Scientific Research Publishing

Int. J. Communications, Network and System Sciences, 2019, 12, 113-123
https://www.scirp.org/journal/ijcns
ISSN Online: 1913-3723
ISSN Print: 1913-3715

Adaptive Control of Congestion in Tough Wireless Environments Using DCM+ Algorithm

Rushdi Hamamreh, Derar Khader

Department of Electronics and Computer Engineering, Al-Quds University, Jerusalem, Palestine
Email: rushdi@staff.alquds.edu, cedkhader@gmail.com

How to cite this paper: Hamamreh, R. and Khader, D. (2019) Adaptive Control of Congestion in Tough Wireless Environments Using DCM+ Algorithm. *Int. J. Communications, Network and System Sciences*, **12**, 113-123.
https://doi.org/10.4236/ijcns.2019.128009

Received: July 18, 2019
Accepted: August 25, 2019
Published: August 28, 2019

Abstract

The aim of this paper is to present DCM+, a new congestion control protocol for data networks. It stands for Dynamic Congestion control for Mobile networks. New metrics have been newly invented and introduced like *normalized advancing index* (*NAI*) and *complete transmission time* (*CTT*). The simulations are done for a simple single-hop-topology (*sender-router-receiver*). The outcomes of this protocol are excellent and, in most cases, better than other approaches. The excellent properties of our proposed protocol were possible through tracking the available slow-start threshold. We achieved performance improvement, minimized end-to-end delay and large reduction in transmission time. DCM+ was able to combine many advantages at same time of the protocols NewReno and Westwood+. The results show, that DCM+ is extremely adequate for different types of networks. Feedback as main principle of control theory was used to control the congestion in the network. The parameters Round-Trip-Time (*RTT*) and Retransmission Timeout (*RTO*) are used as feedback signals to adjust the next congestion window (*cwnd*).

Keywords

Congestion Control, Westwood+, Slow-Start, Congestion Avoidance, Bandwidth Estimation, Round-Trip Time, Retransmission Timeout, Feedback, ns-3

1. Introduction

It is vital to manage the congestion in the networks, especially those, that rely mainly on TCP traffic. It helps achieve high performance and throughput. This will prevent the big collapse in the global networks like the internet [1] [2]. Congestion control exists as indispensable software module in most operating systems within the network stack [3] [4] [5] [6]. Many protocols have been proposed and implemented since ’86. Their role was to control the traffic between

the different nodes. TCP NewReno is one of those variants. It is a prominent one of old days [7] [8] [9] [10], which was found to have limitations, especially in wireless environments [2] [11] [12] [13]. TCP NewReno also shows another limitation, when it comes to mobility [2] [8] [12]. Experiments have shown, that in Wi-Fi and *MANETs* networks, NewReno has little chances to perform well.

TCP DCM+ [14] has come to mitigate those limitations of old schemes. DCM+ stands for *dynamic congestion control for mobile* systems. It makes use of the same TCP Westwood+ algorithm to estimate the available bandwidth. DCM+ is intended to avoid the problem of congestion in mobile networks, but it is also appropriate for wired networks. The novelty of this paper is the extended analysis of this protocol. It also extensively compares the metrics of performance for DCM+ against other well-known approaches. In this paper, we have proposed a technique that is appropriate for different types of networks, because it minimizes drops of congestion windows, and hence, minimizes delays.

This paper is structured as follows. We state in Section 2 the problems, which cause the degradation of performance in most congestion control approaches. In Section 3, related works and new methods and techniques for congestion control are presented. Then, in Section 4, we present our research work of DCM+. The results of the simulation are shown in TCP NewReno can't distinguish the reasons for packet losses [2] [11]. The main two reasons for packet losses are 1) the "full buffer" of the intermediate router [9] [15] [16] [17] [18] [19], and 2) a signal error on the wireless channel, which is known as link error (*LE*) [2] [11] [13] [20] [21]. In both cases, TCP NewReno behaves the same. It drops its *cwnd* to the half, even if no real network congestion exists, and the packet was only dropped because of a bad wireless link [2] [10] [21] [22] [23]. This is the reason for bad performance of TCP NewReno in wireless or mixed networks. So, a new approach is needed, that can distinguish the reason for packet loss. DCM+ has come to solve this problem and it shows intelligent behavior.

2. Related Works

Many approaches have been suggested to find the reason for packet loss [20] [23]. Modern research topics like Fuzzy Logic (*FL*) and Machine learning (*ML*) are some of these fields. Fuzzy inference systems [24] [25], ANFIS [26], ML classification [27] [28], neural networks [29] [30] and random forests are just some of the modern approaches to distinguish between *true* and *false* congestion events in the different types of networks. DCM+ is not causing any congestions during the transmission, because it does not aggressively increase the size of *cwnd* during the phase of congestion-avoidance (*CA*). It also helps achieve high performance by using previous values of round-trip-time (*RTT*) and retransmission-timeout (*RTO*). These parameters are needed to predict the probability of a network congestion, and thereafter to set the appropriate value for *cwnd*. This way, DCM+ extremely minimizes the possibility of a congestion. This behavior of DCM+ also helps speed-up the transmission and hence, outperforms other

TCP variants in various network topologies.

The used topology is of simple nature, and unneeded network complexity is avoided. This topology is intended to concentrate only on comparing the performance of the different protocols instead of causing additional network complexity and overhead. This topology is preferred because it causes minimal delay and presents best available performance of the different approaches. A comparison of Westwood+ with DCM+ shows similar performance, but Westwood+ shows much more drops in *cwnd* value, which indicates more congestion events. This is a big advantage for DCM+, especially in wireless or MANET environments.

3. Proposed Approach (DCM+)

DCM+ is an End-To-End (E2E) approach. For the bandwidth-estimation (BWE), the same algorithm explained in TCP Westwood+ [11] [22] [31] [32] will be used. It describes a sender-side modifications of TCP NewReno protocol in either phases (*SS* and *CA*). DCM+ needs no additional information as it is TCP protocol compliant. Hence, the overhead is equal to the standard TCP protocol header, not more. Depending on the current value of *BWE*, DCM+ calculates the future values of other parameters like *cwnd* and *ssthresh*. The behavior of *cwnd* is observed to be dynamic, in that it continuously tracks the state of slow-start threshold (*ssthresh*). If a change (increase or decrease) of *ssthresh* has been observed within a time interval, then DCM+ keeps using the current value of *cwnd* until a newer *ssthresh* has been settled. After that, *cwnd* moves to new value. It remains on the new state of *ssthresh* for the new time interval until a change of link capacity occurs. This way, *cwnd* will not exceed the available *ssthresh*. Hence, congestion events are extremely minimized. **Figure 1** depicts the behavior of DCM+ for packet-error-rate = 7.5e−3 and for maximum transmission unit (MTU size) = 1200 bytes. The design of DCM+ [14] is similar to NewReno, which is detailed in [10]. DCM+ is built from 4 phases like NewReno (*SS*, *CA*, fast retransmission (*FR*) and fast recovery (*FV*)). In DCM+, the behaviors in (*SS*) and (*CA*) have been so modified to enforce the *cwnd* to track *ssth* in the next interval. In addition to *cwnd*, *RTT* and *RTO* have been used as feedback to control the values of *ssth* and *cwnd* in the next interval.

The idea behind DCM+ design is, that a continuous increase in *RTT* value leads gradually to an increase in the length of the queue on the intermediate node. This results after some time in a *true* congestion event (*i.e.*, packet drop). As a consequence of the congestion, the network will suffer a higher delay. So, we tied this idea of congestion with the parameters *RTT* and *RTO*, and we introduced a new parameter called congestion rate or *rateCA*, which is an important indicator for the congestion on the link. This parameter is critical for determining the next appropriate values of *cwnd*. It is defined as the ratio of the previous *RTT* divided by the current RTT. As a result, all parameters (*cwnd*, *RTT*, *RTO* and *rateCA*) in the next interval are affected, and therefor dynamically changing during the transmission.

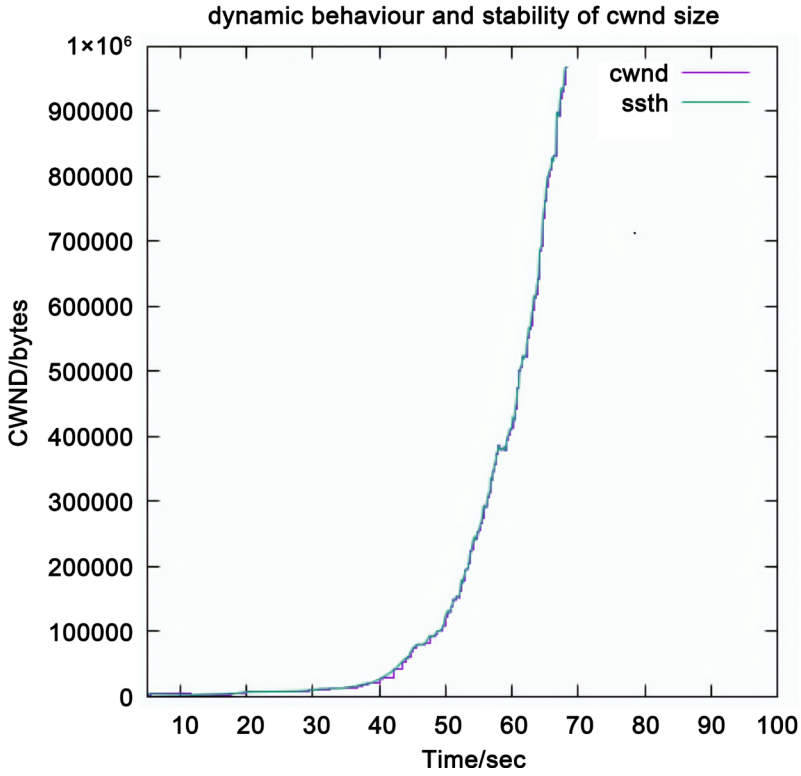

Figure 1. Dynamic behavior of cwnd in DCM+.

$$rateCA = \frac{RTT_{old}}{RTTnew} \tag{3.1}$$

Now, we consider the following two cases: 1) *rateCA* higher than 1 will be considered as *advance* or "Link Capacity Increasing", and 2) values of *rateCA* lower than 1 as *danger* or "Link Capacity Decreasing". Regarding the DCM+ algorithm given in [14], if *cwnd* is less than *ssth*, then *rateCA* will be used to start the retransmission in wide steps (increased size of *cwnd*), otherwise, retransmission goes slowly to prevent any possible congestions. Refer to **Figure 2** to see the status of *rateCA* during the transmission. At each time sample 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, because no congestion is expected. The following equations describe this behavior:

$$rateCA = \frac{RTT_{old}}{RTTnew} = \frac{RTO_{old}}{RTOnew} \tag{3.2}$$

Equation (3.2) can be reformulates as:

$$RTOnew = \frac{RTO_{old}}{rateCA} \tag{3.3}$$

As described in *CA* phase of DCM+, refer to [14], next value of *ssth* depends on the available channel capacity, which is calculated regarding TCP Westwood+ algorithm [11] [22] [32], while next *cwnd* depends additionally on the current *rateCA* and previous value of *cwnd*.

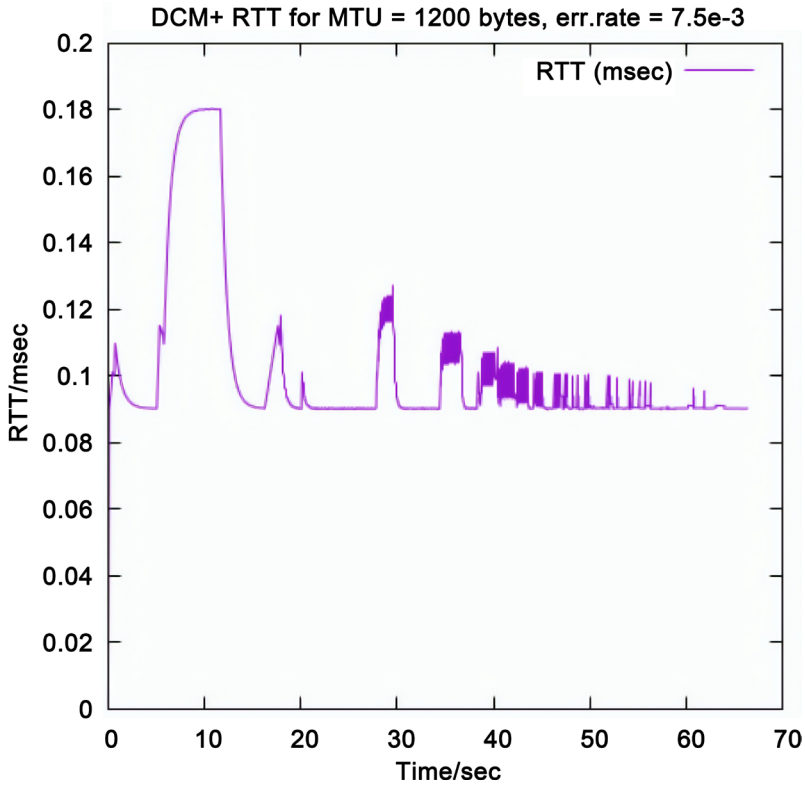

Figure 2. The changing of RTT as indicator of increase/decrease of cwnd.

Our simulations for many cases with different parameters like error rate, data size, MTU size, used TCP protocol, bottleneck bandwidth and access bandwidth show, that next *cwnd* does not exceed *ssth*, and hence barely cause new congestion events. We figure out, that *cwnd* is changing dynamically as a reaction to the changing channel capacity. This can be reflected as a higher improvement of throughput, complete-transmission-time (*CTT*) and normalized-advancing-index (*NAI*).

4. Results

Following figures show the measured values of performance metrics using ns-3 simulator for different TCP congestion control protocols (DCM+, NewReno, BIC, Ledbat and Hybla). The simulations are executed for different packet error rates. The simulation environment is ns-3.29 [33] under Ubuntu Linux VM under Oracle VirtualBox 5.2.22. The following parameters in **Table 1** are constant throughout the tests:

4.1. Throughput

In **Figure 3**, we see the plots of throughput for different protocols, and we clearly see the advantage of DCM+ over most protocols. This high throughput extends over the most range of error rates, which is from 1e−6 to 0.05. For error rates less than 1e−3, we see that only BIC protocol performs better, but that is at the expense of other metrics like packet delivery ratio (PDR), average delay and normalized advancing index, where BIC performs worst.

4.2. 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/second), 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. From Figure 4 we figure out, that DCM+ performs better than all other protocols.

$$NAI = \frac{\text{Throughput}}{\text{ErrorRate} * \text{Packetslost} * \text{MTUsize}} \tag{4.1}$$

Table 1. Simulation environment's parameters.

Data size	BW	Access BW	MTU Size	Simulation Duration (sec)
100 MB	1 Gb/sec	100 Mb/sec	1500 Bytes	5000

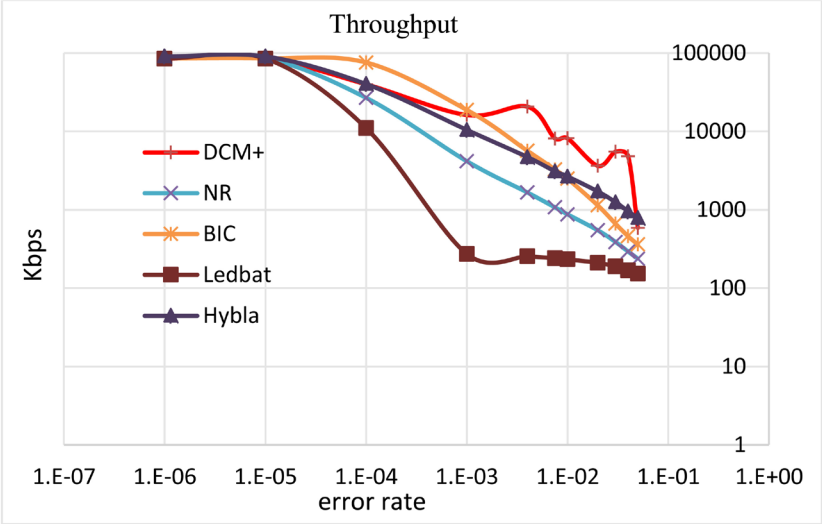

Figure 3. Throughput comparison with different TCP Protocols.

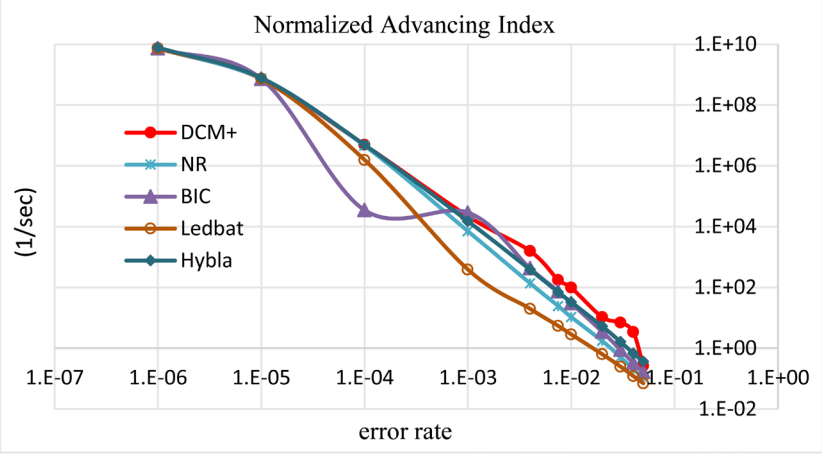

Figure 4. NAI comparison with different protocols.

We clearly see that DCM+ has the best results for all used error rates. This reflects best transmission speed and quality for the used applications.

4.3. Complete Transmission Time (CTT)

It is a good advantage to finish transmission in short time without causing congestions, if possible. This is the case for DCM+ protocol as depicted in Figure 5. It has the lowest (*CTT*) among all tested protocols, which is defined as the time needed for the last ACK segment to arrive at the sender. Based on that result, TCP applications and devices can extremely speedup data transmission and stop using the link earlier. This results in less power consumption. For the error rate (0.05), only TCP Hybla performs a little better.

5. Assessment of the Results

We have demonstrated, that our proposed approach (DCM+) owns a better behavior than the other TCP approaches used in this paper. It may change from one case to another, but DCM+ remains faster, provides higher throughput and shows less false drops. The simulations are done for different situations like packet error rates and sizes of the sent packets (MTU).

Following two cases are considered:

Case 1: *same error rate*, *different MTU sizes*

In the following we see, that unwanted drops are occurring as a side effect of lowering the MTU size. For example, if MTU = 600 bytes, we have a lower time to finish transmission compared with the case MTU = 1500, but it suffers 3 unwanted drops. On the other hand, if MTU = 1500 bytes, we have longest time to finish transmission (*CTT*), but only one unwanted drop. It is also clear, that the performance for MTU = 1200 is the best. It finishes transmission faster and suffers no unwanted drops, as shown in Figure 6.

Figure 6 gives a hint, that an additional increase of the performance of DCM+ may be possible if the used MTU size is optimized before the transmission.

***Case* 2**: *different error rates*, *same MTU size*

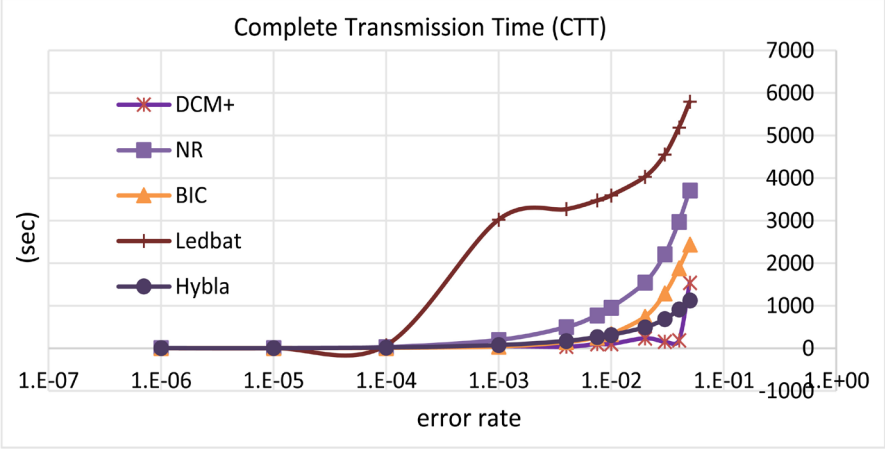

Figure 5. Complete Transmission Time (CTT).

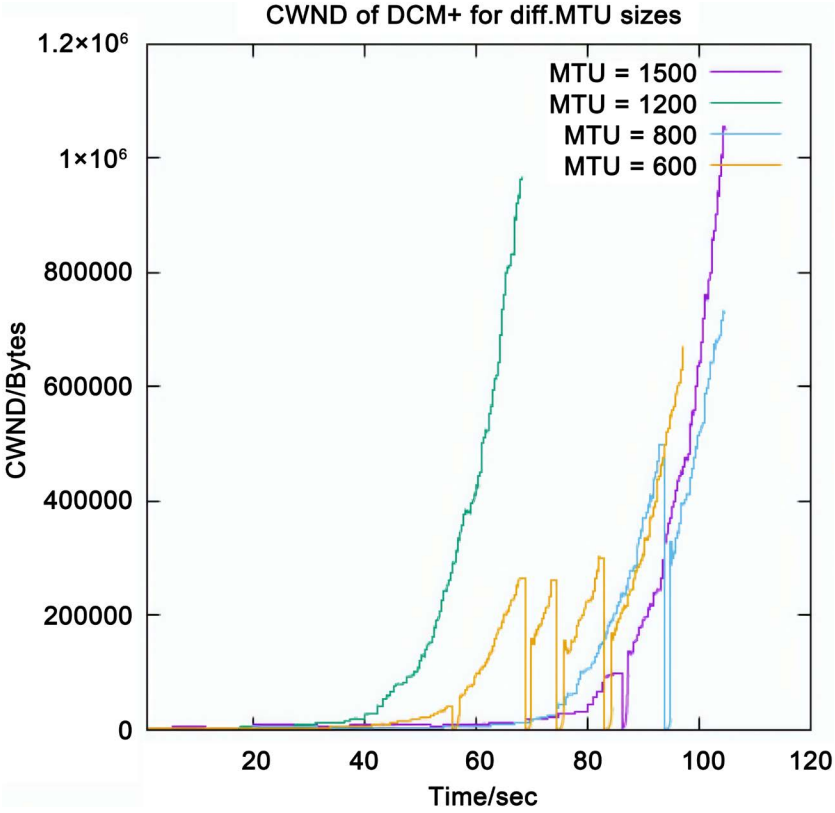

Figure 6. cwnd for different MTU sizes, but same error rate.

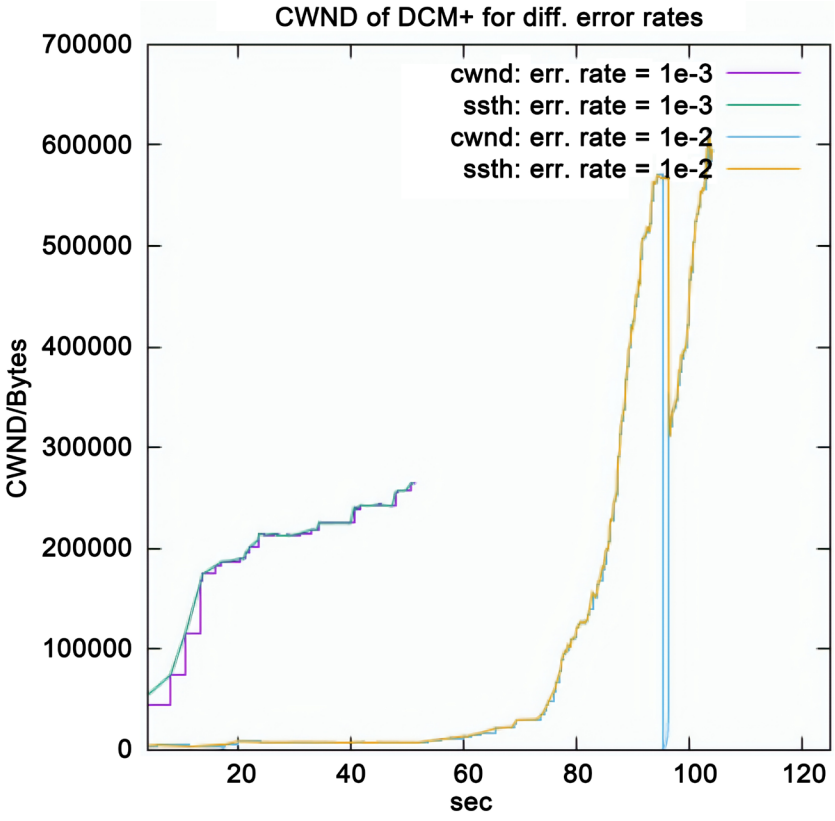

Figure 7. cwnd size for different error rates, but same MTU size (1500 bytes).

In Figure 7, the size of data to be transmitted is (100 MB), and the used error rates are: (1e−3, 1e−2). We see, that for the error rate = 1e−3, the transmission is much quicker (*CTT* is reduced by the half), and the throughput is around 200%. Besides, it does not make any false drops.

In Figure 8, we present a comparison of different TCP congestion control protocols with MTU = 1500 bytes, and error rate = 0.0075. The advantage of DCM+ over the other protocols is clear. DCM+ shows best performance among all tested approaches. It finishes transmission much faster; it has the highest throughput and it causes no congestions on the transmission link. The figure shows clearly, that the *cwnd* values are exactly tracking the available bandwidth. This is of great benefit for new devices, that may be currently using the same channel.

6. Conclusion

We have presented a protocol, that can be used in wireless, mixed and MANET networks. It is robust and adaptive for all changes in its environment. It has the ability to minimize the average delay and packet loss, but also to improve the throughput and the speed of the transmission even under high error rates. DCM+ is a new approach for managing congestions in mobile and wired networks. 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

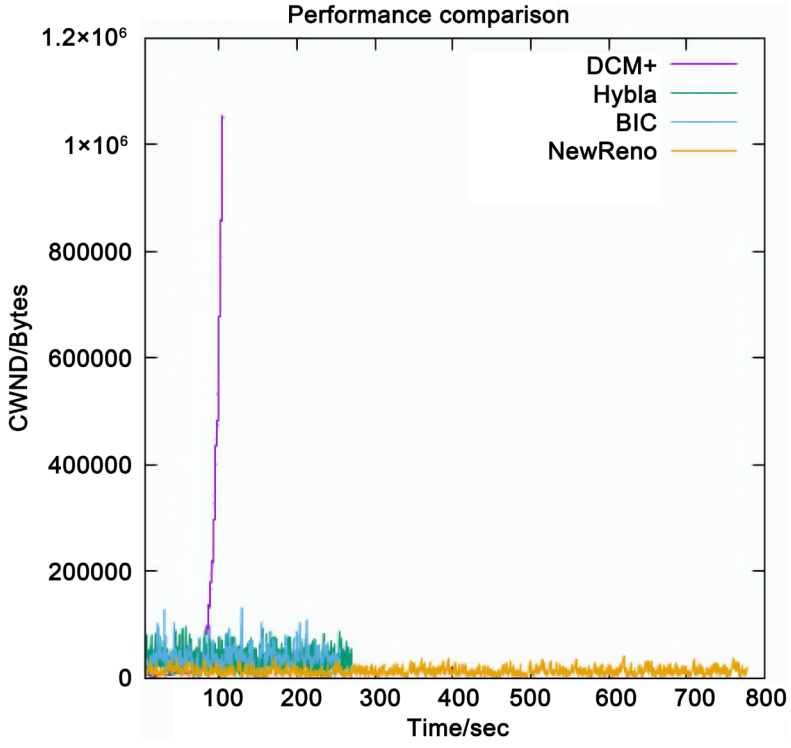

Figure 8. Comparing performance of different CC schemes.

channel capacity. Thereafter, it calculates the appropriate value for *cwnd* depending on the feedback signals *RTT* and *RTO*, the parameter *rateCA*, and whether the calculated *cwnd* is less than *ssth* or not.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Jacobson, V. (1988) Congestion Avoidance and Control. *ACM SIGCOMM Computer Communication Review*, **18**, 314-329. https://doi.org/10.1145/52325.52356

[2] Tian, Y., Xu, K. and Ansari, N. (2005) TCP in Wireless Environments: Problems and Solutions. *IEEE Communications Magazine*, **43**, S27-S32.

[3] Miller, K. and Hsiao, L. (2016) TCPTuner: Congestion Control Your Way.

[4] Arianfar, S. (2012) TCP's Congestion Control Implementation in Linux Kernel.

[5] Sarolahti, P. and Kuznetsov, A. (2002) Congestion Control in Linux TCP.

[6] Microsoft Networking Blog. Category-Ledbat. 2018. https://blogs.technet.microsoft.com/networking/category/windows-transports/ledbat/

[7] Floyd, S., Henderson, T. and Gurtov, A. (2004) The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 3782. https://doi.org/10.17487/rfc3782

[8] Henderson, T., Floyd, S., Gurtov, A. and Nishida, Y. (2012) The NewReno Modification to TCP's Fast Recovery Algorithm, RFC 6582. https://doi.org/10.17487/rfc6582

[9] Eckberg, A.E. and Luan, D.T. (1989) Meeting the Challenge: Congestion and Flow Control Strategies for Broadband Information Transport. 1989 *IEEE Global Telecommunications Conference and Exhibition* "*Communications Technology for the 1990s and Beyond*", Dallas, TX, 27-30 November 1989, 1769-1773.

[10] Yaghmaee, M.H., Fatemipour, F., Bahekmat, M. and Barasani, A. (2015) A New Fuzzy Logic Approach for TCP Congestion Control.

[11] Grieco, L. and Mascolo, S. (2004) Performance Evaluation and Comparison of Westwood+, New Reno, and Vegas. *ACM SIGCOMM Computer Communication Review*, **34**, 25-38. https://doi.org/10.1145/997150.997155

[12] Patel, K. and Rathod, J. (2013) A Survey on Effectiveness of TCP Westwood in Mixed Wired and Wireless Networks. *International Journal of Scientific and Engineering Research*, **4**, 197-205.

[13] Tan, K., Jiang, F and Zhang, Q. (2007) Congestion Control in Multihop Wireless Networks. *IEEE Transactions on Vehicular Technology*, **56**, 863-873. https://doi.org/10.1109/TVT.2007.891405

[14] Hamamreh, R. and Khader, D. (2019) DCM+: A Multi-Purpose Protocol for Congestion Control. 2019 *IEEE 7th Palestinian International Conference on Electrical and Computer Engineering* (*PICECE*), Gaza, Palestine, 26-27 March 2019, 1-7. https://doi.org/10.1109/PICECE.2019.8747237

[15] Yang, C.-Q. and Reddy, A.V.S. (1995) A Taxonomy for Congestion Control Algorithms in Packet Switching Networks. *IEEE Network*, **9**, 34-45. https://doi.org/10.1109/65.397042

[16] Bala, K., Cidon, I. and Sohraby, K. (1990) Congestion Control for High Speed Packet Switched Networks. *IEEE INFOCOM'90*, San Francisco, CA, 3-7 June 1990, 520-526.

[17] May, M., Bolot, J., Diot, C. and Lyles, B. (1999) Reasons Not to Deploy RED. 1999 *Seventh International Workshop on Quality of Service*, London, UK, 31 May-4 June 1999, 260-262.

[18] Torres, R., Border, J., Choquette, G., Xu, J. and Jong, J.-H. (2012) Congestion Control Using RED and TCP Window Adjustment. *MILCOM* 2012-2012 *IEEE Military Communications Conference*, Orlando, FL, 29 October-1 November 2012, 1-6. https://doi.org/10.1109/MILCOM.2012.6415799

[19] Floyd, S. and Jacobson, V. (1993) Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, **1**, 397-413. https://doi.org/10.1109/90.251892

[20] Olsen, J. (2004) On Packet Loss Rates for TCP Network Modeling.

[21] Parsa, C. and Garcia-Luna-Aceves, J.J. (2000) Differentiating Congestion vs. Random Loss: A Method for Improving TCP Performance over Wireless Links. 2000 *IEEE Wireless Communications and Networking Conference*, Chicago, IL, 23-28 September 2000, 90-93.

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

[23] Allman, M., Paxson, V. and Blanton, E. (2009) TCP Congestion Control. RFC 5681. https://doi.org/10.17487/rfc5681

[24] Elaarag, H. and Wozniak, M. (2010) Using Fuzzy Inference to Improve TCP Congestion Control over Wireless Networks. BSc. Thesis, Stetson University DeLand, Florida.

[25] Hadi Hosseini, S.M. and Araabi, B.N. (2009) A Neuro-Fuzzy Control for TCP Network Congestion. In: Mehnen, J., Köppen, M., Saad, A. and Tiwari, A., Eds., *Applications of Soft Computing. Advances in Intelligent and Soft Computing*, Vol. 58, Springer, Berlin, Heidelberg, 93-101. https://doi.org/10.1007/978-3-540-89619-7_10

[26] El Khayat, I., Geurts, P. and Leduc, G. (2004) Enhancement of TCP over Wired/Wireless Networks with Packet Loss Classifiers Inferred by Supervised Learning. Tech. Report, Montefiore Inst., Belgium.

[27] Geurts, P., El Khayat, I. and Leduc, G. (2005) A Machine Learning Approach to Improve Congestion Control over Wireless Computer Networks. University of Liège, Belgium.

[28] Shajahan and Alavandar, S. (2015) ANN Based Intelligent Congestion Controller for High Speed Computer Networks. *Journal of Electrical Engineering*.

[29] Niu, L. (2015) Applying the Linear Neural Network to TCP Congestion Control. 5*th International Conference on Advanced Design and Manufacturing Engineering*.

[30] Yang, P., Shao, J., Luo, W., Xu, L., Deogun, J.S. and Lu, Y. (2014) TCP Congestion Avoidance Algorithm Identification. *IEEE/ACM Transactions on Networking*, **22**, 1311-1324. https://doi.org/10.1109/TNET.2013.2278271

[31] Dell'Aera, A., Grieco, L.A. and Mascolo, S. (2004) Linux 2.4 Implementation of Westwood+ TCP with Rate-Halving: A Performance Evaluation over the Internet. 2004 *IEEE International Conference on Communications*, Paris, 20-24 June 2004, 2092-2096. https://doi.org/10.1109/ICC.2004.1312887

[32] Mascolo, S., Grieco, L.A., Ferorelli, R., Camarda, P. and Piscitelli, G. (2004) Performance Evaluation of Westwood+ TCP Congestion Control. *Performance Evaluation*, **55**, 93-111.

[33] ns-3 Network Simulator. https://www.nsnam.org/