Enhancing DCCP Congestion Control Mechanism for Long Delay Link

*Shahrudin Awang Nor, Suhaidi Hassan, Osman Ghazali, and Mohd. Hasbullah Omar
InterNetWorks Research Laboratory, School of Computing, UUM College of Arts and Sciences,
Universiti Utara Malaysia, 06010 UUM Sintok, Kedah Darul Aman, Malaysia.
{shah, suhaidi, osman, mhomar}@uum.edu.my

***Abstract*—Most of the multimedia applications use the User Datagram Protocol (UDP) as a transport layer protocol because it is suitable for the delivery of multimedia data over the Internet. However, the use of UDP could endanger the stability of the network because there is no congestion control applied. To a certain extent, the network can collapse if too many applications deliberately use this protocol. Subsequently, instead of using the UDP, the applications have choices to use the Datagram Congestion Control Protocol (DCCP), which has a built-in congestion control that can provide a better network. Nevertheless, the congestion control mechanism in the Congestion Control Identifier (CCID)-2 TCP-like can cause problems when delivering multimedia data over a long delay link. To alleviate the problems, such as longer time taken for achieving maximum throughput, and throughput fluctuation during a congestion avoidance phase, two approaches have been used, i.e. setting of an appropriate slow-start threshold value and adjusting congestion window during a congestion avoidance phase. "TCP-like Threshold Window (TCP-like TW)" has been developed and modeled in the Network Simulator 2 (*ns-2*). For a long delay link, the TCP-like TW congestion control mechanism is able to minimize the time taken to achieve the maximum throughput. It also can smooth the fluctuation of throughput after achieving the maximum throughput.**

***Keywords*—Datagram Congestion Control Protocol; TCP-like; congestion control**

I. Introduction

Datagram Congestion Control Protocol [1] is a congestion-controlled transport protocol for the delivery of multimedia data in the Internet. Basically, DCCP is a transport protocol which is an unreliable protocol like User Datagram Protocol (UDP), but it provides congestion control like Transmission Control Protocol (TCP). In this paper, a new congestion control mechanism is introduced, namely TCP-like Threshold Window (TCP-like TW). It can alleviate the congestion problem in DCCP and achieve maximum throughput faster than the traditional DCCP mechanisms over long delay link networks. In addition, it also can improve the throughput, jitter, and with acceptable packet loss.

II. Datagram Congestion Control Protocol

Datagram Congestion Control Protocol (DCCP) [1] is a new transport protocol. In short, DCCP is an unreliable transport protocol like UDP, but it provides congestion control mechanism like TCP. The historical record and the motivation behind the DCCP is described in Request For Comment (RFC) 4336 [2]. DCCP can be easily extended to support multiple congestion control mechanisms. Currently, DCCP has three congestion control mechanisms, TCP-like, TCP-Friendly Rate Control (TFRC) and TCP-Friendly Rate Control for Small Packet (TFRC-SP), specified. The introduction of DCCP has taken the TCP's friendliness issue in mind. DCCP is designed so that it can deliver multimedia data like UDP but with congestion control so that it behaves fairly, or friendly to other transport protocols such as TCP when they coexist in the network. In contrast to UDP, DCCP is an unreliable with congestion control mechanism transport protocol that can share the bandwidth fairly with other congestion controlled protocol such as TCP. In case of congestion in the network, the increase in the occupation of sender's queue introduces higher end-to-end delay, and subsequently packet discarding will occur in sender's queue [3].

*Corresponding Author

III. Related Work

The appearance of various real-time applications on the Internet has led to new types of real-time traffic. Nowadays, the most common real-time applications are bursty, such as video conferencing, IP telephony, and audio and video streaming. For these applications, the current network often assumes traffic with smooth changes in required bandwidth. However, more bursty real-time traffic is becoming increasingly common and must be accommodated. Applications that create such traffic include online games, video conferencing, and streaming with compressed encoding. Real-time traffic not only creates burstiness on the Internet, but also requires short delay transmission. Supporting this sort of traffic on the Internet is increasingly important.

Nowadays, TCP is still the dominant protocol on the Internet [4], [5], [6]. In order to successfully deploy DCCP widely on the Internet, it is vital to ensure that DCCP is fully compatible with the existing TCP flows, as showed by Nor et al. regarding the friendliness of DCCP towards TCP flows in the network [7]. Furthermore, DCCP with CCID-2 utilizing a TCP congestion control and it works similar to the congestion control of TCP. So it is important to study the multimedia traffic using DCCP with CCID-2 because DCCP is foreseen as a replacement transport protocol for TCP in the future.

There are several works done by researchers regarding multimedia traffic for DCCP. Quoc Truong Tong et al. [8] evaluated and proposed an algorithm to improve the TCP fairness of DCCP flow control for bursty real-time applications. They showed that their proposed algorithm improves the fairness between TCP and DCCP flows without negative effect on real-time characteristics. The performance evaluations of DCCP for bursty multimedia traffic in real-time applications are done by S. Takeuchi et al. [9]. Their work investigated the DCCP performance for various traffic flows, focusing on how DCCP flows affect TCP flows and vice versa. Through those simulations, they examined an unfair bandwidth distribution problem caused by the incompatibility of DCCP with the fast recovery algorithm of TCP.

IV. Initial Slow-start Threshold in Slow-start Phase

The normal network scenario, i.e. the network with short delay link, the size of initial slow-start threshold state variable (*ssthresh*) size is set to 20 packets to work well. The problem arises when there is a long delay link in the network connection. When there is a long delay link to deliver data, there will be a longer time for the the data delivery to achieve the maximum bandwidth available. This is due to the reason that long delay link has higher Round Trip Time (RTT), so that it requires more time to deliver the data to the receiver, as well as the Acknowledgments (ACKs) which is used by the sender to increase the congestion window for the next sending data.

The congestion window concept is important in TCP, as well as DCCP. With higher congestion window, the higher the volume of packets can be sent at the certain time simultaneously. Like buffer in the network, if congestion window grows faster, then the throughput will reach the maximum level faster. During slow-start phase, the congestion window grows exponentially, e.g. if starting from the value of 1, then at the next RTT it will become 2, then 4, 8, 16, 32, and so on. The slow-start concept in TCP is defined in RFC 2581 [10], RFC 2001 [11] and RFC 1122 [12]. The limited slow-start for TCP with large congestion window is also defined in RFC 3742 [13].

Since the congestion window concept in TCP is also applicable DCCP, the faster the congestion window grows, then the faster the maximum bandwidth will be achieved. It leads to better utilization of bandwidth for the data delivery by DCCP. Nevertheless, similar to TCP, the initial slow-start threshold in DCCP plays an important role in terms of time consumed in order to get the maximum throughput. The value of initial slow-start threshold size limits the current congestion window grow during the slow-start phase at the beginning of the connection where the congestion window grows exponentially.

The idea in this research regarding the use of higher initial slow-start value at the beginning of slow-start phase is to make the congestion window size (*cwsize*) to increase to the value of initial slow-start value faster, or in other words, to minimize the time taken for congestion window (*cwnd*) state variable to exceed its maximum value in slow-start phase. In slow-start phase, the *cwsize* will increase exponentially. Once the value of *cwnd* exceeds the initial slow-start value, it then enters the second phase, the congestion avoidance phase where the *cwsize* increases linearly. During this phase, the congestion window increases slower than during the slow-start phase. In brief, the purpose of increasing the initial *ssthresh* of the congestion window is to make the expansion of *cwnd* to be faster, so that the maximum bandwidth will be achieved faster.

Since DCCP is a new transport protocol, most of the research works of DCCP are related to its performance [14], [15]. Nevertheless, since TCP-like congestion control mechanism follows the standard TCP, the improvement works on the TCP regarding slow-start threshold [16], [17] and standard TCP-like [18], [19] are related to DCCP as well.

V. Congestion Window in Congestion Avoidance Phase

TCP-like utilizes congestion window to cope with the congestion like the one in TCP. It is like a buffer in the network where it can expand and shrink in size depending on the condition of the network. For TCP, *cwnd* is a state variable that limits the amount of data a TCP can send. At any given time, a TCP might not send data with a sequence number higher than the sum of the highest acknowledged sequence number and the minimum of *cwnd* and receiver window (*rwnd*). In more specific definition, *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 *rwnd* is a receiver-side limit on the amount of outstanding data.

Similar to TCP, *cwsize* in TCP-like will be reduced after congestion events are detected. There are two types of congestion events, i.e. through timeout and through the receiving of three duplicate ACKs by the sender. If timeout is detected, the process will start all over again, that is, *cwnd* will be initialized to one and the slow-start phase will initiate with a new *ssthresh* value which is half of the current *cwnd* value. During slow-start phase, the value of *cwnd* increases exponentially until the *cwnd* reaches the new *ssthresh* size. Then the congestion avoidance phase will take place after the *cwnd* reaches the new *ssthresh* size and *cwnd* will increase by one unit linearly until a congestion event is detected through the receiving of three duplicate ACKs. After this, the current *cwnd* will be halved, and the process of increasing *cwnd* by one continues, and so on and so forth. The proses of halving the *cwnd* size is kept continuing until timeout is detected. Upon the detection of timeout, the congestion avoidance phase will be ended and the *cwnd* size is initialized to the very beginning state and the slow-start phase is revoked gain, with the new *ssthresh* is set to half of the current *cwnd*.

One formula commonly used to update *cwnd* during congestion avoidance is as in Equation 1 taken from RFC 2581 [10]:

$$cwnd \quad += \quad SMSS * SMSS/cwnd \tag{1}$$

where SMSS is the sender maximum segment size. This adjustment is executed on every incoming non-duplicated ACK. It provides an acceptable approximation to the underlying principle of increasing *cwnd* by one full-sized segment per RTT. If the formula yields zero, the result is rounded up to one byte. Another acceptable way to increase *cwnd* during congestion avoidance is to count the number of bytes that have been ACKed for new data. When the number of bytes ACKed reaches *cwnd*, the *cwnd* is incremented by up to SMSS bytes. When a TCP sender detects segment loss using the retransmission timer, the value of *ssthresh* must be set according to Equation 2 [10]:

$$ssthresh \quad = \quad max(FlightSize/2, 2*SMSS) \qquad (2)$$

where FlightSize is the amount of outstanding data in the network.

VI. Simulation Environment

All the experiments are done as a basis for the design of the new congestion control mechanism for DCCP. It is based on the adjustment of two parameters in standard TCP-like, namely slow-start threshold and congestion window. The experiments determine the performance of the TCP-like with the modification of the two parameters and they are done individually in order to identify specifically for each of them.

These experiments apply to long delay link networks, where the threshold size adjustment will be applied at the slow-start phase, and congestion window during congestion avoidance phase. The simulations are set under controlled-environment where all the bandwidth, nodes and routers are pre-set with only TCP and DCCP flows competing with each other.

In the simulation experiments, the senders and receivers of TCP-like and TCP are used over long delay link network. The experiments are carried out by means of simulation with the simulation topology as shown in Fig. 1. The network simulation topology uses classic dumbbell topology. Dumbbell topology is a very common topology that has been used in many TCP network simulations.

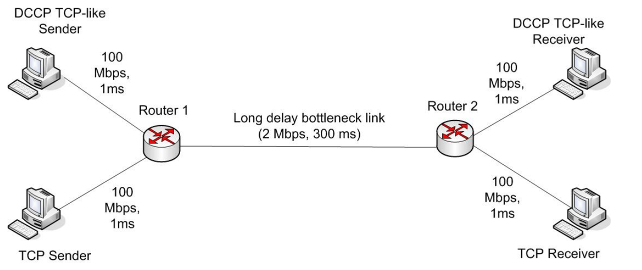

Fig. 1. Simulation topology

For all the experiments, the simulations consist of TCP and DCCP senders. At the receiver's side, there are TCP and TCP-like receivers. All the senders and receivers are connected to the routers through 10 Mbps links with 1 ms propagation delay.

In the simulation environment, DCCP has been simulated as a competing protocol to TCP, so that it can be seen how the other protocol such as DCCP behaves when they coexist with TCP. The utilization of bandwidth by these two competing protocols is set into a scenario so that a DCCP sender will fully utilize the 2 Mbps bandwidth with the sending rate of 2 Mbps Constant Bit Rate (CBR) traffic. The CBR packet size used is 500 bytes. In this case, TCP sender sends the file transfer data using File Transfer Protocol (FTP) application so that the friendliness of DCCP protocol towards TCP can be monitored. Unlike DCCP, where the transmission bit rate can be set by the application like CBR, the maximum bit rate occupied by FTP application on TCP will be calculated by the transport protocol itself based on the link bandwidth provided, packet size, propagation delay, etc. From the simulation results, the analysis will be done to see how *cwsize* drop affects the performance of DCCP.

The network topology used in the simulation includes two interconnected routers, Router 1 and Router 2 with queue size of 20 packets. For the router to router connection, a long delay bottleneck link is set to have a bandwidth of 2 Mbps with 300 ms propagation delay. This long delay bottleneck link can be used as an emulation of satellite or wireless links with a fixed forward link delay of 300 ms and fixed return link delay of 300 ms. This assumption is reasonable based on Henderson and Katz [20] for the satellite link. There is also research done by other researchers that used this assumption for a long delay link [21]. In addition, the bottleneck link is considered to have enough bandwidth allocation for the data transfer to flow from the sender to the receiver. For simplicity, instead of using other types of queue management such as Random Early Detection (RED), the type of queue management used in this link is Drop Tail, which implements First-In First-Out (FIFO). The network simulator *ns-2* [22] with DCCP module [23] installed is chosen for the simulation.

TCP NewReno is used for all of the simulations because it is one of the most popular TCP variants used in the Internet nowadays [24], [25], [26], and for DCCP, the congestion control used is TCP-like. The throughput is measured between Router 1 and Router 2 where the TCP-like and TCP flows compete with each other on the long delay link. The TCP connection is monitored while it coexists with DCCP connection.

VII. Experiments and Results

There are two main experiments done to improve the performance of TCP-like. Both experiments, i.e. Experiments VII-A and VII-B are related to initial *ssthresh* size and congestion window adjustment, respectively, with the main concentration on the throughput of the TCP-like flow. Susequently, Experiment VII-C gives the performance of the newly designed congestion control, TCP-like TW.

A. Experiment 1 - Initial Slow-start Threshold size

The common initial *ssthresh* size for DCCP is 20 packets and the optimal size of initial *ssthresh* for the best performance of TCP-like is investigated. The simulation time for Experiment 5-1 is set to 200 seconds because this period is long enough to get the whole picture of the performance of TCP-like which is affected by the initial *ssthresh* size. Six different initial slow-start values are used, i.e. 20, 50, 100, 200, 300 and 400 packets. In the case, the slow-start phase that is investigated happens at the beginning of the connection. In all the simulation experiments, the FTP application using TCP is started first, i.e. at time 0.5 seconds, whereas the CBR

TABLE I. AVERAGE THROUGHPUT, PACKET LOSS, AVERAGE DELAY AND AVERAGE JITTER FOR DIFFERENT CWND DROP

		***cwnd* Size Drop for TCP-like**		
		50%	25%	5%
Average Throughput (kbps)	TCP-like	1396.65	1607.06	1737.36
	TCP	282.46	281.33	272.74
Packet Loss (%)	TCP-like	0.002400	0.003994	0.016820
	TCP	0	0	0
Average Jitter (ms)	TCP-like	1.285047	1.277461	1.273508
	TCP	0.000086	0.000146	0.000226
Average Delay (ms)	TCP-like	309.9404	313.2521	330.1065
	TCP	305.7689	307.7152	316.9234

application for TCP-like is started at time 10 seconds. It is assumed that 10 seconds is enough to allow the TCP data flow to utilize the bandwidth without any contention with another flow, so that the effect on throughput of having other flows joining the bottleneck link after that can be monitored.

The purpose of this experiment is to find the optimum value for initial *ssthresh* size in DCCP with TCP-like congestion control mechanism. The graph in Fig. 2 shows several different times taken to exceed the maximum throughput with different initial *ssthressh* size, i.e. 20, 50, 100, 200, 300 and 400 packets. It is noted that the results for the 300 and 400 packets are the same as for the 200 packets one. For normal initial *ssthresh* size, where the size is 20, it can be seen that it exceeds the maximum bandwidth at about the simulation time of 160 ms, whereas for the initial *ssthresh* size of 200 packets, the maximum throughput is gained at the time of about 100 ms. This is because the increase of throughput is exponential during slow-start phase, i.e. until its current *cwnd* exceeds the initial *ssthresh* size. After that, the congestion avoidance phase will be entered where the throughput is increasing linearly. An experiment

It shows that with the initial *ssthresh* of high value, in this case of 200 packets, the maximum throughput can be obtained faster. The values of initial *ssthresh* of higher than 200 packets do not give any better result. The optimum value of 200 packets for the initial *ssthresh* size has given the best performance in this research in terms of throughput.

It is shown that TCP-like with the adjustment of initial *ssthresh* size can improve the performance of the TCP-like in terms of faster time required to obtain the matured throughput during slow-start phase over long delay link network. The slow-start phase can be in action during the initial connection establishment or reconnection after idle time.

From the experiment result for packet loss as given in TABLE I, the percentage of packet loss or Packet Loss Ratio (PLR) is 0.0277% for initial *ssthresh* size of 200 packets, which is within the specification by Telecommunication Standardization Sector of the International Telecommunication Union (ITU-T). ITU-T recommendation G.1010 [27] states that PLR must be less than 1% for video data.

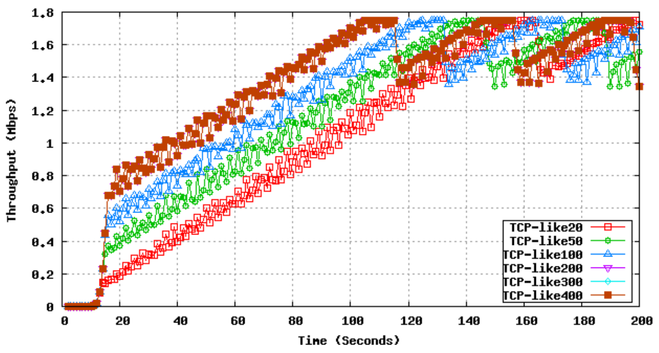

Fig. 2. Throughput for TCP-like with Different Initial Slow-start Size

B. Experiment 2 - Congestion Window Drop

In all simulations, all data traffics have to go through a bottleneck link with 2 Mbps bandwidth and 300 ms propagation delay for long delay link. This bottleneck link is the link that connects the two routers in the simulation topology. So it is the link that limits the sending rate of the application data between these two routers. The reduction of *cwsize* drop of 25% and 5% are done for TCP-like congestion control mechanisms. The results presented are given in the TABLE I which shows the average throughput, packet loss, delay and jitter for TCP-like and TCP flows. As for the throughput, Fig. 3 gives the throughput graph of the TCP and TCP-like where the congestion window adjustment is applied.

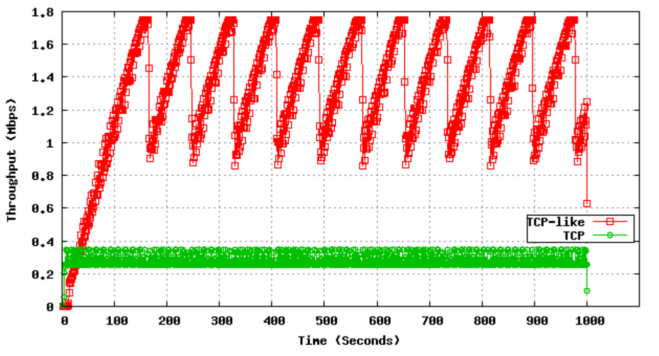

Fig. 3. Throughput for TCP and TCP-like

The result is as in the Fig. 3, where the standard TCP-like mechanism is used with the drop of 50% when congestion events are detected.

1) Congestion Window Drop of 50%: In normal case when the congestion is detected during congestion avoidance phase, the current *cwsize* is halved, or in other words, the current *cwsize* is divided by two. This is a standard in the operation of TCP, as well as TCP-like. Same as depicted in Fig. 3, it shows the throughput graph of TCP-like over time. The halving or drop of 50% from the current congestion window is equivalent to the current value of the congestion window divided by two.

Similar to TCP, *cwsize* in TCP-like is halved whenever there is a congestion event detected during congestion avoidance phase through the receiving of three duplicate acknowledgments by the sender. Fig. 3 shows that the throughput of TCP-like is fluctuated like the zigzag pattern when it enters congestion avoidance phase at the time around 170 seconds until the end of the simulation time.

2) Congestion Window Drop of 25%: Fig. 4 shows the throughput graph of TCP-like over time with *cwnd* drop of 25%. This is equivalent to the value of current *cwnd* multiplied by 0.75, or in other words, the new congestion window is 25% below the current one.

As in Fig. 4, the throughput is improved compared to the previous experiment with *cwnd* drop of 50%. There are better throughput and jitter for TCP-like flow with acceptable packet loss, with packet loss ratio (PLR) is less than 1% as ITU-T Recommendation G.1010 [27] sets the PLR for video data to be less than 1% as a standard. ITU-T Recommendations is a recommendation by International Telecommunication Union (ITU) for defining elements in information and communication technologies (ICTs) infrastructure. G.1010 is a recommendation by ITU for End-user multimedia QoS categories, under G series which is for Transmission systems and media, digital systems and networks.

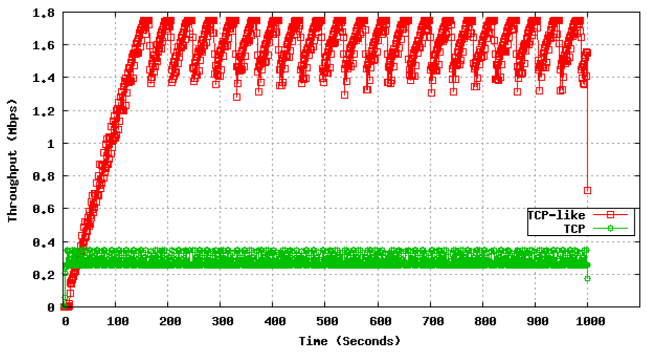

Fig. 4. Throughput of TCP-like with *cwnd* Drop of 25%

3) Congestion Window Drop of 5%: Fig. 5 shows the throughput graph of TCP-like over time with *cwnd* drop of 5%. This is equivalent to the value of current congestion window multiply by 0.95. It means that the new *cwnd* is just 5% below the current one. Fig. 5 depicts the result that shows how the throughput and jitter are improved a lot for TCP-like flow when the drop of TCP-like's *cwsize* is reduced by 5%. The throughput and jitter for TCP-like flow are better compared to the previous experiment with *cwnd* drop of 25% with acceptable packet loss. PLR is also less than 1% and it is in accordance to ITU-T Recommendation G.1010 [27].

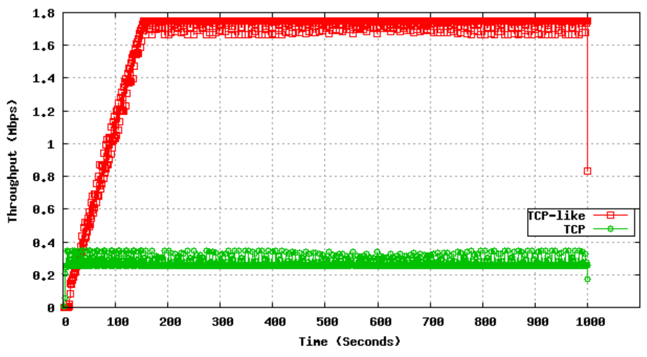

Fig. 5. Throughput of TCP-like with *cwnd* Drop of 5%

C. Experiment 3 - Performance of TCP-like TW

The purpose of the experiment is to study the performance of TCP-like TW with the existing TCP-like over long delay link. A single DCCP sender and a single TCP sender are used, together with their receivers. The common initial slow-start threshold state variable (*ssthresh*) size is 20 packets. Then, the optimized size of initial *ssthresh* for the best performance of TCP-like TW can be found.

Fig. 5 and Fig. 3 show the friendliness of DCCP TCP-like TW and DCCP TCP-like, respectively, with TCP flows over long delay link. They can coexist together in harmony over long delay link network and the throughput is shared fairly between the two flows.

From the simulation result for the experiment of TCP and DCCP TCP-like TW, the throughput for TCP flow is about 300 kbps while it is 1.7 Mbps for TCP-like TW flow. Likewise, for the experiment of TCP and DCCP TCP-like, the throughput for TCP flow is similar, i.e. 300 kbps while the throughput for DCCP TCP-like is 1.3 Mbps. It is obvious that TCP-like TW outperforms the TCP-like.

The comparison of TCP-like TW and TCP-like is depicted as in Fig. 6. The overall view shows that TCP-like TW algorithm can achieve the maximum throughput value faster and with more stable state for the rest of the connection.

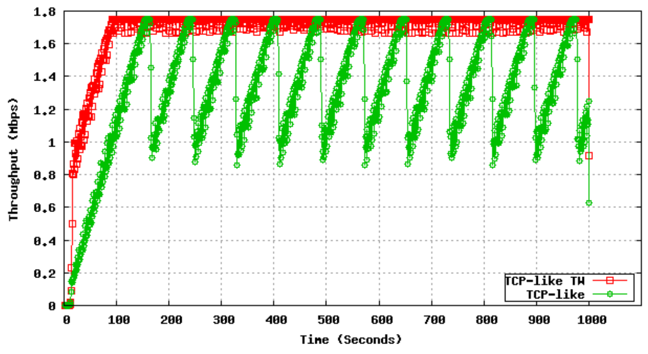

Fig. 6. Comparison of Throughput for TCP-like TW and TCP-like over Long Delay Link

The beginning part of the throughput of TCP-like TW and TCP-like as displayed in Fig. 6 shows the difference between these two congestion control schemes and how the TCP-like TW outperforms the TCP-like during the beginning and the entire time of the connection.

The throughput for the TCP-like is having a zigzag pattern which is not intended for the entire connection. As in the Fig. 6, the TCP-like throughput takes longer time to exceed the maximum value of the throughput because it has to enter the congestion avoidance phase after the congestion window value exceeds the initial *ssthresh* in the first phase, i.e. slow-start phase, which is small.

At the second phase, the drop of the congestion window state variable (*cwnd*) of TCP-like which is 50% consumes time for the current *cwnd* to increase to the maximum value, i.e. before a congestion event is detected. For TCP-like TW, the drop is reduced to only 5% of the current *cwnd*, and it is proven that the performance of DCCP is enhanced through smoother and more stable throughput.

In more detail, as shown by TABLE II, the average throughput and average jitter of TCP-like TW outperform the TCP-like whereas TCP-like TW gives a bit higher of average delay and packet loss.

TABLE II. Comparison of TCP-like TW, TCP-like and TCP in terms of Average Throughput, Packet Loss, Average Delay and Average Jitter for Long Delay Link

Average Throughput (kbps)	TCP-like TW	1697.57
	TCP	273.12
	TCP-like	1301.43
	TCP	282.11
Packet Loss (%)	TCP-like TW	0.021031
	TCP	0
	TCP-like	0.002399
	TCP	0
Average Delay (ms)	TCP-like TW	328.493
	TCP	329.155
	TCP-like	309.645
	TCP	309.076
Average Jitter (ms)	TCP-like TW	0.642
	TCP	55.016
	TCP-like	1.841
	TCP	52.827

VIII. Conclusion

A new congestion control technique introduced for DCCP, called TCP-like TW, is a combination of two improved techniques, i.e. Threshold and Window. TCP-like TW is proven through simulation to give better performance on long delay link. With this new TCP-like TW, the time taken to achieve the maximum bandwidth utilized can be optimized during slow-start and congestion avoidance phases, and the throughput will be more stable with less fluctuation when it is at its stable state at the maximum bandwidth utilized. For multimedia data, TCP-like TW mechanism gives better throughput with a bit higher in packet loss and it meets the specification by ITU-T for video data delivery. As a conclusion, TCP-like TW can perform better performance over long delay link where it can reduce the time taken for exceeding the maximum throughput and smoothen the throughput at the stable state for the entire of the connection with better delay.

References

[1] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," Internet Engineering Task Force, RFC 4340, Mar. 2006.

[2] S. Floyd, M. Handley, and E. Kohler, "Problem Statement for the Datagram Congestion Control Protocol (DCCP)," Internet Engineering Task Force, RFC 4336, Mar. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4336.txt

[3] R. N. V. M. S. Nunes, "Selective Frame Discard for Video Streaming over IP Networks," in *Proceedings of the 7th Conference on Computer Networks (CRC2004)*, 2004.

[4] S. Floyd, "Congestion Control Principles," Internet Engineering Task Force, RFC 2914, Sep. 2000.

[5] J. Yu and S. Choi, "Modeling and analysis of TCP dynamics over IEEE 802.11 WLAN," in *Fourth Annual Conference on Wireless on Demand Network Systems and Services 2007 (WONS '07)*, Jan. 2007, pp. 154 –161.

[6] M. Arlitt and C. Williamson, "An analysis of TCP reset behaviour on the internet," *ACM SIGCOMM Computer Communication Review*, vol. 35, pp. 37–44, Jan. 2005. [Online]. Available: http://doi.acm.org/10.1145/1052812.1052823

[7] S. A. Nor, S. Hassan, O. Ghazali, and A. S. M. Arif, "Friendliness of DCCP towards TCP over large delay link networks," in *the Proceedings of International Conference on Information and Network Technology 2010 (ICINT 2010)*, vol. 5, Shanghai, China, 22–24 June 2010, pp. 286–291.

[8] Q. T. Tong, H. Koga, K. Iida, and Y. Sakai, "TCP Fairness Improvement of DCCP Flow Control for Bursty Real-Time Applications," *First International Conference on Communications and Electronics 2006 (ICCE '06)*, pp. 66–71, Oct. 2006.

[9] S. Takeuchi, H. Koga, K. Iida, Y. Kadobayashi, and S. Yamaguchi, "Performance evaluations of DCCP for bursty traffic in real-time applications," *Proceedings of The 2005 Symposium on Applications and the Internet (SAINT '05)*, pp. 142–149, 31 Jan.-4 Feb. 2005.

[10] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," Internet Engineering Task Force, RFC 2581, Apr. 1999.

[11] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," Internet Engineering Task Force, RFC 2001, Jan. 1997.

[12] R. Braden, "Requirements for Internet Hosts – Communication Layers," Internet Engineering Task Force, RFC 1122, Oct. 1989.

[13] S. Floyd, "Limited Slow-Start for TCP with Large Congestion Windows," Internet Engineering Task Force, RFC 3742, Mar. 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3742.txt

[14] S. Bhatti, M. Bateman, and D. Miras, "A comparative performance evaluation of DCCP," in *International Symposium on Performance Evaluation of Computer and Telecommunication Systems 2008 (SPECTS 2008)*, 2008, pp. 433–440.

[15] I. S. Chowdhury, J. Lahiry, and S. F. Hasan, "Performance analysis of Datagram Congestion Control Protocol (DCCP)," in *12th International Conference on Computers and Information Technology 2009 (ICCIT '09)*, 2009, pp. 454–459.

[16] Z. Kaiyu, K. L. Yeung, and V. O. K. Li, "Throughput modeling of TCP with slow-start and fast recovery," in *IEEE Global Telecommunications Conference 2005 (GLOBECOM '05)*, vol. 1, 2005, p. 5.

[17] R.-S. Cheng, H.-T. Lin, W.-S. Hwang, and C.-K. Shieh, "Improving the ramping up behavior of TCP slow start," in *19th International Conference on Advanced Information Networking and Applications 2005 (AINA 2005)*, vol. 1, Mar. 2005, pp. 807 – 812 vol.1.

[18] A. Chaintreau, F. Baccelli, and C. Diot, "Impact of TCP-like congestion control on the throughput of multicast groups," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 500–512, 2002.

[19] Y. Zhao and L. Song, "Stability of TCP-Like Congestion Control Algorithm," in *2nd International Symposium on Computational Intelligence and Design 2009 (ISCID '09)*, vol. 1, 12-14 Dec. 2009, pp. 374–377.

[20] T. R. Henderson and R. H. Katz, "Transport Protocols for Internet-Compatible Satellite Networks," vol. 17, no. 2, 1999, pp. 326–344.

[21] A. Sathiaseelan and G. Fairhurst, "Use of Quickstart for Improving the Performance of TFRC-SP Over Satellite Networks," in *International Workshop on Satellite and Space Communications (IWSSC2006)*, Spain, 14–15 Sep. 2006, pp. 46–50.

[22] "The VINT Project. The Network Simulator - ns-2," http://www.isi.edu/nsnam/ns/, retrieved on 20 January 2011.

[23] N.-E. Mattsson, "A DCCP module for ns-2," Master's thesis, 2004.

[24] A. Chydzinski and A. Brachman, "Performance of AQM Routers in the Presence of New TCP Variants," in *Proceedings of Second Int Advances in Future Internet (AFIN) Conf*, 2010, pp. 88–93.

[25] S. Henna, "A Throughput Analysis of TCP Variants in Mobile Wireless Networks," in *Proceedings of Third International Confonference on Next Generation Mobile Applications, Services and Technologies (NGMAST '09)*, 2009, pp. 279–284.

[26] C. Grimm and H. Schwier, "Empirical Analysis of TCP Variants and Their Impact on Grid FTP Port Requirements," in *Proceedings of Third International Conference on Networking and Services (ICNS)*, 2007.

[27] "ITU-T Recommendation G.1010 End-user multimedia QoS catagories," International Telecommunication Union, Tech. Rep., Mar. 2003. [Online]. Available: http://www.itu.int/rec/T-REC-G.1010