CLTCP: An Adaptive TCP Congestion Control Algorithm Based on Congestion Level

Xianliang Jiang and Guang Jin

***Abstract*—The Transmission Control Protocol (TCP) has been profusely used by diverse applications such as FTP, email, and HTTP. In recent decades, numerous TCP variants have been developed to fit the fast increasing of network capacities and improve the performance of TCP algorithms in various scenarios, particularly in the high-bandwidth-delay-product (high-BDP) and lossy networks. Although the performance improvements are different for different TCP under different networking conditions, implementing a new congestion control algorithm that is suitable for a wide range of network conditions is still a challenge. In this letter, we propose a novel scheme, i.e., the congestion-level-based TCP (CLTCP), which could perform effectively in both high-BDP and lossy networks. Different from the TCP-FIT, the CLTCP does not use the delay variation but the congestion level as a signal to control the number of *virtual* parallel flows in a TCP connection. This could avoid the influence of the delay measurement error in the TCP-FIT. Extensive experiments in NS-2 show that the performance of the CLTCP is significantly improved, as compared to other state-of-the-art algorithms, while maintaining good fairness.**

***Index Terms*—Virtual parallel TCP, congestion level, high bandwidth-delay product network, lossy network.**

I. Introduction

THE Transmission Control Protocol (TCP) using the Additive Increase Multiplicative Decrease (AIMD) mechanism [1] has been instrumental to the successful development and growth of Internet. However, due to the prevalence of high bandwidth-delay product (high-BDP) and lossy networks, the existing AIMD mechanism is ill-suited. For example, TCP requires 8333 RTTs, or 27 minutes, to recover from one packet loss in the scenario of 1 Gbps link with 1500 byte packets and 200 ms RTT. Furthermore, the increasing diversity of applications carried over Internet has stressed the congestion control mechanism in TCP. Hence, how to design a more efficient, fair, robust, and easy-to-deploy alternative congestion control mechanism has gain in importance.

The standard TCP algorithms, such as Reno [1], have achieved great success. But they are found to perform poorly over high-BDP and lossy networks. To improve the performance over these networks, many TCP variants have been proposed, including Veno [2] for lossy networks and CUBIC [3] for high-BDP networks. Although these algorithms have achieved success in their respective target scenarios, designing and implementing a TCP variant algorithm that performs efficiently in both high-BDP and lossy networks is still a challenge. Furthermore, with the deployment of high-speed wireless networks as well as high bandwidth, real time applications, the TCP variants have to handle both non congestion loss as well as congestion-introduced issues. In short, the study of TCP algorithms is still significance and encounter new problems.

In [4], *Afanasyev et al.* divides the end-to-end TCP variants into three categories, namely Loss-based TCPs, Delay-based TCPs and Hybrid TCPs. Therein, Loss-based TCPs, such as Reno [1], CUBIC [3], uses the packet loss as a symptom of network congestion. However, Loss-based TCPs suppose that packet losses are caused by overloading the network only. It's no longer reasonable in lossy networks. Furthermore, Loss-based TCPs require a very low (10^{-7} or lower) random packet loss rate to fully utilize network capacity in high-BDP networks. This is far from the reality of the network condition nowadays. In Delay-based TCPs, such as FAST [5], the variation of queuing delay was used as a symptom of network congestion. Massive studies show that Delay-based TCPs are more resilient to transient changes of network conditions and also suitable for high-BDP networks. However, when Loss-based and Delay-based TCPs coexists, Loss-based TCPs would lead to the bandwidth "starvation" for Delay-based TCPs. Furthermore, in the event of a reverse congestion, the queuing delay cannot be estimated correctly. This also degrades the performance of Delay-based TCPs. In Hybrid TCPs, such as Compound TCP [6], ACCF [7], both packet loss and queuing delay were used for congestion control. Compared with two former schemes, Hybrid TCPs can obtain better performance for high-BDP and lossy networks. But they are difficult to perform well in high bandwidth wireless networks.

In this letter, we propose a novel TCP algorithm, namely the Congestion Level based TCP (CLTCP), for high-BDP and lossy networks. The algorithm was motivated by the TCP-FIT [8] and DCTCP [9] algorithms. Therein, the former uses N virtual Reno sessions, which can be adjusted according to the end-to-end queuing delay, to simulate a single Reno session. The latter uses explicit congestion notification (ECN) [10] to estimate the congestion level and adjust the MD factor in TCP Reno. Different from TCP-FIT, CLTCP doesn't use the end-to-end queuing delay but the congestion level like DCTCP algorithm to control N virtual Reno sessions. Extensive experiments in NS-2 simulator show that the performance of the CLTCP is significantly improved as compared to other state-of-the-art algorithms, while maintaining good fairness.

Manuscript received April 28, 2015; revised May 31, 2015; accepted June 15, 2015. Date of publication June 19, 2015; date of current version August 10, 2015. This work was supported in part by the Natural Science Foundation of Zhejiang Province of China under Grant LY12F02013 and in part by the Ningbo Municipal Technology Innovation Team of China under Grant 2011B81002. The associate editor coordinating the review of this paper and approving it for publication was B. Rong.

X. Jiang is with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310012, China (e-mail: jiang86@mail.ustc.edu.cn).

G. Jin is with the Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo 315211, China (e-mail: jinguang@nbu.edu.cn).

Digital Object Identifier 10.1109/LCOMM.2015.2447541

II. Congestion Level

How to estimate the congestion level is a hot topic in the TCP congestion control. In [9], *Alizadeh et al.* believe that the ECN bits stream can reflect the congestion level. Using a simple marking scheme at routers for setting the congestion experienced (CE) codepoint of packets when the router's backlog exceeds a fixed threshold, the sender can derive multi-bit feedback from the information present in the single-bit sequence of marks and estimate the congestion level precisely. Indeed, one can also view path delay information as implicit multi-bit feedback to measure the extent of the network congestion. However, the path delay measurement is frequently interfered by reverse congestion and route change. Therefore, the similar strategy like [9] is used for estimating the congestion level in this letter.

In practice, the RED (a well-known active queue management scheme) implemented by most modern routers is employed and only one parameter, the marking threshold K, is necessary. Both the low and high thresholds of the RED algorithm are fixed to K and packets are marked based on instantaneous, instead of average queue length. The CE codepoint of an arriving packet is marked when the queue occupancy is greater than K. Otherwise, it is not marked. This ensures that the sender can be quickly notified. In [9], $K > C \cdot RTT/7$ (C is the bottleneck bandwidth). Upon receiving the marked packet, the receiver echoes the congestion signal back by setting the ECN-Echo (ECE) bits in ACK packets. Hence, by maintaining a weighted average of the fraction of marked ACKs, the sender can measures the congestion level, α, like [9] for every window of data by

$$\alpha \leftarrow (1-g)\cdot\alpha + g\cdot F \tag{1}$$

where $0 < g < 1$ is a weighted factor for estimating α. In [9], $g < 1.386/\sqrt{2(C\cdot RTT + K)}$. α represents a single point of the queue size distribution at the bottleneck queue. F is the fraction of packets that were marked in the last window of data and can be calculated by

$$F = \frac{M_a}{T_a} \tag{2}$$

where M_a is the number of marked ACKs. T_a is the total number of ACKs. As can be seen, the more marked packets make α bigger.

III. The CLTCP Algorithm

In this section, we present the principle and throughput model of the CLTCP algorithm, which is based on TCP Reno and uses the AIMD mechanism to adjust the congestion window W in the congestion avoidance stage directly. In TCP Reno algorithm, the W is inflated linearly when a new ACK is received. Otherwise, the W is reduced exponentially when a packet loss event occurs. Generally, the W of the AIMD mechanism can be updated by

$$\begin{aligned} &\text{For Each RTT: } W \leftarrow W + a \\ &\text{For Each Loss: } W \leftarrow (1-b)\cdot W \end{aligned} \tag{3}$$

where a and b are linear increasing and exponential decreasing factors respectively. In TCP Reno, $a = 1$ and $b = \frac{1}{2}$. In order to gain N times throughput that of a single TCP Reno session, the TCP-FIT [8] sets $a = N$ and $b = \frac{2}{3N+1}$. Actually, this modification was firstly proposed in [11]. But the N can be adjusted dynamically according to the end-to-end queuing delay in [8]. Using the same AIMD mechanism, the congestion window W of the CLTCP algorithm can be updated by

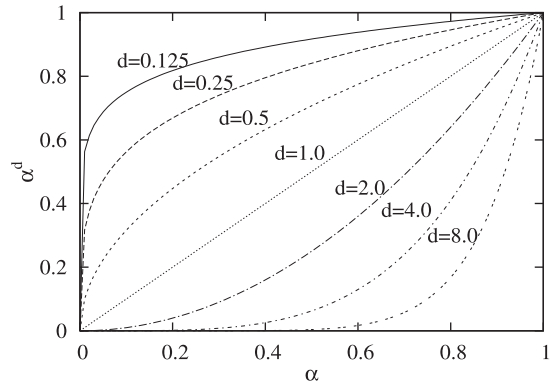

Fig. 1. α^d with different d.

$$\begin{aligned} &\text{For Each RTT: } W \leftarrow W + N \\ &\text{For Each Loss: } W \leftarrow \left(1 - \frac{2}{3\cdot N + 1}\right)\cdot W \end{aligned} \tag{4}$$

where N is an important parameter to control aggression in the CLTCP. If N is too small, the network capacity may not be fully utilized. Otherwise, other TCP flows may be starved to death. Similar to [8], the N is a dynamic parameter in the CLTCP and is updated periodically. For the i-th update period, the N can be updated by

$$N_i = N_{i-1} + 1 - \alpha^d \cdot N_{i-1} \tag{5}$$

where $\alpha \leq 1$ is the congestion level and can be computed by the expression (1). A large α denotes the high network congestion. When $\alpha \to 1$, the $N \to 1$. d is a controlled coefficient. Obviously, no matter what is the value of d, α^d in Eq. (5) is a gamma-correction function and always less than or equals 1. As an option, d can be expressed as the flows' deadline and priority, etc.

Fig. 1 plots a number of different curves on α^d with various values of d. Therein, The dot line in the middle is $d = 1$. The curves below the dot line are for $d > 1$, and the curves above the dot line are for $d < 1$. In this letter, d is fixed to 0.5 (empirical value). This balances the TCP friendly (inter-fairness) and efficiency of the CLTCP algorithm.

In aforementioned section, CLTCP simulates the behavior of N TCP Reno flows in a single TCP session to fully utilize the network capacity and uses the congestion level to adjust the N value. Next, we will introduce an approximate throughput model for CLTCP in the steady-state condition. The model follows the TCP Reno throughput model in [12], and assumes that all loss events are detected by triple-duplicate ACKs and does not consider the slow start and fast retransmission. In [12], the throughput T_{reno} of Reno can be represented as

$$T_{reno} = \frac{1}{RTT}\cdot\sqrt{\frac{3}{2\cdot PLR}} \tag{6}$$

where *PLR* is the averaged packet loss rate. *RTT* is the averaged round-trip time. Since CLTCP simulates N TCP Reno flows using one congestion window, the throughput model of CLTCP can be represented as

$$T_{cltcp} = N \cdot T_{reno} = \frac{N}{RTT} \cdot \sqrt{\frac{3}{2 \cdot PLR}} \tag{7}$$

Plugging (5) into (7), the throughput model of CLTCP for the given averaged *PLR* and *RTT* is

$$T_{cltcp} = \frac{1}{\alpha^d \cdot RTT} \cdot \sqrt{\frac{3}{2 \cdot PLR}} \tag{8}$$

Suppose that *RTT* is composed of propagation delay D and queuing delay q_d, and *PLR* is composed of the inherent losses P of the link and the congestion losses p, the throughput model of CLTCP can be rewritten to

$$T_{cltcp} = \frac{1}{\alpha^d \cdot (D + q_d)} \cdot \sqrt{\frac{3}{2 \cdot (P + p)}} \tag{9}$$

According to (5) and (9), N is increased when networks are not fully utilized to improve performance. Conversely, CLTCP decreases N to maintain friendliness with TCP Reno flows when network suffers from congestion.

IV. Efficiency and Fairness

To obtain insight into the performance of the CLTCP algorithm, in this section, we use a simple model to analyze its efficiency and fairness (TCP friendly). In this model, U TCP flows indexed by $u \in \{1, \cdots, U\}$ share a bottleneck link simultaneously. The bandwidth of the bottleneck link is B. When U TCP flows pass through the bottleneck link, the relationships among the congestion level α, the congestion packet loss rate p and the aggregate throughput of U TCP flows, $T_U = \sum_{u \in U} T_u$, are given by

$$T_U = \sum_{u \in U} T_u = \begin{cases} \leq B, \text{ if } \alpha = 0, \ p = 0, & (a) \\ = B, \text{ if } \alpha = g(q_c), \ p = f(q_c), & (b) \end{cases} \tag{10}$$

where q_c is the number of packets in the bottleneck queue. T_u is the throughput of flow u.$g(.)$ and $f(.)$ are non-decreasing function of q_c. They are determined by the queue management algorithm in the bottleneck queue. In this letter, we assume that $g(q_c < K) = 0$ and $f(0) = 0$.

Theorem 1: If a CLTCP flow belongs to U flows, the bottleneck link operates in (b) of (10).

Proof: Suppose that the bottleneck link operates in (a) of (10), the aggregated throughput, T_U, will be less than or equals B and α is always zero. According to (9), the CLTCP's throughput, T_{cltcp}, is close to positive infinity. This contradicts $T_U \leq B$. Therefore, the bottleneck link is always in (b) of (10). □

As can be seen from the Theorem 1, the CLTCP can fully utilize the network capacity. However, according to (6), we cannot obtain the similar conclusion for TCP Reno.

Next, we analyze the fairness of CLTCP with TCP Reno, e.g. TCP friendliness.

Theorem 2: If CLTCP flows and TCP Reno flows coexist and the bottleneck link operates in (b) of (10), the CLTCP is TCP friendly.

Proof: In Theorem 1, we have proved that the bottleneck link operates in (b) of (10) when CLTCP flows and Reno flows coexist. In addition, the Reno algorithm uses packets losses to detect the network congestion. This cause $q_c \geq K$ at most cases. According to (1) and (5), $N \to 1$. When $N = 1$, the CLTCP has the same fairness property as TCP Reno. Therefore, the CLTCP is TCP friendly.

As can be seen from the Theorem 2, the CLTCP can ensure inter-fairness with TCP Reno. Moreover, when U CLTCP flows coexist, their values of α are similar. Hence, the CLTCP can ensure intra-fairness.

V. Evaluation and Analysis

To illustrate the performance of the CLTCP, we use a canonical packet-level simulator NS-2 [13], which has been extended with a CLTCP module, to conduct a set of simulations in high-BDP and lossy networking scenarios. The compared schemes include Reno [1], CUBIC [3], CTCP [6], Veno [2], DCTCP [9] and TCP-FIT [8]. Therein, first four schemes have been integrated into NS-2 and their parameter settings are the optimum value recommended by their respective authors. And we implemented TCP-FIT and DCTCP in NS-2 according to [8], [9]. Furthermore, the experimental topology has one bottleneck link and its bandwidth is 100 Mbps. All queues use DropTail to manage packets. Note that the bottleneck queue uses a modified ECN-supported DropTail for CLTCP algorithm and RED for DCTCP algorithm like [9]. The marking threshold, K, is twenty percent of the bottleneck queue limit. No other description, all flows' RTT is 80 ms and the bottleneck queue size equals the bandwidth-delay product, or two packets per-flow, whichever is larger. The data packet size is 1000 bytes, while the ACK packet size is 40 bytes. All flows traverse the same bottleneck link and all simulations are run for at least 100 s unless specified otherwise.

First of all, we use the convergence time (the bottleneck link utilization greater than 0.95) to compare the performance of different TCP variants over the high-BDP network. There are two cases. In the first case, we vary the flows' RTT in [10 ms, 400 ms] and set five flows for different TCP variants. And in the second case, we vary the bottleneck link capacity in [10 Mbps, 200 Mbps] and also set five flows for different TCP variants. Note that the simulation lasts for 300 s and the result obtained from the last 200 s. Other configuration is the default values as previous. As shown in Fig. 2, the convergence time of most TCP variants is approximate except Reno, DCTCP and CUBIC algorithms when flows' RTT and the bottleneck link capacity varied respectively.

Secondly, we compare the performance of different TCP variants over the lossy network. There are also two cases. In the first case, we fix the packet loss rate of the bottleneck link to 1% and five flows for different TCP variants. And in the second case, we fix the packet loss rate to 5% and the number of TCP flows is the same as that in the first case. In the two cases, we vary flows' RTT in [10 ms, 400 ms] and measure the normalized throughput. Other configuration is the default values as previous. As shown in Fig. 3, the CLTCP performs

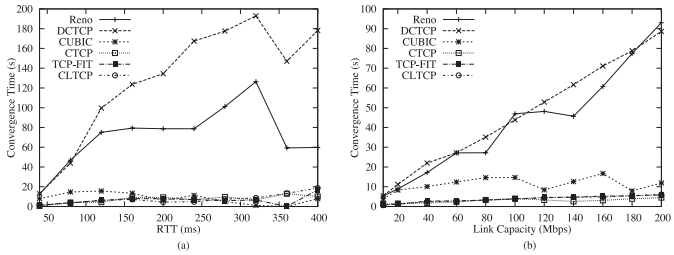

Fig. 2. Performance of different TCP variants over the high-BDP network. (a) RTT variation. (b) Capacity variation.

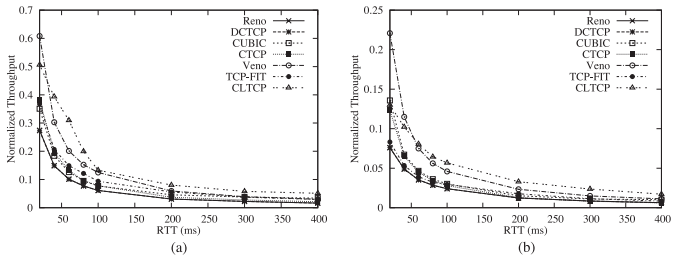

Fig. 3. Performance of different TCP variants over the lossy link. (a) Packet loss rate = 1%. (b) Packet loss rate = 5%.

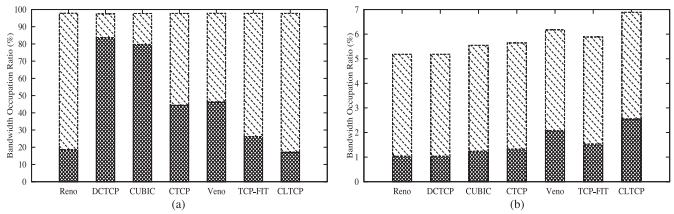

Fig. 4. Bandwidth occupation of TCP variants competing with TCP Reno. (a) Packet loss rate = 0%. (b) Packet loss rate = 1%.

better than other compared TCP variants. These also reflect that the CLTCP can improve the throughput over the lossy network.

Thirdly, we measure the bandwidth occupation of different TCP variants competing with the TCP Reno when the packet loss rate of the bottleneck link is 0% and 1% respectively. For the convenience, we set four Reno flows and one TCP variant flow to compete for the bottleneck capacity. Other configuration is the default values as previous. As shown in Fig. 4(a), the bandwidth occupation of the CLTCP is close to TCP Reno when the packet loss rate is 0%. It also indicates that the CLTCP is more TCP friendly than other TCP variants. Since the bottleneck link has spare capacity when the packet loss rate is 1%, the CLTCP can gain more bandwidth than other TCP variants as shown in Fig. 4(b). Note that the grey grid part and white grid part are the bandwidth occupation of the TCP variants and the TCP Reno respectively.

Last but not the least, we investigate the bandwidth stolen rate (BSR) and Jain's fairness index (JFI) [14] for TCP variants competing with TCP Reno. For measuring the BSR, we set five Reno flows and five TCP variant flows to compete for the bottleneck capacity and vary the packet loss rate of the bottleneck link from 10^{-7} to 10^{-4}. Note that the calculation of the BSR is the same as that in [6]. As shown in Fig. 5(a), the BSR of the CLTCP is smaller. It denotes that the CLTCP

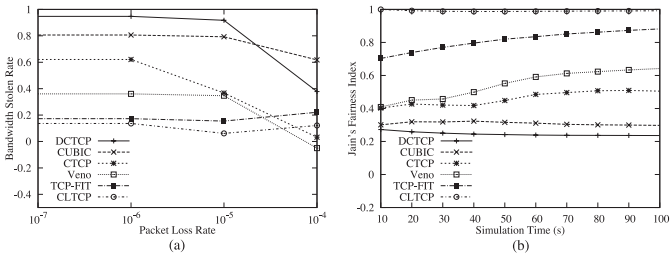

Fig. 5. Fairness of different TCP algorithms competing with TCP Reno. (a) Bandwidth stolen rate. (b) Jain's fairness index.

is more TCP friendly than other TCP variants. For measuring the JFI, we set four Reno flows and one TCP variant flow to compete for the bottleneck capacity. Note that the JFI for five flows is computed every 10 s. As shown in Fig. 5(b), the CLTCP is fairer than other TCP variants.

VI. Conclusion

In this letter, we have presented a novel TCP algorithm for applications in high-BDP and lossy networks. When the packet loss rate varies, the CLTCP can achieve the high utilization and maintain the friendliness with TCP Reno algorithm. Theoretical and experimental results show that the CLTCP can gain significant performance improvements in throughput, fairness or robustness.

References

[1] J. Postel, "Transmission control protocol," Fremont, CA, USA, RFC 793, Sep. 1981. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt

[2] C. Fu and S. C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 2, pp. 216–228, Feb. 2003.

[3] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[4] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock "Host-to-host congestion control for TCP," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 3, pp. 304–342, Jun. 2010.

[5] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: Motivation, architecture, algorithms, performance," in *Proc. IEEE INFOCOM*, 2004, pp. 2490–2501.

[6] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *Proc. IEEE INFOCOM*, 2006, pp. 1–12.

[7] M. Wang, J. Wang, and S. Han, "Adaptive congestion control framework and a simple implementation on high bandwidth-delay product networks," *Comput. Netw.*, vol. 64, pp. 308–321, May 2014.

[8] J. Wang, J. Wen, J. Zhang, and Y. Han "TCP-FIT: An improved TCP congestion control algorithm and its performance," in *Proc. IEEE INFOCOM*, 2011, pp. 2894–2902.

[9] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," in *Proc. ACM SIGCOMM*, 2010, pp. 63–74.

[10] M. Kuhlewind, S. Neuner, and B. Trammell, "On the state of ECN and TCP options on the internet," in *Proc. PAM*, 2013, pp. 135–144.

[11] M. Nabeshima, "Performance evaluation of MulTCP in high-speed wide area networks," *IEICE Trans. Commun.*, vol. 88, no. 1, pp. 392–396, Jan. 2005.

[12] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proc. ACM SIGCOMM*, 1998, pp. 303–314.

[13] "The Network Simulator version 2," 2012. [Online]. Available: http://www.isi.edu/nsnam/ns/

[14] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Comput. Netw. ISDN Syst.*, vol. 17, no. 1, pp. 1–14, Jun. 1989.