DCM+: Robust Congestion Control Protocol for Mobile Networks

Rushdi A. Hamamreh
Al-Quds University
Jerusalem, Palestine
Email: rushdi@staff.alquds.edu

Derar Khader
Al-Quds University
Jerusalem, Palestine
Email: cedkhader@gmail.com

Abstract—**This paper aims at presenting a new robust congestion control protocol for mobile networks. It also can be used for mixed networks and mobile adhoc networks (MANETs). The proposed protocol is called D*ynamic Congestion Control Protocol for Mobile Networks* (DCM+). It makes use of the bandwidth estimation algorithm used in Westwood+ algorithm. We evaluate DCM+ on the basis of known metrics like *throughput, average delay, packet loss* and *Packet-Delivery-Ratio (PDR).* New metrics like *Normalized* A*dvancing Index (NAI)* and *Complete-Transmission-Time (CTT)* have been introduced for a comprehensive comparison with other congestion control variants like NewReno, Hybla, Ledbat and BIC. The simulations are done for a one-way single-hop-topology (*sender->router->receiver)*. The findings in this paper clearly show excellent properties of our proposed technique like robustness and stability. It avoids congestions, increases performance, minimizes the end-to-end delay and reduces the transmission time. DCM+ combines the advantages of the protocols NewReno and Westwood+. The simulation results show high improvements, which make this approach extremely adequate for different types of networks.**

Keywords-Congestion control; DCM+; wireless; ns3 simulator.

I. INTRODUCTION

Congestion control is a vital process for data networks, especially those that rely mainly on TCP (Transmission Control Protocol) traffic. It has a central role for achieving high performance and throughput through managing congestions. This results in preventing the global networks like the Internet from collapse [2][3]. Since 1986, many protocols have been proposed and implemented for controlling data transmission between hosts. TCP NewReno is one of the most prominent variants of the old days [4][9][13][25], which though has some drawbacks and limitations, especially in wireless, mobile and mixed networks [3][5][7][20]. Another limitation of TCP NewReno is its little support for mobility [3][7][9], which makes it unusable in MANETS. TCP NewReno has been implemented in the TCP protocol stack of different applications and operating systems. Recently, newer TCP variants like TCP Westwood+, BIC, CUBIC, HighSpeed, Scalable, Hybla and Ledbat are available in modern applications and operating systems like Linux [6][8][10][11]. TCP Ledbat, for example, is implemented under MS Windows Server 2019, and also in MS Windows 10 [12].

TCP DCM+ is a new end-to-end approach that we have proposed in [1]. It stands for dynamic congestion control for mobile systems. It uses the Bandwidth Estimation (*BWE*) algorithm of TCP Westwood+, and hence comes the (+) sign. DCM+ is designed to avoid the congestion events in wireless and mobile networks. It also improves the performance in wired and mixed networks.

Despite the appropriate design for managing congestions in old (wired) networks, the main weakness of TCP NewReno is that it cannot distinguish the reasons for packet losses [3][5]. Two main reasons are known for packet losses. The first reason is a “full buffer” of the intermediate router, which is known as “network congestion”. In this case, the data packet could be dropped intentionally from the router [13]-[18] like in Random-Early-Discard (RED). The aim of this strategy is to mitigate the large number “queue” of packets waiting for entrance into the router interface. The second reason is a signal error on the wireless channel, which is known as Link-Error (*LE*) [3][5][19][20][23]. In both cases, TCP NewReno drops its Congestion Window (*cwnd)* to the half, even if no real congestion exists and the packet was only dropped because of a bad wireless link [3][4][22]-[25]. This is the main reason for the bad performance of TCP NewReno in wireless and mobile networks.

This paper contains 5 sections. It is structured as follow: In section 2, works related to congestion control are mentioned. In section 3, we present our proposed technique. In section 4, the results and the simulations are shown. Section 5 is the conclusion and possible future work.

II. RELATED WORK

Many approaches dealing with modelling and identification of packet losses have been suggested [19][24], but this problem is still an active research topic. Fuzzy Logic (***FL***) and Machine Learning (***ML***) are some of the fields that have been used and tested to answer the question: “why is the packet lost?”. Fuzzy inference systems [26][27], ANFIS [28], ML classification [29][30], neural networks [31][32] and random forests are just some of the modern approaches and algorithms to distinguish between *true* and *false* congestion events in mixed and mobile networks. The correctly identified congestion events are known as TP or “*True Positives*”. In this case, Congestion-Avoidance (***CA)*** phase will be launched and new values for both Slow-Start Threshold (***ssth)*** and ***cwnd*** will be calculated. Otherwise, if the packet is dropped because of a link error, then the transmission continues without any change [27][29][30]. Hence, no or little false drops will occur, and thus, throughput will not suffer as in old TCP variants.

DCM+, on the other hand, is not causing any congestions during the transmission. It increases its *cwnd* size during the *CA* phase depending on the values of previous and current Round-Trip Times (***RTT***). Hence, DCM+ high performance is achieved because of using *RTT* as implicit feedback to predict the probability of a congestion, and thereafter to put the appropriate *cwnd* on the channel. This way, DCM+ reduces the probability of a congestion to an extremely low level. As a result, theoretically, the number of *cwnd* drops will be zero or very small. This results in high Packet-Delivery-Ratio (PDR), even when the packet-error-rates is too high. An example of the dynamics of TCP transmission using DCM+ is shown in Figure 1.

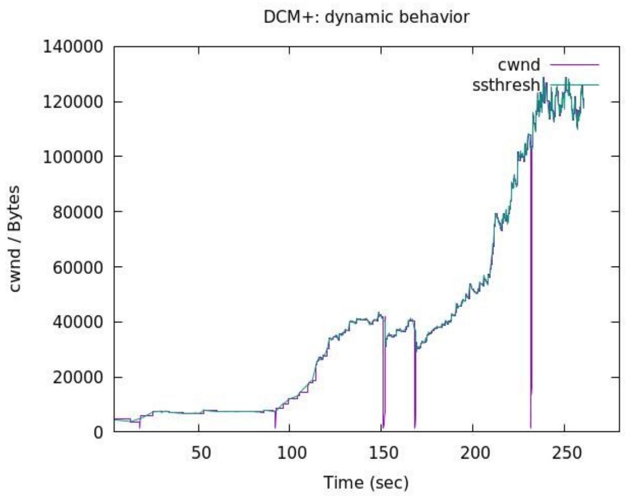

Figure 1. dynamic behavior of cwnd in DCM+

We see that ***cwnd*** is always tracking the actual state of slow-start-threshold (***ssthresh***). This causes a speedup in the transmission and hence, outperforms other TCP variants. Except at countable time points, which represent the lost packets, ***cwnd*** is tracking the state of ***ssthresh***. When a packet is lost because of a bad link conditions, the timeout counter signals this through a drop in the window size. This simulation is done using ns-3 with the following parameters:

- Bottleneck Bandwidth = 10 Mbps
- Access Bandwidth at the destination = 100 Mbps
- Packet-error-rate = 0.01
- Maximum Transmission Unit (MTU) = 1500 Bytes.

III. PROPOSED APPROACH

DCM+ is an End-To-End approach that uses the same algorithm explained in TCP Westwood+ [5][11][21][22] to find the accurate estimation of available bandwidth on the link. It describes a sender-side modification of ***CA*** phase of TCP Westwood+ protocol. Depending on the current discrete values of ***BWE***, DCM+ calculates the values for the next interval. The behavior of ***cwnd*** is observed to be dynamical. If a change (increase/decrease) of ***ssthresh*** has been observed within a specific time interval, then ***cwnd*** of DCM+ keeps using the same value of ***ssthresh*** until a newer state of ***ssthresh*** has been reached. After that, ***cwnd*** moves and remains at the new state for a new time interval. This way, ***cwnd*** will never (barely) exceed the available ***ssthresh***. Hence, congestion events will be extremely minimized. Figure 1 shows this behavior for packet-error-rate = 0.01, MTU =1500 bytes, bottleneck BW=10 Mbps and access-BW=100 Mbps.

Steady-state and stability for packet error rates lower than (0.05) can be observed from the simulations. Higher packet error rates, different MTU sizes and different sizes of TCP buffer can affect the dynamics of DCM+. Hence, the number of the packet drops is affected as shown in Figure 2 and Figure 3. The simulations are executed under ns-3.29 using the file 'tcp-variants-comparison.cc'. The used topology is a simple one-hop network. The topology is built of (TCP Source-> Router -> TCP Destination). The traffic is one-way TCP traffic only. No reverse traffic is used. The simplicity of this topology is vital to show the best performance that can be achieved by the different TCP variants. The used TCP variants are part of the simulator. NewReno, Hybla, Ledbat, BIC, Westwood/Westwood+ are implemented as C++ files. DCM+ has been implemented as a modified TCP Westwood+. The wireless channel is represented as a channel with high variable sporadic packet error rates.

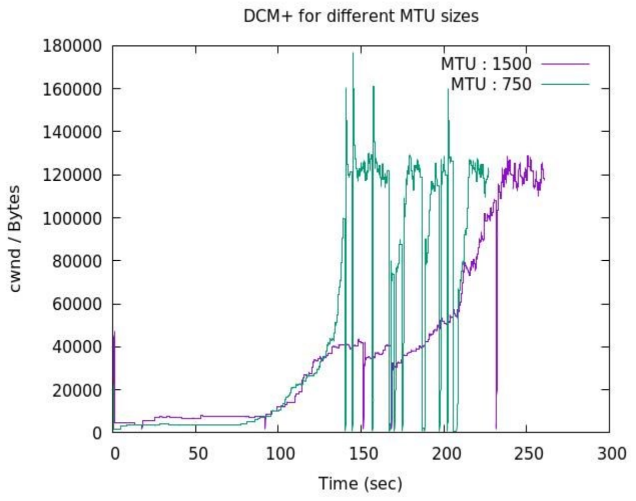

Figure 2. DCM+ behavior for different MTU

The design of DCM+ is similar to NewReno, which is detailed as an RFC [25]. DCM+ uses the same 4 phases like NewReno (***SS***, ***CA***, fast retransmission (***FR***) and fast recovery (***FV***)). In DCM+, the behavior in ***CA*** has been so modified to enforce the ***cwnd*** to track ***ssth*** in the next time interval. TCP timing parameters ***RTT*** and ***RTO*** have been used as feedback signals to control the values of ***ssth*** and ***cwnd*** in the next interval.

$$rateCA = RTT\_old / RTT\_new \qquad (1)$$

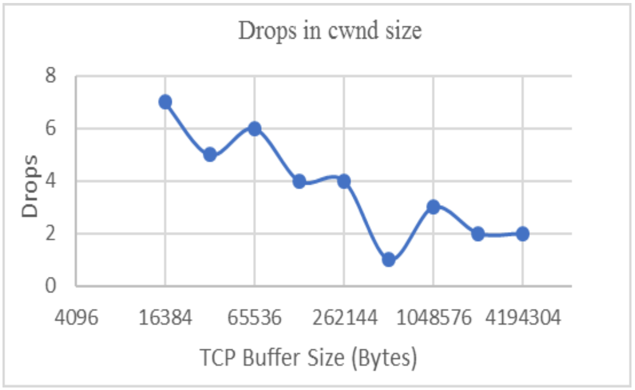

Figure 3. DCM+ drops vs. TCP buffer size

Figures 3 and 4 are shown for different TCP buffer sizes of the intermediate node. Figure 3 shows how many ***cwnd*** drops occur depending on the buffer size. According to [1], these drops occur only if a packet is lost because of a bad wireless link as no congestion events are allowed. We see that we get a minimum of drops when the buffer size is equal 512 KB. On the other hand, in Figure 4, we have the complete transmission time (***CTT***) as a function of TCP buffer size. Per definition, ***CTT*** is the difference between the arrival time of last ACK and first ACK segments:

$$CTT = last\_ACK\_time – first\_ACK\_time \qquad (2)$$

Figure 4 shows that for TCP buffer sizes equal or higher than 512KB, the TCP connection will have the shortest possible CTT.

DCM+ follows the following principle: it considers values of ***rateCA*** higher than **1** as ***advance*** or “Link Capacity Increasing”, and values lower than **1** as ***danger*** or “Link Capacity Decreasing”. Depending on the conditions stated in the algorithm of ***CA*** phase in [1], if ***cwnd*** is less than ***ssth***, then ***rateCA*** will be used to start the retransmission in wide steps, otherwise, retransmission goes slowly, which prevents any possible congestions. Please, refer to Figure 5 to see the changing of ***rateCA*** during the transmission.

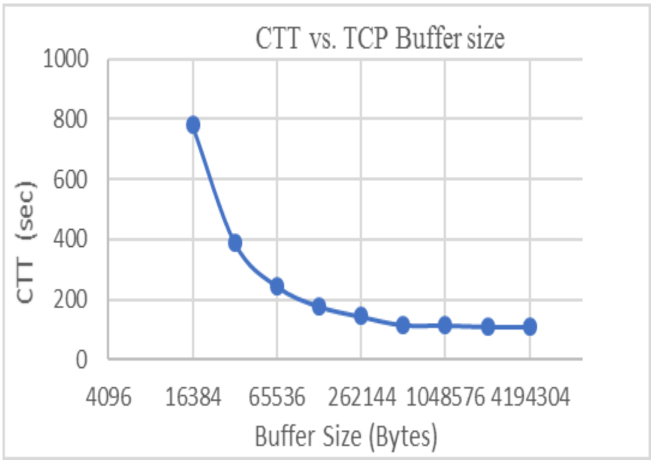

Figure 4. CTT vs. TCP buffer size

Figure 5 depicts the timing parameters for the simulation in Figure 1. We see that ***cwnd*** drops occur at the points: 21 sec, 90 sec, 151 sec, 168 sec and 240 sec. These points coincide with the spike points in Figure 5.

We discovered that if current ***RTT*** value is less than previous ***RTT***, then we have an increase in the ***cwnd*** size. Otherwise, if a spike occurs, then a packet is lost, and this is signaled through a spike on the RTT curve. When a spike occurs, RTO counter is exceeded, and a packet is lost. Hence, RTO timer is reset to 1, and this leads to the ***cwnd*** size to be reset to 1 packet. Look at Figure 5, and compare the time points of spikes and the ***cwnd*** drops.

At each time point during the transmission, the value of the next ***RTO*** is affected by the newly calculated ***rateCA***. If the current ***RTT*** is decreasing, then ***RTO*** shall be also reduced, as no congestion is expected. As described in the algorithm of ***CA*** phase in [1], next value of ***ssth*** depends on the available channel capacity, which is calculated regarding TCP Westwood+ algorithm [5],[21],[22]. The calculation of next ***cwnd*** depends on current ***rateCA*** and previous ***cwnd***.

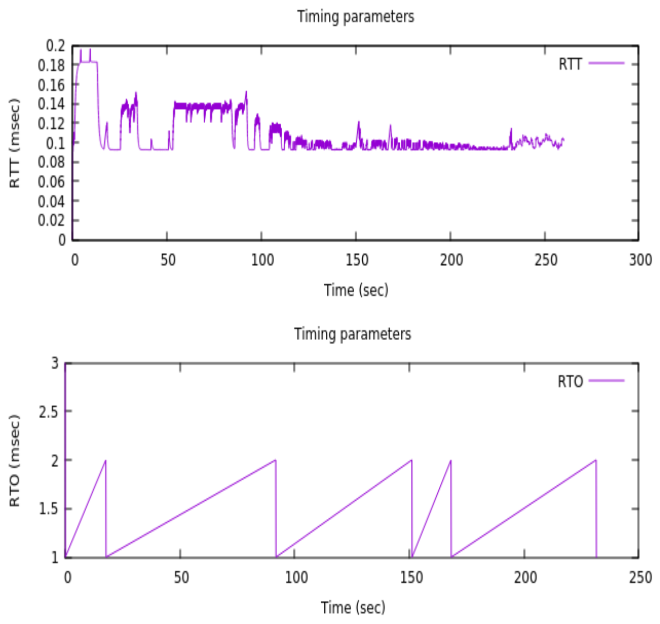

Figure 5. Timing parameters during the transmission

After we executed 1000's of simulations with different parameters, we found that our technique poses excellent stability and robustness properties.

Our simulations of the mentioned topology for many cases with different parameters show that next ***cwnd*** does not exceed the available ***ssth***. According to the theoretical results of the simulations, we make the assumption that DCM+ does not suffer or cause any congestion events, because it estimates the available channel capacity before sending data. More complex simulations are still to be executed to intensively study fairness and friendliness in the presence of other TCP sources and destinations.

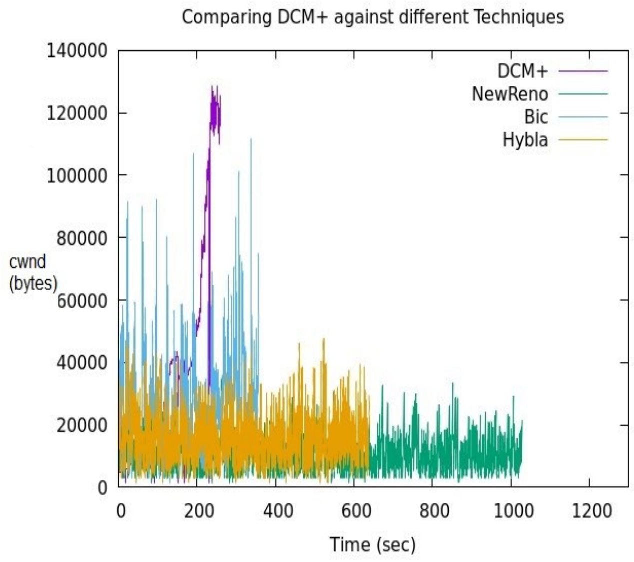

Figure 6. DCM+ performance compared with other techniques

We see that the quick tracking of the state of ***ssth*** and the smart way of selecting the transmission size are the main reasons for the improved performance and robustness of DCM+, as depicted in Figure 6, which is created with same parameters as Figure 1. Even better results are expected for higher bandwidth-delay-products due to the quick dynamic behavior of ***cwnd*** that is not available in other techniques.

IV. RESULTS

Table 1 depicts the used parameters to create Figures 7, 8 and 9. The simulations are executed for different packet error rates (1e-6 to 0.05). The used environment is ns-3.29 [33] under Ubuntu Linux VM inside Oracle VirtualBox 5.2.22.

TABLE I. PARAMETERS OF THE SIMULATION ENVIRONMENT

Data size	BW	Access BW	MTU Size	Duration (sec)
100 MB	1 Gb/sec	100 Mbps	1500 Bytes	2000

Figures 7, 8 and 9 below show the performance metrics for some TCP congestion control protocols (DCM+, NewReno, BIC, Ledbat and Hybla). Newer approaches like TCP CUBIC, TCP PCC and TCP ex Machina are to be compared against our approach in other works.

A. Throughput

In Figure 7, we see the throughput of different protocols, and we clearly see the advantage of DCM+ over other protocols. The high throughput extends nearly over the complete range of error rates, which is from 1e-6 to 0.05. For error rates less than 1e-3, only BIC protocol performs better, but that is at the expense of other metrics like PDR, average delay and packets losses, where BIC performs worst. Lost packets of BIC are highest in the range 1e-5 to 1e-3.

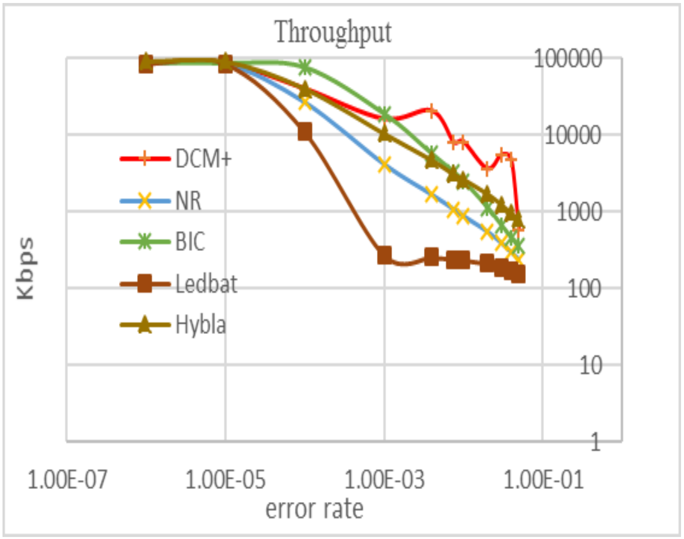

Figure 7. Throughput for different Protocols

B. Normalized Advancing Index (NAI)

For the reason of detailed comparison, we introduced a new metric, which we called *normalized advancing index* (***NAI***). It is defined as the ratio of throughput divided by the product of lost packets (given in bytes) and error rates. Its unit is (1/sec), and should indicate the speed of delivering the complete size of data from one end to the other despite the existence of lost packets at a specific error rate.

The robustness of DCM+ is visible in Figure 8. It shows that DCM+ performs better than all other protocols mentioned in this paper. This robustness is a result of less packet losses, lower average delay and a higher throughput than other approaches.

$$NAI = \frac{Throughput}{(ErrorRate * LostPackets * MTUsize)} \qquad (3)$$

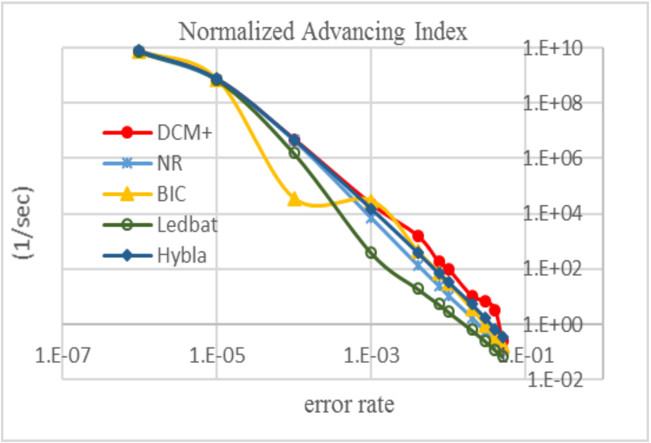

Figure 8. NAI as robustness indicator for different protocols

We clearly see that DCM+ has the best results over the whole range of simulated error rates. This reflects the best transmission speed and quality for the underlying TCP applications.

C. Complete Transmission Time (CTT)

It is a good advantage to finish transmission in short time without causing congestions, if possible. This is the case with DCM+ protocol as depicted in Figure 9. It has the lowest (***CTT***) among all tested protocols. ***CTT*** is defined as the time needed for the last ACK segment to arrive at the sender. We see from Figure 4 that the performance of ***CTT*** can be improved through changing the size of TCP buffer in the sender, receiver and intermediate router.

Based on the results presented above, TCP applications and devices that use DCM+ can extremely accelerate the data transmission and hence finish using the link earlier. This results in less power consumption.

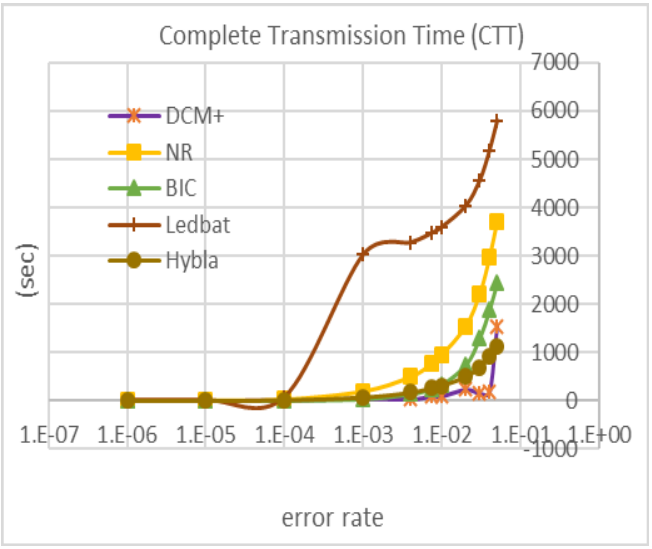

Figure 9. CTT for different protocols

V. CONCLUSION

We have demonstrated a new approach (DCM+) that has better performance than all other used approaches. We made the assumption that it does not cause any congestions as DCM+ is TCP fair and friendly. It is usable in the different types of networks, but more adequate for mobile/wireless and MANET networks. In this research work, we have shown that our approach is robust. It has the ability to minimize the average delay and packet losses, but also to improve the throughput and the speed of the transmission under high error rates. It is designed in similar fashion like TCP NewReno. It is an end-to-end technique, which will be used from the TCP sender to control the sent amount of data on the transmission link. It has a modified behavior in ***CA*** phase. It uses the ***BWE*** algorithm described in TCP Westwood+ protocol to estimate the available channel capacity. Thereafter, it calculates the appropriate values for both ***ssth*** and ***cwnd*** depending on the feedback signals ***RTT*** and ***RTO***, the parameter ***rateCA***, and whether the calculated ***cwnd*** is less than ***ssth*** or not. As feedback signals, we used previous states of both ***RTT*** and ***RTO.***

We found through intensive simulations that DCM+ has improved properties like high throughput, low delay, low drops and extremely fast speed in delivering data to the end device. We also introduced new performance metrics, *NAI* and *CTT* to show the advantages of the dynamic behavior of DCM+. In the future, these results are to be validated through more complex topologies in the presence of different traffic types. Also, a comprehensive mathematical model will be presented to show the theoretical limits of this approach. A comparison with newer techniques like CUBIC and ex Machina is planned as a future work.

REFERENCES

[1] R. Hamamreh and D. Khader, “DCM+: a multi-purpose protocol for congestion control”**,** 2019 IEEE 7th Palestinian International Conference on Electrical and Computer Engineering (PICECE)**,** Date of Conference: 26-27 March 2019, DOI: 10.1109/PICECE.2019.874723.

[2] V. Jacobson and M. J. Karels, “Congestion Avoidance and Control”, Computer Communication Review, 18(4), pp. 314 – 329, Aug. 1988.

[3] Y. Tian, K. Xu and N. Ansari, “TCP in Wireless Environments: Problems and Solutions”, IEEE Radio Comm., pp. S27 – S32, March 2005.

[4] S. Floyd and T. Henderson, “The NewReno Modification to TCP’s Fast Recovery Algorithm”, RFC 3782, 2004.

[5] L. Grieco and S. Mascolo, “Performance Evaluation and Comparison of Westwood+, New Reno, and Vegas”, ACM SIGCOMM Computer Communication Review, Volume 34 Issue 2, pp. 25-38, April 2004.

[6] K. Miller and L. Hsiao, “TCPTuner: Congestion Control Your Way”, Stanford University, 2016.

[7] P. Kaushika and R. Jagdish, “A survey on effectiveness of TCP Westwood in mixed wired and wireless networks”, International Journal of Scientific & Engineering Research, Volume 4, Issue 6, pp. 197 – 205, June-2013.

[8] S. Arianfar, “TCP’s Congestion Control Implementation in Linux Kernel”, Aalto University, 2012.

[9] T. Henderson, S. Floyd, A. Gurtov and Y. Nishida. “The NewReno Modification to TCP's Fast Recovery Algorithm”, RFC 6582. April 2012.

[10] P. Sarolahti and A. Kuznetsov, “Congestion Control in Linux TCP”, *Institute of Nuclear Research at Moscow, 2002.*

[11] A. Dell’Aera, L. A. Grieco and S. Mascolo, “Linux 2.4 Implementation of Westwood+ TCP with rate-halving: A Performance Evaluation over the Internet”, Tech. Rep. No. 08/03/S , 2004.

[12] Microsoft Networking Blog. Category- “Ledbat”. https://blogs.technet.microsoft.com/networking/category/windows-transports/ledbat/, [retrieved: September-2019].

[13] A.E. Eckberg and D.T. Luan, “Meeting the challenge: congestion and flow control strategies for broadband information transport”, 1989 IEEE Global Telecommunications Conference and Exhibition 'Comm. Technology for the 1990s and Beyond'. 1989.

[14] C. Yang and A.V.S. Reddy, “A taxonomy for congestion control algorithms in packet switching networks”, IEEE Network, Volume: 9, Issue: 4, pp. 34 – 45, 1995.

[15] K Bala, I. Cidon and K. Sohraby. “Congestion control for high speed packet switched networks”, IEEE INFOCOM, 1990.

[16] M. May, J. Bolot, C. Diot, and B. Lyles, “Reasons not to deploy RED”, Inria, Sprint Labs.

[17] R. Torres, J. Border, J. Xu and J. Jong, “Congestion control using RED and TCP window adjustment”, MILCOM 2012 - 2012 IEEE Military Comm. Conf., 2012.

[18] S. Floyd and V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance”, IEEE/ACM Transactions on Netw., Vol. I, No. I, pp. 397 – 413, 1993.

[19] J. Olsen, “On Packet Loss Rates used for TCP Network Modeling”, Dep. of Math., Uppsala Univ., Sweden. 2004.

[20] K. Tan, F. Jiang and Q. Zhang, “Congestion Control in Multihop Wireless Networks”, IEEE Transactions on Vehicular Technology, Vol. 56, No.2, March 2007.

[21] S. Mascolo, L.A. Grieco, R. Ferorelli, P. Camarda and G. Piscitelli, “Performance evaluation of Westwood+ TCP congestion control”, ResearchGate, uploaded in May 2014.

[22] S. Mascolo, “Testing TCP Westwood+ over Transatlantic Links at 10 Gigabit/Second rate”, (2005).

[23] C. Parsa and J.J. Garcia-Luna-Aceves, “Differentiating Congestion vs. Random Loss: A Method for Improving TCP Performance over Wireless Links”, Computer Engineering Dep., Baskin School of Engineering, University of California.

[24] M. Allman, V. Paxson and E. Blanton, “TCP Congestion Control”, RFC: 5681. 2009.

[25] M. H. Yaghmaee, F. Fatemipour. M. Bahekmat and A. Barasani, “A New Fuzzy Logic Approach for TCP Congestion Control”, Researchgate, 2015.

[26] H. Elaarag and M. Wozniak, “Using Fuzzy Inference to improve TCP congestion control over wireless networks”, BSc. Thesis, Stetson University, DeLand, Florida. 2010.

[27] S. M. Hosseini and B. N. Araabi, “A Neuro-Fuzzy Control for TCP Network Congestion”, Advances in Intelligent and Soft Computing, Springer Verlag, Sep. 2009.

[28] I. Elkhayat, P. Geurts and G. Leduc, “Enhancement of TCP over wired/wireless networks with packet loss classifiers inferred by supervised learning”, Tech. Report. Montefiore Inst., Belgium. 2004.

[29] P. Geurts, I. Elkhayat and G. Leduc, “A Machine Learning Approach to Improve Congestion Control over Wireless Computer Networks”, University of Li`ege, Belgium, 2005.

[30] S. Alavandar, “ANN Based Intelligent Congestion Controller for High Speed Computer Networks”, Journal of Electrical Engineering. 2015.

[31] L. Niu, “Applying the Linear Neural Network to TCP Congestion Control”, Fuyang Teachers College, china, Published by Atlantis Press, 2015.

[32] P. Yang, J. Shao, W. Luo, L. Xu, J. S. Deogun and Y. Lu, “TCP Congestion Avoidance Algorithm Identification”, CSE Journal Articles, IEEE/ACM Transactions on Networking, Vol. 22, No. 4, August 2014.

[33] Ns-3 network simulator. Website: https://www.nsnam.org/, [retrieved: September-2019].