CLEAN: Minimize Switch Queue Length via Transparent ECN-proxy in Campus Networks

Xiaojie Huang*, Jiaqing Dong*, Wenzheng Yang*, Chen Tian*
Jun Zhou†, Yi Kai†, Mingjie Cai†, Nai Xia*, Wanchun Dou*, Guihai Chen*
*State Key Laboratory for Novel Software Technology, Nanjing University, China
†Huawei, China

2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS) | 978-1-6654-1494-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/IWQOS52092.2021.9521295

***Abstract*—Campus networks are widely deployed for organizations like universities and large companies. Applications and network-based services require campus networks to guarantee short queue and provide low latency and large bandwidth. However, the widely adopted packet-loss-based congestion control mechanism in client hosts builds up long queues in the switch buffer, which is prone to packet loss in burst scenarios, resulting in great network delay. Therefore, a scheme for efficiently controlling queue length of shallow buffer switches in campus networks is urgently needed. Explicit Congestion Notification(ECN) as an explicit feedback mechanism is widely adopted in data center networks to build lossless networks. In this paper, we propose CLEAN, an efficient queue length control scheme based on transparent ECN-proxy for campus networks. CLEAN is able to exert fine-grained control over arbitrary client TCP stacks by enforcing per-flow congestion control in the access point(AP). It allows the campus network switches to maintain a low queue length, resulting in high throughput, low latency and zero packet loss. Evaluation results demonstrate that CLEAN reduces the maximum queue length of the switch by 86% and reduces the 99th percentile latency by 85%. CLEAN also achieves zero packet loss in burst scenarios.**

I. INTRODUCTION

The campus network has been widely deployed and shows great value in the market. Enterprises and organizations deploy campus network to enhance the internet connectivity, improve the efficiency of enterprise collaboration, and accelerate innovation. Applications and network-based services for enterprises and organizations require the campus network to provide low latency and large bandwidth.

Most of today's widely deployed congestion control algorithms in end host devices, such as NewReno [1], CUBIC [2], etc., are based on packet loss, which is likely to make the switches pile up long queues. Long, greedy TCP flows will cause the length of the bottleneck queue to grow until packet drop happens. In this case, packet loss is easy to happen when bursts occur, which introduces high latency and reduces throughput, affecting the application performance. More importantly, most end host devices in campus networks are generally personal user devices such as mobile phones and notebooks. It is not practical to deploy customized protocol stacks nor do any modifications to the original protocol stacks on these end host devices.

We have two observations here. One is that most commodity switches in campus networks are ECN-capable. The other is that it is relatively easy to do some modifications to the access point devices in a campus network.

The basic idea inspired by the observations to solve the proposed problem is that we modify the AP device and build an ECN proxy at the AP for flows. The ECN proxy is transparent to the end host and is able to let the intermediate switches regard the flow as ECN-enabled and use ECN flags to notify congestion signal instead of simply dropping packets. We reduce the receive window field in the returned ACK to make the sender to slowdown when congestion occurs. Through this mechanism, we can slowdown the traffic in congestion without packet loss.

The challenge is how to calculate the appropriate window size during the lifecycle of a flow. As mentioned above, when congestion happens, we should modify the receive window field in the returned ACK packets. Simply halve the window size or set it to a very small value does slowdown the sender but will also hurt the bandwidth utilization and throughput at the same time.

In this paper, we propose CLEAN, a queue length control scheme based on transparent ECN-proxy for shallow-buffered switches in campus networks. CLEAN takes over the client-side congestion control at the last hop(usually the access point, AP) of the campus network. In CLEAN, the ECN function of the campus network switches are enabled. It makes the switchs regard all flows are ECN-enabled through marking the ECN field of packets as ECT at the entry point of the campus network. We name this mechanism as transparent ECN-proxy. In addition, CLEAN takes the advantage of the traditional TCP window principles, where the actual sending window should be the minimum of the congestion window and the receive window. It forces the end hosts to obey congestion control rules through modifying the receive window field of the returned ACK packets. CLEAN maintains a status table for each flow at the AP, meanwhile, it updates and enforces receive window of each flow according to the network status. With this mechanism, CLEAN is able to let the campus network switches maintain a short queue, which avoids congestion, eliminates packet loss due to buffer overflow, and greatly reduces end-to-end delay. It is worth noting that CLEAN is implemented under the premise that the end hosts do not necessarily support ECN and modification to the end hosts' network protocol stack is not required. CLEAN only requires the network switches to be ECN-enabled which is already satisfied by most commodity

978-0-7381-3207-5/21/$31.00 ©2021 IEEE

978-1-6654-1494-4/21/$31.00 ©2021 IEEE

switches, and do some modifications to the AP devices, which is practical since all these devices are campus-owned.

CLEAN comes with a carefully designed congestion control algorithm, which is able to keep the switches always be in a stable short queue state and guarantees low latency and high throughput.

We use NS3 simulation to evaluate the performance of CLEAN. Evaluation results show that CLEAN can always maintain a short queue for the switches while ensuring throughput. In frequent burst scenarios, the maximum queue length of the switch can be reduced by 86% compared against the original. The short queue length brings an improvement in latency. For the 99th percentile, CLEAN reduces the latency by 85%.

II. BACKGROUND AND CHALLENGES

This section first discusses the packet loss encountered in the production environment of the campus network. Experience tells that packet loss scenarios are widespread, so a solution to the campus network packet loss problem is urgently needed. Some challenges are discussed afterwards.

A. Packet Loss in Campus Networks

A typical campus network topology is illustrated in Fig. 1. The network includes the core layer, the convergence layer, and the access layer. AP devices are connected to the access switches. Promoted by new connection technologies such as 5G and Wi-Fi 6, wireless access has become more and more important in campus networks and campus networks are gradually evolving to an all-wireless network architecture. The swarming effect of mobile users decides sudden high-density access may occur anywhere, indicating that bursts may happen anywhere in the wireless campus.

In such a campus network architecture, the AP as the last hop is generally the bottleneck. And due to the shrinkage of bandwidth between layers, core switches and convergence switches may also become bottlenecks. Becoming a bottleneck means that the switch/AP will become a congestion point, which will cause the accumulation of packets in the buffer, and packet loss will occur when the buffer overflows. Applications and network-based services for enterprises and organizations such as real-time communication, real-time video, real-time collaboration, and real-time project management, require the campus network to provide low latency and large bandwidth. Packet loss will hinder the performance of these applications. Additionally, when the large flow and the short flow coexist, the establishment of the long queue in the switch harms the short flow's demand for low latency, even when no packets are lost.

From the topological structure diagram alone, the access switch does not seem to be the bottleneck and should not experience packet loss. But our production experience tells that even the non-bottleneck access switches suffer packet loss. This is caused by the slow start mechanism of traditional TCP. The windows size will increase exponentially(doubles in every round trip time) in the slow start phase. For downstream traffic(from the cloud to end hosts), if the rate of the last hop

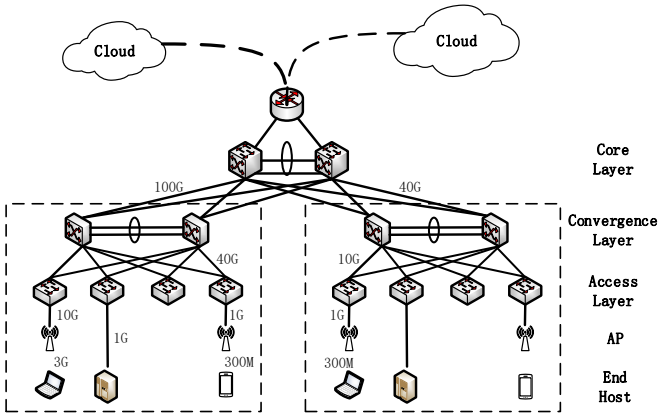


Fig. 1. A typical campus network topology.

AP exceeds half of the bandwidth capability of the up-layer access switch, the downstream packet sending rate driven by the slow start mechanism will exceed the capability of the downstream port of the access switch after one RTT.

Therefore, even if the access switch is not the bottleneck in the topology diagram, packet loss still happens. We further simulated this phenomenon in NS3, and the results verified our experience, as shown in the Fig. 3(the TCP congestion control protocol is configured as NewReno). At the beginning(Time = 0), we start a TCP flow, and add some burst traffic afterwards(Time = $1 \rightarrow 3$). It can be observed from the figure that the packet is lost twice, which are caused by the slow start mechanism of TCP traffic.

B. ECN-based solution and challenges

As an extension of the IP protocol and the TCP protocol, ECN-enabled switches will set a mark in the IP header a packet instead of simply drop it to notify the sender when congestion happens. ECN has been widely used to replace packet-loss based congestion control mechanisms in data center networks, such as DCTCP [3] and DCQCN [4].

In this work, CLEAN also takes advantage of ECN mechanism to solve the proposed problem. However, ECN mechanism cannot be directly applied to campus networks. ECN requires the joint coordination of intermediate network equipments and end hosts. However, different from the data center networks, the end host devices in a campus network are generary personal devices like mobile phones and notebooks belonging to individuals, rather than servers centrally controlled by the manager of a data center network.

In order to solve the problem, CLEAN introduces a transparent ECN-proxy mechanism and uses the receive window to control the sending window of the source (section III). With this specific design, there is still a challenge in order to keep the switch in a short queue state.

CLEAN modifies the receive window field of returned ACKs at the AP to enforce calculated window size. The followed question is, what kind of congestion control algorithm should CLEAN apply to calculate the appropriate receive window size? We can reduce the window very low when we perceive ECN-marked ACKs, which will obviously empty the queue

quickly, ensuring zero packet loss. However this will greatly harm the throughput. On the other hand, when the network is not congested, how to increase the window is also worthy of consideration.

III. Design

This section provides the design of CLEAN.

As mentioned earlier, we had three core requirements for CLEAN: (*i*) No modification to end host devices, because the manager of the campus network is only able to control the intermediate equipments of the network, namely switches or APs. (*ii*) Able to maintain short queue for the switch/AP thus ensuring low latency and zero packet loss even in frequent burst scenarios. (*iii*) High bandwidth utilization.

A. CLEAN *overview*

To meet the design requirements, CLEAN comes with an ECN-proxy based flow control mechanism, which is transparent to the end hosts and requires no modifications to the end host devices. Instead, CLEAN requires the cooperation between APs and the network switches.

Specifically, CLEAN requires the switches in the campus network to be ECN-enabled and then takes over the congestion control at the AP. In order to take over the congestion control, CLEAN does some modifications to the core switch and the AP device. The first one is that CLEAN forcely marks the ECN field of packets as ECT(01 or 10) during the negotiation phase of connection establishment between the host and the server so that the intermediate switches in the campus will regard the client and server as ECN-enabled and enable ECN mechanism for this connection. The second one is that CLEAN maintains a status table recording the window size and packet sequences for each flow at the AP, and updates the window size according to customized congestion control algorithm. CLEAN takes the advantage of the traditional TCP window principle, where the actual sending window should be the minimum of the congestion window and the receive window. It forces the end hosts and servers to obey its window size through modifying the receive window field of the ACK packets.

The intermediate switches will regard all connections as ECN-enabled and will mark the ECN field as CE(11) to indicate that the network is congested when the queue length in the switch exceeds a certain threshold.

With these modifications, CLEAN is able to take control of congestion control and actively respond to ECN notifications sent by intermediate switches.

The workflow is as follows: when traffic enters the campus network, the core switch and AP modifies the ECN field in the IP header of the packet to ECT. When the network is congested, the AP will receive the ECN notifications(packets marked with CE), and then notify the source to slow down through modifying the receive window field of ACKs. After receiving the returned ACK, the source uses the smaller of the receive window value and the congestion window value maintained by the source as its send window size.

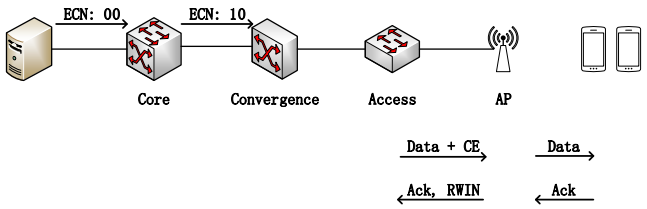


Fig. 2. The overview of CLEAN.

It is worth noting that ECN fields are cleared before a packet being sent back to the sender or receiver, so that the congestion window value maintained by the source will be large and meaningless, and its send window will be dominated by the receive window.

B. Congestion control

1) ECN configuration for the switch: All switches in the campus network enable ECN mechanism through the RED active queue management approach commonly available in modern switches. In the RED queue, there are upper and lower thresholds, and the queuing rule performs probability packet loss according to the length of the queue. In order to be able to react quickly to network congestion, CLEAN only sets one threshold and ECN field will be marked as soon as the queue length exceeds the threshold. Moreover, the RED queue's judgment on the queue length is the result of the Exponentially Weighted Moving Average (EWMA) calculation, taking into account historical factors. CLEAN sets the weight of the historical queue to 0 and set the weight of the current queue to 1, making the queue length equal to the current instantaneous queue, speeding up the network's response to congestion.

2) Congestion control at the AP: CLEAN maintains status for each flow(window size, packet sequence, flags, etc.,) at the AP and runs congestion control algorithm, which also has slow start, window decrease during congestion, and window increase during the recovery detection phase. But the congestion control at the AP will be enforced through receive window and is much simpler than the standard congestion control protocol. It does not need to consider timeouts, dupACK, and other complex states, because CLEAN only needs to ensure that the source can decrease the rate (reduce the window) when congestion occurs. Such congestion control separates the control of the window from the packet loss, and eliminates the drawbacks of the traditional TCP protocol.

Algorithm 1 illustrates the overall process of congestion control logic at the AP for a single flow. At the beginning when a flow starts, the `isCE` flag which refers to the congestion signal sent by ECN-enabled switches is set to $false$, and the `isSlowStart` flag is set to $true$ indicating that the flow enters slow start phase similar to the standard TCP protocol. The `isSlowStart` flag will be set to $false$ as soon as congestion happens and `isCE` becomes $true$, the flow then enters the decrease phase. The param `ecnEchoSeq` indicates the current sequence number of received CE-marked

Algorithm 1 Congestion control at the AP

```
1: if isCE == true then
2:     epochStartTime ← Now;
3:     if ecnEchoSeq>ecnCWRSeq then
4:         W_max ← windowSize;
5:         ecnCWRSeq ← ecnEchoSeq + (windowSize − 1) ∗ MSS;
6:         windowSize ← windowSize/2;
7:         W_min ← windowSize;
8:         isSlowStart ← false;
9:     end if
10: else if isSlowStart == true then
11:     windowSize+ = 1;
12: else
13:     K ← ((W_max − W_min)/C)^(1/3);
14:     t ← Now − epochStartTime;
15:     windowSize ← C ∗ (t − K)^3 + W_max;
16: end if
```

packet and `ecnCWRSeq` refers to the next minimum sequence number that window decrease should happen, ensuring that window size be decreased at most once in a window time, similar to the mechanism in DCTCP [3]. If a packet is not CE-marked, and the flow is not in slow start phase, the flow then enters the fast recovery and active detection phase.

When the network is not congested, it implies that the network may have the remaining bandwidth. There are two stages here, one is fast recovery, that is, the window is restored to the size of the original congestion point, and the other is the active detection stage. For the behavior of how to increase the window, CUBIC is a more mature solution. Therefore, in the window increase part, we use the CUBIC curve to calculate the target value that the receive window should reach. The CUBIC curve's expression is as follows:

$$W = C \times (t - K)^3 + W_{\max}, \tag{1}$$

where $W_{\max}$ is the window size of the last congestion, C is a CUBIC parameter, t is the elapsed time from the last window reduction, and K is the time required to increase W to $W_{\max}$.

When the network does not encounter congestion, it first quickly restores to the window size of the last congestion, and then maintains it for a long time nearby. When there is no congestion for a long time, it implies that the network has a large amount of remaining bandwidth unused, so it quickly detects the upper limit of the bandwidth that can be used.

C. Optimization of window decrease algorithm

From the description of the congestion control algorithm, we can see that the AP roughly reduces the receive window to half of the original when congested, which may cause throughput loss. Therefore, we optimize the window decrease part of the algorithm. Inspired by DCTCP, we calculate the corresponding multiplicative decrease factor α according to the degree of network congestion, use this factor to reduce the window, instead of fixedly reducing it to half, as shown in (2).

$$W = W \times (1 - \alpha/2)\,. \tag{2}$$

DCTCP counts the proportion of CE-marked packets at the receiver. In order not to destroy the Delayed ACK [5] mechanism, DCTCP needs to use a two state state-machine at the receiver. Different from DCTCP, CLEAN directly estimates the degree of network congestion at the AP. The AP observes whether the incoming data packet has CE marking and performs statistics. It has no effect on the Delayed ACK mechanism. The congestion level F is updated once in a window, and the multiplicative subtraction factor is calculated using the EWMA method, as shown in (3). According to the DCTCP recommendation, g is set to $1/16$ in CLEAN.

$$\alpha = (1 - g) \times \alpha + g \times F. \tag{3}$$

We replaced Line 6 of Algorithm 1 with (2). This change is small but very effective. This optimization allows the link bandwidth to be fully utilized. In our simulation tests, although the algorithm can guarantee zero packet loss before and after optimization, the throughput of a single flow is only 820Mbps before optimization (the bottleneck is 850Mbps), and after optimization the bandwidth can be fully utilized.

IV. EVALUATION

A. Setup

We build a campus network topology according to Fig. 1 in NS3. A large server is set to simulate the cloud outside the campus and sends traffic into the campus network. The link bandwidth between the server and the campus network core switch is 100Gbps. The link bandwidth between the core switch and the aggregation switch is 40Gbps, and the link bandwidth between the aggregation switch and the access switch is 10Gbps. There is an AP connected to the access switch with an 1Gbps link. We configure the wireless protocol standard as 802.11ax_5G and configure 2x2 MIMO, which can make the maximum rate of the AP reach 850Mbps. Both the core switch and the aggregation switch have a shared buffer of 12MB. The buffer size of the access switch is set to 4MB, while the AP has a larger buffer of 12MB. The topology is setup according to our real-world campus network experience and is supposed to simulate the real campus network architecture to the greatest extent.

As CUBIC is widely deployed on Windows and Linux machines, we configure the TCP congestion control protocol as TCP-CUBIC in NS3 in order to show the packet loss situation encountered by the campus network under the traditional WAN TCP protocol. We also repeat the experiments with TCP-NewReno for comparison.

The base RTT is set to 10ms, which is the mainstream scenario for WAN traffic in campus networks. The ECN function of the switch is enabled and the threshold is set 150 packets according to the DCTCP recommendation [3].

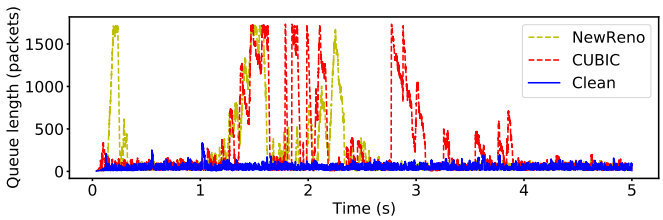


Fig. 3. The queue length of the access switch at the non-bottleneck.

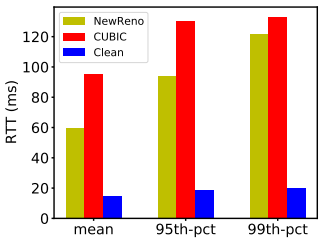


Fig. 4. RTT.

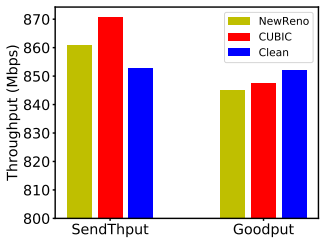
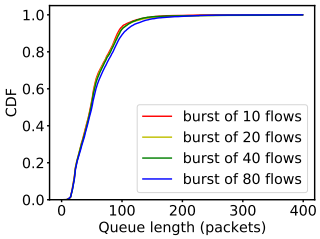
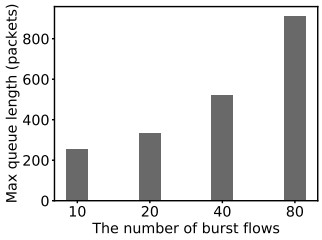


Fig. 5. The throughput of the sender and Goodput.



(a) CDF of queue length



(b) max queue length

Fig. 6. The queue length in the 10, 20, 40, and 80 burst flows scenario.

B. Performance

1) Queue length and throughput: We first start a TCP flow, and then add 20 burst flows to test the buffer pressure of the switch in the burst scenario. As shown in the Fig. 3, when CLEAN is not applied, due to the greedy feature of TCP, the flow quickly occupies the buffer of the switch, and a queue is established in the switch, which is prone to packet loss. After applying CLEAN, we can see that the length of the queue is always maintained near the ECN threshold we set. This not only can significantly reduce the RTT of the traffic, but also leaves a lot of buffer space to deal with bursts.

It is worth noting that even if the congestion control protocol is both based on packet loss, the queue length curve of CUBIC is different from that of NewReno. When the TCP flow starts, NewReno will lose packets due to slow start, but CUBIC will not. This is because Hystart [6] is enabled in the newer CUBIC version, and slow start is exited when continuous ack is received quickly. Nevertheless, CUBIC still cannot cope with packet loss in burst scenarios. It can be observed from the figure that after burst traffic enters, the switch buffer is quickly filled up and packet loss occurs.

As shown in the Fig. 4, in CLEAN, since both the switch and the AP maintain a short queue length, the queuing time is very small, and the RTT is close to the propagation time plus the transmission time. As NewReno or CUBIC are both congestion control protocols based on packet loss and packets will occupy all available buffers. This causes their queuing time to be several times or even ten times that of the base RTT, which harms their real RTT.

According to [7], the algorithm based on packet loss has such a conclusion: if the number of buffers exceeds the BDP of the connection, then the periodic packet loss caused by buffer overflows does not result in a reduction in TCP throughput. Therefore, as shown in the Fig. 5, the sender throughput of NewReno and CUBIC is slightly larger than the bottleneck bandwidth, but due to packet loss, their Goodput will be smaller than the bottleneck bandwidth. In contrast, CLEAN is basically consistent with the bottleneck bandwidth in terms of both the sender throughput and goodput, indicating a high bandwidth utilization.

In addition to the 20 burst flows scenario, we also tested the queue length of the switch in the 10, 40, and 80 burst flows scenario. As shown in the Fig. 6, after applying CLEAN, the switch can maintain the queue length near the threshold in any scenario. Fig. 6 (b) also shows the maximum queue length generated at the burst point in the switch. The results demonstrates that with CLEAN, the established queue can be emptied quickly.

2) Convergence: To test whether CLEAN can quickly converge to a fair share, we start a single flow at the beginning, and then add a flow every 5s. The results are shown in the Fig. 7. It can be observed from the figure that each joined flow can converge to their fair share in a short period of time, which shows that CLEAN performs well in terms of convergence.

V. RELATED WORK

The congestion control protocol of the WAN is developing very rapidly, from the earlier Reno [8] to NewReno [1], High-speed TCP [9], BIC [10] and CUBIC [2]. These protocols work hard to improve TCP performance on paths with high BDP. However, as congestion control protocols based on packet loss, they do not care about the length of the queue, so that they bring high throughput while also bringing high delay. BBR [11] does not take the occurrence of packet loss or delay increase as a signal of congestion. Alternatively, it periodically detects the capacity of the network. As a result, it has a better control over the queue length. However, we cannot require all senders and receivers in the campus network to use BBR. In addition, if the switch buffer is large, the bandwidth occupied by BBR will be smaller when competing with more aggressive protocols(e.g. CUBIC).

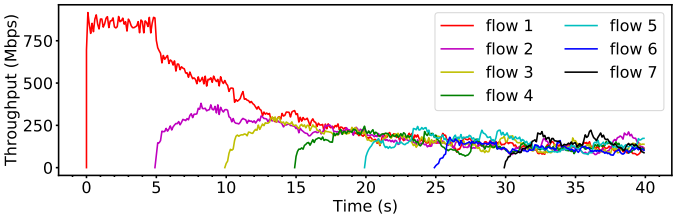


Fig. 7. Time series of throughput.

In order to control the queue length better, some schemes based on explicit feedback are proposed, such as ECN [12], VCP [13], RCP [14], etc. Schemes like VCP and RCP require switches to perform computation which is not widely available in most commodity switches. Therefore, ECN is widely used, especially in the data center networks, such as DCTCP [3], D2TCP [15], DCQCN [4] and some other strategies [16]–[18]. They show that ECN can effectively control the queue length to achieve low latency and high throughput simultaneously. HPCC [19] uses the INT feature to implement high precision congestion control. For these protocols, whether using ECN or INT, they all need the joint support of network intermediate equipments and end hosts, which is always a major challenge for campus networks.

To solve this problem, we used the receive window to control the end hosts. The earlier literature related to this area has [20]–[22]. They modify the receive window in TCP acknowledgments returning to the source. These methods are relatively crude and cannot maintain the switch queue at a stable length. If the topology has multiple bottlenecks, each switch needs to perform similar calculations, which is expensive. They have also been found to be vulnerable to packet loss and poor compatibility with some TCP sending operating systems. The recent literature related to the use of the receive window is VCC [23], [24], which decreases the receive window to force the guest to have fewer send packets in the hypervisor. However, VCC is used in multi-tenant data centers, and it is not suitable for campus network environments. And more importantly, it does not mention how to calculate the appropriate window size.

VI. Conclusion

In this paper, we propose CLEAN, a queue length control scheme via transparent ECN-proxy for campus networks. CLEAN takes over the congestion control for end host devices transparently through modifying the ECN and receive window field at the last hop of the campus network. Evaluation results show that CLEAN can achieve zero packet loss without loss of throughput, which greatly reduces latency. For the 99th percentile, CLEAN reduces the latency by 85%.

Acknowledgment

The authors would like to thank anonymous reviewers for their valuable comments. This research is supported by the Key-Area Research and Development Program of Guangdong Province 2020B0101390001, the National Natural Science Foundation of China under Grant Numbers 61772265, 61802172, and 62072228, the Fundamental Research Funds for the Central Universities, the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Jiangsu Innovation and Entrepreneurship (Shuangchuang) Program. Jiaqing Dong and Chen Tian are co-corresponding authors.

References

[1] S. Floyd, "RFC 6582," https://tools.ietf.org/html/rfc6582.

[2] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[3] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *Proceedings of the ACM SIGCOMM 2010 conference*, 2010, pp. 63–74.

[4] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale RDMA deployments," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 523–536, 2015.

[5] R. Braden, "RFC 1122," https://tools.ietf.org/html/rfc1122.

[6] S. Ha and I. Rhee, "Taming the elephants: New TCP slow start," *Computer Networks*, vol. 55, no. 9, pp. 2092–2110, 2011.

[7] S. Varma, *Internet congestion control*. Morgan Kaufmann, 2015.

[8] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM computer communication review*, vol. 18, no. 4, pp. 314–329, 1988.

[9] S. Floyd, "RFC 3649: HighSpeed TCP for large congestion windows," https://tools.ietf.org/html/rfc3649.

[10] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in *IEEE INFOCOM 2004*, vol. 4. IEEE, 2004, pp. 2514–2524.

[11] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[12] K. Ramakrishnan, S. Floyd, and D. Black, "RFC 3168: The addition of explicit congestion notification (ECN) to IP," https://tools.ietf.org/html/rfc3168.

[13] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One more bit is enough," in *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, 2005, pp. 37–48.

[14] N. Dukkipati, *Rate Control Protocol (RCP): Congestion control to make flows complete quickly*. Citeseer, 2008.

[15] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 115–126, 2012.

[16] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan, "Minimizing flow completion times in data centers," in *2013 Proceedings IEEE INFOCOM*. IEEE, 2013, pp. 2157–2165.

[17] D. Shan, W. Jiang, and F. Ren, "Absorbing micro-burst traffic by enhancing dynamic threshold policy of data center switches," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 118–126.

[18] D. Shan and F. Ren, "Improving ECN marking scheme with micro-burst traffic in data center networks," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[19] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh *et al.*, "HPCC: High precision congestion control," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 44–58.

[20] L. Kalampoukas, A. Varma, and K. Ramakrishnan, "Explicit window adaptation: A method to enhance TCP performance," in *Proceedings. IEEE INFOCOM'98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No. 98*, vol. 1. IEEE, 1998, pp. 242–251.

[21] S. Karandikar, S. Kalyanaraman, P. Bagal, and B. Packer, "TCP rate control," *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 1, pp. 45–58, 2000.

[22] H.-Y. Wei, S.-C. Tsao, and Y.-D. Lin, "Assessing and improving TCP rate shaping over edge gateways," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 259–275, 2004.

[23] K. He, E. Rozner, K. Agarwal, Y. J. Gu, W. Felter, J. Carter, and A. Akella, "ACDC TCP: Virtual congestion control enforcement for datacenter networks," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 244–257.

[24] B. Cronkite-Ratcliff, A. Bergman, S. Vargaftik, M. Ravi, N. McKeown, I. Abraham, and I. Keslassy, "Virtualized congestion control," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 230–243.