Swift

A. *Outstanding Segment & Speedy Start*

To increase the network bandwidth utilization, we replace the dummy segments by outstanding segments for probing the available network resources. Dummy segment is a copy of the last transmitted data segment, whereas outstanding segments are chosen randomly from the set of packets/segments that have been sent but not-yet-acknowledged. So they serve as a backup in case the earlier sent packets are lost. This gives some extra loss repairing capability as compared with the dummy segment approach. Please note that the outstanding segment concept and the NIL segments in [9] share the same idea.

The priority implementation of outstanding segments can be supported in IP layer by the Type of Service (TOS) option in IPv4. Some recent commercial routers, like Cisco 7000 series and 12000 series emerge to support IP TOS.

The speedy start algorithm is summarized in Fig. 2 for replacing the original sudden start of TCP-Peach.

```
Speedy Start ( )

    cwnd = 1;
    τ = RTT / rwnd;
    send(data_segment);      //the first data segment;
    for ( i= 1 to rwnd – 1)
        wait (τ) ;
        send(outstanding_segment);
    end;
end:
```

Fig. 2. Speedy start algorithm

B. *Congestion Window Increment Function & Speedy Recovery*

To improve the efficiency of congestion window increment in WEP for a large bandwidth-delay product transmission path (like satellite links). We propose that TCP sender should be aggressive in increasing its transmission rate in order to fill up the available transmission pipe.

In our TCP-Swift, the resulting algorithm is called speedy recovery, as detailed in Fig. 3. (The definitions of variables used are described in Table 1.) A *congestion window increment function* is added to the speedy recovery. The operation of this function is described as follows. When a sender exits from the rapid recovery in Fig. 1 (i.e. also the start time of WEP), the sequence number of the last data segment sent is recorded in a variable *swift_wep_recv*. During WEP, in case of the arrival of dummy ACKs, the sender behaves like TCP-Peach. For each data ACK arrived, the sender increases its congestion window by one until the ACK corresponding to the value of *swift_wep_recv* is received. Therefore, the congestion window size opens up more quickly and aggressively than rapid recovery of TCP-Peach. In short, speedy recovery differs from TCP-Peach's rapid recovery algorithm in two ways:

- Outstanding Segments are used instead of dummy segments. This replacement helps receiver to recover extra segment losses.
- Congestion Window Increment Function is added during the WEP. The sender's window keeps on growing during the WEP so that TCP-Swift is more aggressive to increase its transmission rate after an event of segment loss.

```
Speedy Recovery( )
    aosn = cwnd;
    wdsn = cwnd /2;
    apsn = 0;
    pipe = cwnd – ndupacks + 1;
    END=0;

    while (END=0)
        if (ACK_ARRIVE)
            if (Duplicate ACK)
                    pipe = pipe –1;
                    update scoreboard;
                    cwnd = cwnd + 1;
                    if (ACK_for_outstanding_segment)
                      if (wdsn > 0)
                        wdsn = wdsn – 1;
                      end;
                    end;
                end;
            else if (Partial ACK)
                cwnd = cwnd + 1;
                pipe = pipe – amountacked;
                update HighACK;
                update scoreboard;
            else if (ACK< swift_wep_recv)
                cwnd = cwnd + 1;
                //congestion window increment;
            end;

            if (Recover ACK)
                update HighAck;
                clear scoreboard;
                swift_wep_recv = tSeq;
                End =1;
            end;
            adps = cwnd – pipe;
            nss = min(maxburst, adsn);

            if (nss > 0)
                send nss missing packets and/or new
                  packets;
                pipe = pipe + nss;
            else if (adps>0)
                 send outstanding segment;
                 send outstanding segment;
                 aosn = aosn – 2;
            end;
        end;
end;
```

Fig. 3. Speedy recovery algorithm.

V. Performance Evaluation

We evaluate the performance of TCP-Swift by simulations using ns version 2.1b7a [10]. For comparison, TCP NewReno, TCP NewReno with SACK option [3], TCP-Peach, and TCP-Peachplus [9] are also implemented. We assume that a large file is transmitted between each sender-receiver pair using FTP protocol. Other related simulation parameters are summarized in Table 2.

Variable	Definition
Aosn	number of outstanding data segment can be sent during Speedy Recovery
Wdsn	preventing sender to increase its congestion window size for $cwnd_0/2$ ACKs for outstanding segments
Adps	number of segments allowed to be sent
Nss	number of segments have been sent
Pipe	number outstanding of segments in the network during speedy recovery
HighACK	sequence number of the highest cumulative ACK received
Partial ACK	an ACK that increases the HighACK value but not acknowledging all sent packets
*T_Seqno_*	sequence number of the current segment sent
Ndupacks	number of duplicated ACKs to trigger speedy recovery (a value of 3 is used).
Amountacked	number of segments have been acknowledged
swift_wep_recv	sequence number of the last segment sent when exits from rapid recovery.

Table 1 Variable definitions in speedy recovery algorithm of TCP-Swift.

Parameters	Symbols	Values
Number of senders/Receivers	N	20
Buffer size at ground stations A & B	K	50 segments
Receiver's window size	rwnd	64 segments
Satellite link capacity	C	10Mbps
Satellite link packet loss rate	P_{loss}	10^{-2}~10^{-5}
Round trip time of satellite link	RTT	550ms
Initial TCP Slow Start threshold	ssthresh	64 pkts
TCP segment size	Packet_size	1000 bytes
ACK size	ACK_size	40 bytes

Table 2 Simulation Parameters

A. Goodput Performance

Our first set of simulations is based on the network topology shown in Fig. 4. The system consists of N sender-receiver pairs. The senders and receivers are connected through satellite channel by ground stations A and B. In this simulation we consider a GEO satellite system where the round trip time of the satellite link is 550ms, and a bandwidth of 10 Mbps is assumed. All terrestrial links are characterized by 1 ms propagation delay and 10 Mbps bandwidth. The router buffer sizes at ground stations A and B are set to 50 and managed by droptail queuing mechanism.

Fig. 5 shows the goodput performance of TCP-Swift against the packet error rate. We can see that TCP-Swift gives the highest goodput performance. This accredits to the speedy start and seedy recovery algorithms adopted by TCP-Swift. We can also see that TCP-

NewReno with SACK option outperforms the general TCP-NewReno. This shows that SACK option can overcome the problem of multiple segment losses within a transmission window.

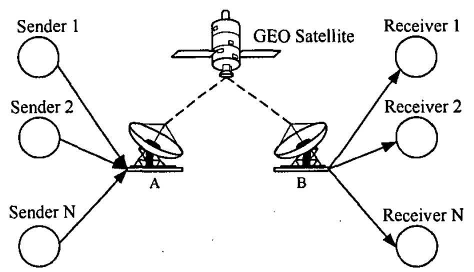

Fig. 4. Simulated network topology 1.

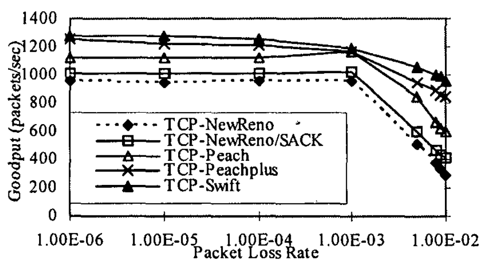

Fig. 5. Goodput performance comparison.

B. Fairness Performance

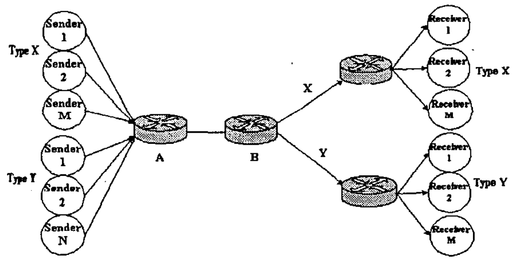

Fig. 6. Simulated network topology 2.

We study the case of heterogeneous scenario by considering the network topology shown in Fig. 6. There are M connections of type X and N connection of type Y. All connections pass through the same bottleneck link that connects routers A and B. The bottleneck link capacity is 10 Mbps. We assume type X connections pass through a GEO satellite link with round trip time $RTT_X = 550$ ms, and packet loss probability P_{LossX} varied from 10^{-6} to 10^{-2}. On the other hand, type Y connections pass through a terrestrial link, where no losses occur due to link error, i.e. $P_{LossY} = 0$ and the round trip time RTT_Y for the terrestrial link is set to 200 ms [4]. In our simulations, the number of connections X and connections Y are set to be the same ($M = N = 5$), and the bottleneck router buffer size is set to 50 and *rwnd = 64* segments.

The fairness performance is measured by the ratio between the average goodput (r_X) of type X's connections and the average goodput (r_Y) of type Y's connections. So $r_X / r_Y = 1$ gives the ideal fairness performance. In Fig. 7, we show that the fairness performance in the heterogeneous scenario. The legend "NewReno & Swift" means that connections of type X use TCP-Swift passing through satellite link and connections of type Y use TCP-NewReno passing through terrestrial link.

From Fig. 7, we can see that "NewReno & Swift" gives the best fairness performance as compared to BewReno & Peach" and "NewReno & NewReno". This is due to TCP-Swift's aggressiveness in maintaining the high transmission rate. So TCP-Swift can acquire enough network resource when competing with TCP-NewReno.

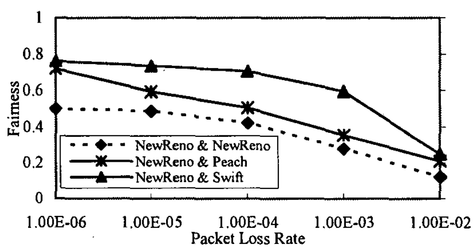

Fig. 7. Fairness performance comparison.

VI. Conclusions

In this paper, a new transport layer protocol called TCP-Swift was proposed for enhancing the TCP performance over satellite IP networks. TCP-Swift replaces the conventional TCP slow start and fast recovery algorithms by speedy start and speedy recovery. With speedy start, a TCP-Swift sender opens up its congestion window in only two round trip times. This significantly shortens the time needed in probing the network for equilibrium state. With speedy recovery, we can infer if a packet loss is due to wireless transmission errors by observing the ACK stream arrived at the sender. The sender's congestion window can be re-opened up more aggressively if the loss is due to transmission error.

Acknowledgements

This work was supported by Competitive Earmarked Research Grant HKU 1180/00E.

References:

[1] C. Metz, "TCP over satellite... The final frontier," IEEE Internet Computing, pp. 76–80, Jan./Feb. 1999.

[2] M. Allman et al., “Ongoing TCP research related to satellites,” RFC 2760, Feb. 2000.

[3] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, “TCP Selective Acknowledgment Options,” RFC 2018, Oct. 1996.

[4] I. F. Akyildiz, G. Morabito, S. Palazzo, “TCP Peach: A New Congestion Control Scheme for Satellite IP Networks,” IEEE/ACM Transactions on Networking, 307-321. Jun. 2001.

[5] V. N. Padmanabhan and R. H. Katz, “TCP Fast Start: A Technique for Speeding Up Web Transfers,” Proc. IEEE Globecom’98 Internet Mini-Conference, Sydney, Australia, November 1998.

[6] C. Partridge and T. J. Shepard, “TCP/IP performance over satellite links,” IEEE Network Mag., pp. 44–49, Sept./Oct. 1997.

[7] T. R. Henderson and R. H. Katz, “Transport protocols for Internet-compatible satellite networks,” IEEE J. Select. Areas Commun., vol. 17, pp. 326–344, Feb. 1999.

[8] M. Allman, "On the generation and use of TCP acknowledgments," ACM Computer Commun. Review, vol. 28, Oct. 1998.

[9] I.F. Akyildiz, X. Zhang and J. Fang, “TCP-Peach+: Enhancement of TCP-Peach for Satellite IP Networks,” IEEE Communications Letters, vol. 6 Issue: 7, pp. 303 -305, Jul. 2002

[10] S. McCanne and S. Floyd. Ns-LBNL Network Simulator. URL: http://www-nrg.ee.lbl.gov/ns/