D-TCP: Dynamic TCP Congestion Control Algorithm for Next Generation Mobile Networks

Madhan Raj Kanagarathinam, Sukhdeep Singh and Irlanki Sandeep
Samsung R&D Institute India Bangalore
Bangalore, India
Email: madhan.raj@samsung.com

Abhishek Roy
Samsung Electronics
Suwon, South Korea

Navrati Saxena
Sungkyunkwan University
Suwon, South Korea

***Abstract*—In the past few decades, many Transmission Control Protocol (TCP) congestion control algorithms have been investigated to meet the growing network demands and to enhance the performance of TCP in lossy or high-bandwidth-delay-product (high-BDP) networks. However, it is still challenging to implement a dynamic congestion control algorithm for wide range of diverse mobile users, network conditions and applications. This paper explores avenues for enhancement of TCP congestion control algorithm for next generation mobile networks by dynamically learning the available bandwidth and deriving the Congestion Control Factor *N*. *N* is used to Adaptive Increase/Adaptive Decrease (AIAD) the Congestion Window (CWND) dynamically instead of using the traditional approach that is Additive Increase/ Multiplicative Decrease (AIMD) paradigm. Once there is congestion, our proposed algorithm will not allow the CWND to decrease multiplicatively or steeply. After dropping to a certain level (lower than legacy), we try to take the CWND to previous state adaptively with the help of calculated bandwidth (based on learning). This in turn helps to efficiently control the CWND for better network utilization especially in case of lossy and high-BDP conditions. As soon as it reaches the original state, it remains stable for longer time as compared to legacy until packet loss or time out. We demonstrate the effectiveness of our algorithm with the help of live air experiments (performed in Samsung R&D India, Bangalore) and NS3 based simulation experiments. Through our experiments, we show that our algorithm outperforms the legacy congestion control algorithms (like CUBIC, RENO, TCPW) and the existing CLTCP algorithm in terms of goodput, intra algorithm fairness and inter algorithm fairness maintaining the scalability and friendliness.**

***Index Terms*—LTE-A, TCP, Congestion Control, Bandwidth Estimation, Throughput**

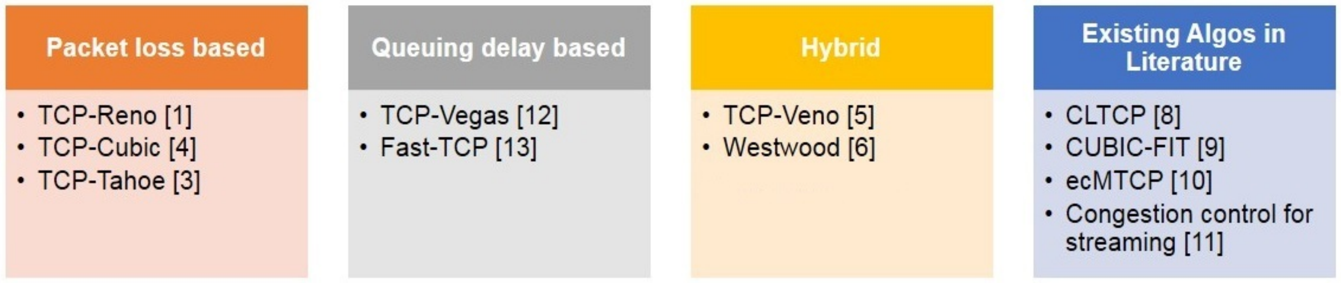
I. INTRODUCTION

The Transmission Control Protocol (TCP) has been adopted widely for reliable and error-checked delivery by extensive applications like streaming media, peer-to-peer file sharing, email, file transfer, World Wide Web (www), remote administration, secure shell and the like. On the other hand, mobile networks have evolved substantially over a decade. The next generation networks aim to provide ultra low latency, high throughput and massive connectivity. With the evolution of mobile networks, many real time mobile applications have emerged demanding high bandwidth and less delay, which at times may cause congestion in the network. Moreover, with the introduction of next generation mobile network services like Internet of Things (IoT), D2D (Device to Device) communications, Internet of Vehicles (IoV), the network conditions change dynamically.

Due to bursty and dynamic nature of next generation mobile data traffic there might be sporadic losses caused due to high Bandwidth Error Rate (BER), network signal condition change, mobility, congested path, handoff problems, lengthy or frequent disconnections and the like. When a user tries to upload a video while traveling, he might experience data stalling due to varying network conditions. On the other hand, during peak hours when a network saturates due to high number of users in a particular cell, there might be high packet loss. The standard congestion control algorithms (like TCP Reno [1], TCP New Reno [2], TCP Tahoe [3] and TCP Cubic [4]) are not designed to dynamically control or minimize these losses and in turn, leads to unnecessary drops in Congestion Window (CWND), which drastically decrease the efficiency of TCP mobile receivers or senders. Most of these TCP variants increase CWND aggressively to address this issue but sometimes fail to control the congestion in case of high load, lossy, dynamic or high-bandwidth-delay-product (high-BDP) networks. Few more variants introduced later (like TCP Veno [5], TCP Westwood [6]) addressed the congestion control issue in mobile networks but failed to achieve higher goodput than afore-mentioned algorithms. This motivated us to design a dynamic algorithm to overcome the aforesaid losses and to attain higher goodput than the existing algorithms while maintaining friendliness and fairness.

Contributions: We propose D-TCP-Dynamic TCP congestion control algorithm for next generation mobile networks. More precisely, major contributions of our paper are:

1) D-TCP provides a real time estimation of available bandwidth of mobile networks. We provide a system that can estimate the real time bandwidth by applying C2D (Continuous to Discrete) time variant filter based on the traffic flowing. We extract the end-to-end characteristics (like traffic intensity, link capacity, packet sending rate) to derive the available bandwidth. D-TCP can quickly adapt to the new circumstances and achieve good estimation performance in case of abrupt changes or burst loss conditions thereby, avoiding the aliasing effect.
2) Based on the estimated bandwidth, we derive the dynamic congestion control factor *N*, which is used to increase/decrease the CWND during the RTT update and



Packet loss based	Queuing delay based	Hybrid	Existing Algos in Literature
• TCP-Reno [1]	• TCP-Vegas [12]	• TCP-Veno [5]	• CLTCP [8]
• TCP-Cubic [4]	• Fast-TCP [13]	• Westwood [6]	• CUBIC-FIT [9]
• TCP-Tahoe [3]			• ecMTCP [10]
			• Congestion control for streaming [11]

Fig. 1. Summary of Related Work

loss detection respectively.

3) We measure the mobile network performance characteristics and adaptively modifies the CWND based on the dynamic congestion control factor N measured. By doing so, D-TCP can avoid CWND drops caused due to spurious packet loss. It can update the CWND appropriately during burst losses, hence, utilizing the complete available bandwidth (partially inspired by Parallel TCP [7]). Different from Parallel TCP, we do not open multiple sockets for full utilization but we control the CWND growth efficiently. Opening multiple sockets is not desirable as it violates the fairness and also impacts the power consumption of mobile networks. Our objective is to develop reliable and efficient available bandwidth measurement method with no or very few requirement of resources for data processing and memory.
4) We simulated D-TCP on a live network using Samsung Galaxy S6 edge. We also performed NS3 based simulations to evaluate the applicability of D-TCP in various network and traffic scenarios. We compared our D-TCP with the legacy congestion control algorithms (such as CUBIC, RENO, Westwood, Tahoe) and the existing CLTCP algorithm [8] in terms of goodput, intra protocol fairness and inter protocol fairness while maintaining the scalability and friendliness.

The remainder of this paper is organized as follows: Section II delineates an overview of related work on the existing TCP congestion control algorithms. The proposed D-TCP algorithm and the system model is outlined in Section III. Section IV specifies the simulation parameters used during live experiment and NS3 based simulations followed by evaluation of experimental results. Finally, concluding remarks are provided in Section V.

II. Related Work

This section provides an overview of the existing literature work of TCP congestion control algorithms. Fig. 1 provides the overview of exsiting TCP congestion control algorithms. Several researchers have worked on increasing the performance of TCP congestion control variants. All the proposals put forward the concept of full utilization of bandwidth while avoiding packet loss. Authors in [8] proposed an adaptive TCP congestion control algorithm i.e. CLTCP that uses congestion levels for controlling the virtual parallel flow numbers in a TCP connection, which avoids delay measurement error witnessed in TCP-FIT [9]. CUBIC-FIT is the another approach proposed by Jingyian Wang et al. in [10] that helped to improve the network performance over large range of conditions and simultaneously maintains the fairness with TCP CUBIC servers. ecMTCP [11] is a multipath TCP congestion control algorithm that takes in to account energy efficiency by moving traffic from the higher energy cost paths to that of lower ones and more congested paths to comparatively lighter paths. This in turn helps to balance the load and promote energy savings. Authors in [12] proposes a congestion control algorithm on the basis of MAC layer contention state for improving streaming services. Each flow's rate of transmission is controlled using estimated contention states notified by marking packets of each flow.

For estimating congestion level, different techniques have been discussed in the literature. These includes Loss based TCP which estimates congestion level based on packet loss. Loss based TCP variants are classified into TCP Reno [1], TCP Tahoe [3] and TCP CUBIC [4]. Delay-Based TCP is based on Queuing Delay that includes TCP Vegas [13] and FAST TCP [14]. On the other hand, there is another estimation, which considers both Packet Loss and Queuing Delay, known as Hybrid TCP. It includes TCP Veno [5], TCP Westwood [6], and Westwood+ [6]. In wireless conditions, the mechanism which detects timeout and packet loss as congestion could be inaccurate. In addition there are error control mechanism (missing ACK because of transmission error) used as indication for congestion. However, TCP cannot differentiate these conditions and act accordingly.

The above mentioned TCP variants cannot understand the available bandwidth and differentiate suprious wireless loss. This end up in reduction of TCP goodput even though there is enough available bandwidth. Therefore, we propose a dynamic approach to control the congestion.

III. Proposed Approach: D-TCP

This section provides an overview of proposed approach. We focus on the bandwidth estimation of D-TCP followed by calculation of dynamic congestion control factor N. We utilize the N to control the CWND. Finally, we present the D-TCP system model.

A. Bandwidth Estimation

This section focuses on the Bandwidth estimation algorithm [6] to determine the estimated bandwidth BW_E. The estimated bandwidth B_i at instance i is:

$$B_i = \frac{d_i}{t_i - t_{i-1}} = \frac{d_i}{\Delta_i} \tag{1}$$

where d_i is the data transferred between interval t_{i-1} and t_i and Δ_i is time difference delta.

We model the estimated available bandwidth as a discrete time linear system. We use a discrete time filter using the Tustin approximation. We derive the filtered available bandwidth B_I at interval T_I as:

$$B_I = \left[\frac{\frac{2\tau}{\Delta_i} - 1}{\frac{2\tau}{\Delta_i} + 1} \times (B_{I-1})\right] + \left[\frac{B_i + B_{i-1}}{\frac{2\tau}{\Delta_i} + 1}\right] \tag{2}$$

where $\frac{1}{\tau}$ is the cut off frequency. Consider a constant α_i which is written as

$$\alpha_i = \frac{2\tau - \Delta_i}{2\tau + \Delta_i} \tag{3}$$

Hence, Using the constant α_i (2) can be rewritten as,

$$B_I = \alpha_i B_{i-1} + (1 - \alpha_i)\left(\frac{B_i + B_{i-1}}{2}\right) \tag{4}$$

We assume the α_i as 0.90 (to give more weightage to our previous learning). Hence equation (4) can be rewritten as

$$B_I = 0.90 B_{i-1} + 0.10 B_i \tag{5}$$

Packet aliasing may occur while using the filter. So, to avoid packet aliasing effect, we normalize B_I from low pass filter output and the B_i to obtain final estimated bandwidth BW_E as follows:

$$norm = B_i - B_I \tag{6}$$

$$BW_E = B_I + g \times norm \tag{7}$$

norm is an error normalization parameter and g is clock granularity(set at 0.5 in our trail to match with the epoch timer). Bandwidth fluctuates widely in the mobile wireless conditions, hence we introduce parameter *norm* to cope up with the fluctuations.

B. Calculation of N

Queue length of a D-TCP flow is proportional to TCP window Size W. Hence the expected queue length $E(Q)$ for our single flow can be written as :

$$E(Q) = \varphi \times E[W_{SF}] \tag{8}$$

where, φ is boost and $E[W_{SF}]$ is the expected window size of the single TCP flow. Here, $\varphi \varepsilon (0,1)$. As we create N different flow in D-TCP, (8) can be re-written as,

$$E(Q) = \alpha \times \frac{E[W]}{E[N]} \tag{9}$$

Where $E[W]$ is expectation of W and $E[N]$. To obtain the queue length in (9), during $i+1^{th}$ we update the parameter N_{i+1} using

$$N_{i+1} \leftarrow max\left(1, N_i + 1 - \frac{Q_i}{\alpha \times wi} N_i\right) \tag{10}$$

Where N_i, Q_i and W_i are N, queue length and window size of the i^{th} period, respectively. The bytes-in-flight queue length can be estimated as:

$$Q_i = (BW_E - BW_C) \times \frac{W_i}{BW_E} \tag{11}$$

Where BW_E is the estimated bandwidth and BW_C is the current bandwidth, which are same as described in above section. Combining (10) and (11), we have expression for updating N:

$$N_{i+1} \leftarrow max\left(1, N_i + 1 - \frac{BW_E - BW_C}{\alpha \times BW_E} W_i\right) \tag{12}$$

C. CWND Adjustment

In D-TCP algorithm, congestion window is adjusted according to:

$$For\ each\ RTT : CWND \leftarrow CWND + N_{i+1} \tag{13}$$

$$For\ each\ Loss : CWND \leftarrow CWND(1 - y) \tag{14}$$

$$where\ y = \frac{\eta}{[(2\eta - 1)N] + 1)} \tag{15}$$

where η is convergence factor for fairness(set as 2 in our trails to avoid aggressiveness).

D. D-TCP Modeling

Triple Duplicate ACK (TD) is considered as loss indication in our modeling [15]. Let TP be the goodput and it can be estimated as:

$$TP = \lim_{t \to \infty} TP_t = \lim_{t \to \infty} \frac{S_t}{t} \tag{16}$$

where S_t is the no. of segments sent during time interval t. From the packet loss probability p we derive $TP(p)$. Lets assume during i^{th} NDT (Non-Dup Ack Time) P_i packets are transmitted in the duration D_i then the steady state TCP goodput can be derived from (16) as :

$$TP = \frac{EP[P]}{EP[A]} \tag{17}$$

$$CWND = CWND + \frac{\Upsilon \times \Upsilon}{cwnd} \tag{18}$$

where Υ is segment size. Now, we will explain how CWND changes during each period. Once TD is triggered, the NDP starts. CWND size will be cut down according to (18) during loss indication from TD. In congestion Avoidance (CA) phase, CWND increases as:

$$CWND = CWND + \frac{\Upsilon \times \Upsilon}{cwnd} + \frac{\Upsilon}{8} \tag{19}$$

We can denote CWND as W_i window segments and hence (19) can be written as:

$$W_i = W_i + \frac{N}{W_i} \tag{20}$$

So during first round, CWND on receiving first ACK will be $W_i + \frac{1}{W_i}$. As we increase N times during each round W_i grows by $\frac{N}{W_i}$. On receiving the first ACK, second round commences and CWND size is only $W_i = W_i + \frac{N}{W_i}$

Currently, the second round starts as soon as the first ACK is received. CWND size at the start of the second round is only $W = W + \frac{N}{W}$. In case of our proposed methodology, left over ACKS for unacknowledged segments of 1st round $W_i = W_i - 2$ have reached the destination. So, before getting 1st ACK of 2nd round, we get $\frac{W_i}{N-1}$ ACKS for the packets transmitted in 1st round. 2nd round commences with $CWND \leftarrow W_i + \frac{1}{W_i} \times \frac{W_i}{N} = W_i + \frac{1}{N}$ instead of CWND of $W_i + \frac{1}{W_i}$. Algorithm 1 delineates the overall flow of D-TCP.

Algorithm 1 Proposed Algorithm

```
 1: Initialization:
 2: $CWND \leftarrow 2 \times \Upsilon, ssthresh \leftarrow 65535$
 3: D-TCP_reset()
 4: Packet Loss:
 5: $CWND \leftarrow CWND - CWND \times \frac{\eta}{[(2\eta-1)N]+1)}$
 6: $ssthresh \leftarrow CWND$
 7: D-TCP_reset():
 8: $N \leftarrow 1, \alpha = 0.2$
 9: $epoch\_start \leftarrow tcp\_time\_stamp()$
10: Timeout:
11: $CWND \leftarrow 2 \times \Upsilon$
12: D-TCP_reset()
13: N_update():
14: if $tcp\_time\_stamp() - epoch\_start > update\_epoch$ then
15:     $epoch\_start \leftarrow time\_stamp$
16:     if $currentBW < estimatedBW$ then
17:         $temp \leftarrow max(1.0, prenValue)$
18:         $nValue \leftarrow nValue + temp$
19:         $prenValue \leftarrow nValue$
20:     end if
21: else if $currentBW > estimatedBW$ then
22:     $nValue \leftarrow nValue + 1$
23:     $prenValue \leftarrow nValue$
24: end if
```

IV. Performance Evaluation

This section evaluates the live experiment and NS3 based simulation results.

A. Live Air Experiment Results

We performed live air experiments in Sasmsung R&D India-Bangalore (SRI-B) using two Samsung Galaxy S6 edge devices (one with TCP-Cubic and another with proposed D-TCP algorithm) by varying the RTT. We measured the network goodput using each device that are connected via wifi routers to different servers situated in Bangalore (India),

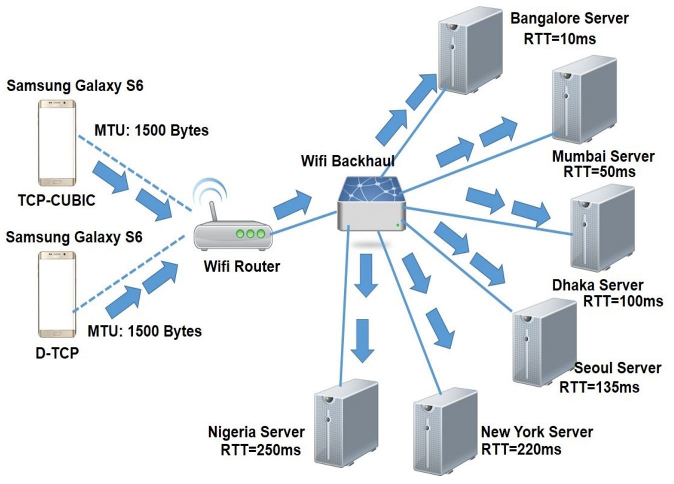


Fig. 2. Setup for live air experiment

Mumbai (India), Dhaka (Bangladesh), Seoul (South Korea), New York (USA) and Nigeria (Africa). Fig. 2 depicts the overall experimental setup. Through our experiments, we

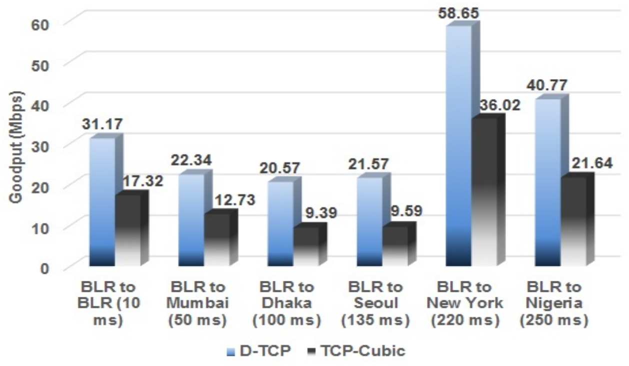


Fig. 3. Live Air Experiment

found that D-TCP outperforms the TCP-Cubic algorithm in terms of goodput. Fig. 3 depicts that the average goodput percentage gain achieved when the devices are connected to server situated in Bangalore (10 ms RTT) is 70.98%. In case, when the server is situated in Mumbai (50 ms RTT), Dhaka (100 ms RTT), Seoul (135 ms), New York (220 ms) and Nigeria (250 ms) the average goodput percentage gain is 75.42%, 118.99%, 124.92%, 62.83% and 88.4% respectively. The phenomenal percentage gain attributes to the intelligent selection of dynamic factor N, which smartly controls the CWND and does not allow it to fall steeply in case of congestion. This in turn helps to utilize the maximum available bandwidth leading to higher goodput values in case of our proposed D-TCP.

B. NS3 Simulation based Experiment Results

NS3 based simulations are performed in an LTE environment, where a device capable of running proposed algorithm is compared with the different legacy algorithms and the

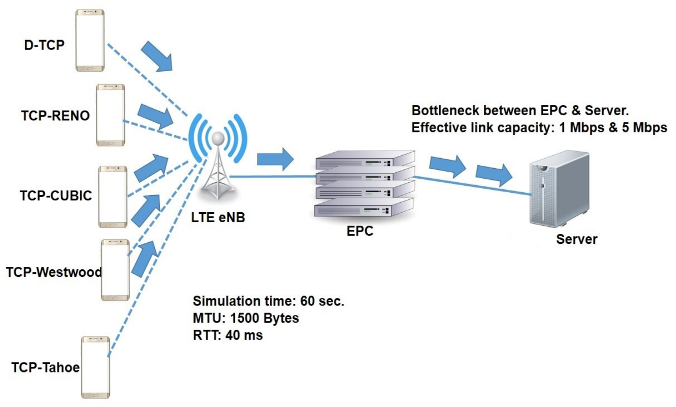


Fig. 4. Simulation setup for NS3 experiment

existing CLTCP [8] algorithm. We measure the accuracy of our algorithm based on CWND values, efficiency in case of lossy, lossless as well as ideal state networks, intra algorithm fairness and inter algorithm fairness. Fig. 4 provides an overview of the experimental setup.

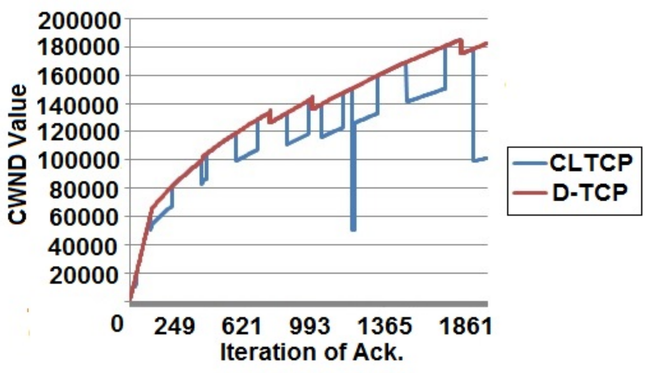


Fig. 5. CWND growth comparison

1) CWND values: We measured the CWND values in random loss conditions with the bottleneck data rate of the channel set to 1 Mbps, RTT delay as 40ms and MTU as 1500. From Fig. 5, we infer that in case of existing CLTCP [8] algorithm deep drops are acknowledged and the CWND raises aggressively. In case of bursty loss conditions, it takes more time for the algorithm to recover to its previous state. In case of our proposed D-TCP, CWND drops are very rare and it is able to utilize the maximum available bandwidth. As explained in Section III, while estimating bandwidth, we used discrete-time low-pass filter to derive estimated bandwidth values from current and previous bandwidth values. Packet aliasing may have occurred after using the aforesaid filter so, we normalized the output of estimated bandwidth, which is the main reason behind no deep drops in D-TCP.

2) Efficiency in lossy, lossless and near ideal conditions: We measure the efficiency of D-TCP in 0.01, 0.0001 as well as 0.000001 packet loss ratio conditions and compared it with TCP-Reno, TCP-Cubic and CLTCP. The bottleneck data rate

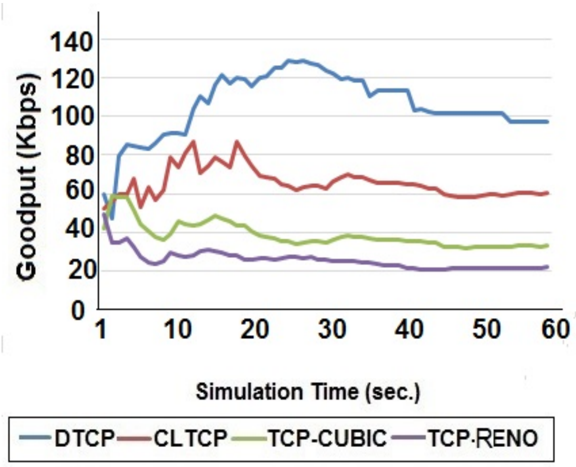


Fig. 6. Goodput comparison in lossy conditions (packet loss ratio=0.01)

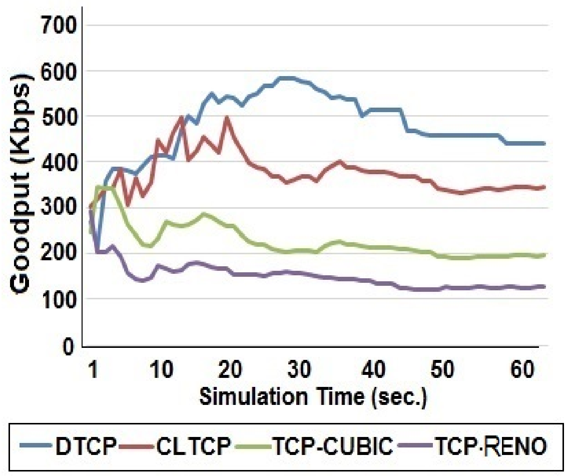


Fig. 7. Goodput comparison in lossless conditions (packet loss ratio=0.0001)

of the channel is set to 5 Mbps with RTT delay as 40ms and MTU as 1500. In case of lossy and lossless network with packet loss ratio of 0.01 and 0.0001 respectively, D-TCP is able to utilize the maximum available bandwidth as compared to other TCP variants as depicted Fig. 6 and Fig. 7. The similar trend can be seen Fig. 8, in case of 0.000001 packet loss ratio. We introduced wireless losses, where the real packet drops does not happen. D-TCP can detect those losses and avoid CWND drops whereas TCP-Cubic, CLTCP and TCP-Reno assumes those wireless losses as congestion in network The maximum goodput in case of D-TCP is 4927 Kbps, which is again substantially higher than the existing algorithms: TCP-Cubic with 2379.68 Kbps, TCP-Reno with 3706.93 Kbps and CLTCP with maximum goodput of 4579.66 Kbps.

3) Intra-Algorithm Fairness: To prove intra-algorithm fairness, we take five connections of D-TCP, introduced one after other. Fig. 9, shows that when different D-TCP connections are introduced one after the other, the bandwidth gets equally shared amongst the five available devices and after introducing

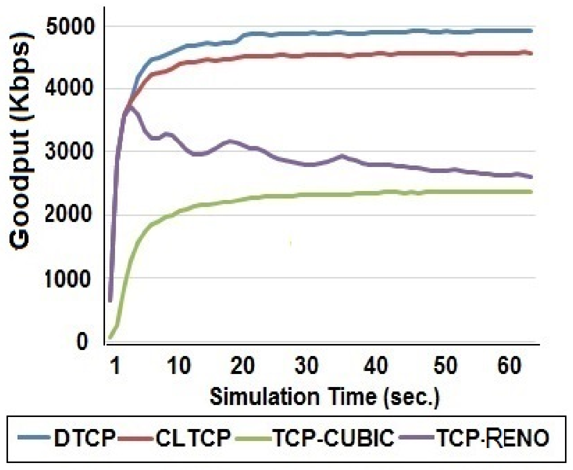


Fig. 8. Goodput comparison in near ideal conditions (packet loss ratio=0.000001)

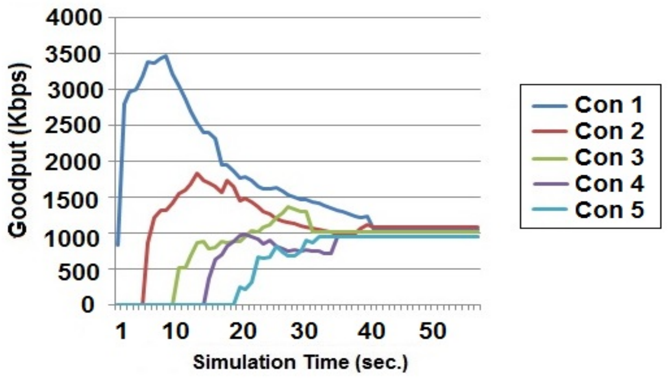


Fig. 9. Intra-Algorithm fairness comparison

all the five connections it converges to the fairness point.

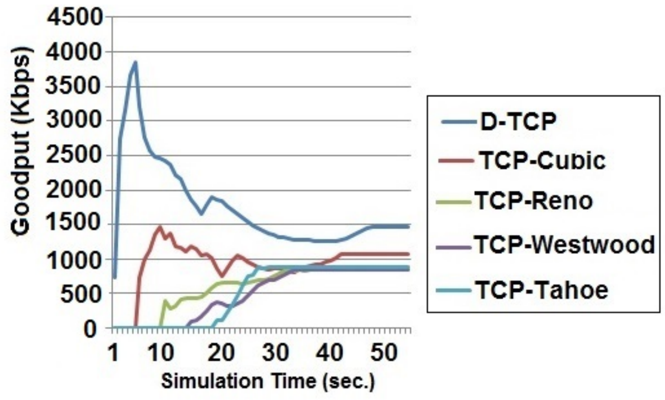


Fig. 10. Inter-Algorithm fairness comparison

4) Inter-Algorithm Fairness: To prove intra-algorithm fairness, we take five devices with different TCP variants. One connection of 5 Mbps is shared by all the five variants. We infer from Fig. 10 that D-TCP has very low BPR (Bandwidth Plucking Rate), whereas, other variants have comparatively high BPR. Due to low BPR, D-TCP is able to even use the un-used bandwidth leading to higher goodput as compared to other TCP variants.

V. Conclusion

This paper proposes a robust TCP congestion control algorithm, which works efficiently in various network conditions (RTT, Packet loss, etc.). Unlike existing works in this domain, D-TCP estimates available bandwidth. It is able to distinguish between packet loss caused due to transmission errors (or other wireless errors) and packet loss due to congestion. It avoids unnecessary packet drops or CWND drops, thereby utilizing the available bandwidth completely and providing substantially higher goodput while maintaining TCP friendliness and TCP fairness. Our future work presupposes a cross layer bandwidth detection and dynamic control of CWND to increase the accuracy of Bandwidth and Handover estimations. We plan to use network parameters such as, SINR (signal to noise interference ratio), RSSI (Signal Indicator) and operational parameters like CPU parameters (governer, clock speed), UE Battery State, UE parallel upload, Upload type (File size, File type, File Priority), etc.

References

[1] V. Jacobsen and M. Karels, Congestion avoidance and control, in Proc. ACM SIGCOMM, 1988.
[2] M. Allma, V. Paxson, and W. Stevens, Tcp congestion control, RFC 2581, April 1999.
[3] S. Floyd and T. Henderson, The newreno modification to tcps fast recovery algorithm, RFC 2582, April 1999.
[4] Sangatae Ha, Injong Rhee, and Lisong Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", ACM SIGOPS Operating Systems Review, Vol. 42, No. 5, 2008.
[5] C. P. Fu and S. C. Liew, Tcp veno: Tcp enhancement for transmission over wireless access networks, in IEEE Journal on Selected Areas in Communication, Feb 2003.
[6] Zehang Chen; Yitong Liu; Yameng Duan; Hao Liu; Gang Li; Yami Chen; Junshuai Sun; Xin Zhang "A novel bandwidth estimation algorithm of TCP westwood in typical LTE scenarios", in Proc. ICCC, 2015
[7] Mohamed A. Alrshah; Mohamed Othman, "Performance evaluation of parallel TCP, and its impact on bandwidth utilization and fairness in high-BDP networks based on test-bed", in Proc. IEEE MICC, 2013
[8] Xianliang Jiang; Guang Jin, "CLTCP: An Adaptive TCP Congestion Control Algorithm Based on Congestion Level" IEEE Communications Letters, Vol. 19 No. 8, 2015.
[9] Wang, Jingyuan, Jiangtao Wen, Jun Zhang, Zhang Xiong, and Yuxing Han. "TCP-FIT: An improved TCP algorithm for heterogeneous networks", Journal of Network and Computer Applications, 2016.
[10] Jingyuan Wang; Jiangtao Wen; Yuxing Han; Jun Zhang; Chao Li; Zhang Xiong, "CUBIC-FIT: A High Performance and TCP CUBIC Friendly Congestion Control Algorithm" IEEE Communications Letters, Vol. 17 No. 8, 2013
[11] Tuan Anh Le; Choong Seon Hong; Md. Abdur Razzaque; Sungwon Lee; Heeyoung Jung ecMTCP: An Energy-Aware Congestion Control Algorithm for Multipath TCP", IEEE Communications Letters, Vo. 16 No. 2, 2012
[12] H.-j. Lee; J.-t. Lim, "Congestion control for streaming service in IEEE 802.11 multihop networks", IET Communications, Vol. 4 No. 12, 2010.
[13] L. Brakmo and L. Peterson, Tcp vegas: New techniques for congestion detection and avoidance, in Proc. ACM SIGCOMM, Aug 1994.
[14] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, Fast tcp: Motivation, architecture, algorithms, performance, in IEEE/ACM Trans. on Networking, 2007.
[15] "Modeling TCP Throughput and Delay" online available at: http://www.ece.virginia.edu/mv/edu/715/lectures/TCP/tcp-model.pdf