International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-2, July 2019

A Hybrid Framework for TCP Incast Congestion Control in Data Center Networks

K. Arun Selvi, K. Kumar, K. Ramalakshmi, A. Sathiya

Abstract*: Cloud data centers with large bandwidth and low latency networks experiences many-to-one traffic pattern called TCP-incast. It occurs in the partition-aggregate architecture and causes emergence congestion at the network wharfconnected to the parent server, overcoming the port emergence buffer. The ending packet loss requires nodes to encounter loss, retransmit data and slowly rise up throughput per definitive TCP behavior. This paper proposes Receiver-oriented Congestion Control with Edge computing approach (RCCE) for enhancing the speed, nature and firmness of traffic performance. Receiver-oriented Congestion Control (RCC) combines both closed and open loop congestion controls at receiverwhereas edge computing involves localization of traffic management in the middle-tier aggregator for reducing Flow Completion Times (FCT) and latency for the entire application processing deployments. In addition, the centralized controller at the edge balances the load during incast by using spanning trees in a well-made manner by implementing multi-stage Clos networks. The entire prototype is implemented in ns3 and simulation results demonstrates that RCCE has an average decrease of 60.2 % in the 99th percentage latency and 50.4 % of mean queue size in the heavy traffic over TCP.

Index Terms: Edge Computing, Data Center Networks, incast, Receiver-oriented Congestion Control, TCP.

I. INTRODUCTION

Data Center Network (DCN) is a communication network which interconnects pool of resources namely computational, storage and network in a data center [1]. DCN runs strongly linked computing tasks to be responsible to user environment, e.g., lots of backend computers (servers) may transfer data to process the client's request and all of the transfers and complete response to the user must be completed with the time limit of 100 ms [2]. In order to satisfy these criteria, DCN offers larger bandwidth and low latency environment since a small delay in transmission can cause degradation in application performance [3].The emergence of applications related to cloud in systems such as Google G Suite, Adobe Creative Cloud, Dropbox, Github and Wordpressattracts an increasing number of users and extent its positioning on DCN [4]. Therefore the trend of DCNs is inevitable. Many cloud applications in DCN generates larger number of traffic that ranges from barrier-synchronized, short-lived flows such as web searches **(mice)** to bandwidth-inclined, long-lived flows such as virtual machine migration and backups **(elephants)**. Recent studies showed that DCNs are often crowded with mice, if occurrence in huge volume renamed to elephants [5].

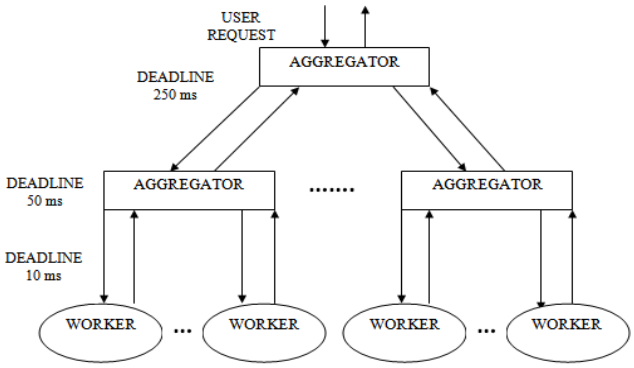


Fig. 1. Partition-aggregate architecture

The DCNs discussed here are wired environments only. GenerallyDCN uses Ethernet switches with limitedbuffers for server's interconnection. The incast many-to-one traffic pattern exposes TCP challenge in the presence of this limited buffer. It occurs on the partition-aggregator architecture as shown in Fig. 1 where many barrier-synchronized client nodes transmit data to the parent nodes. This scenario will easily make room to the **TCP incast congestion problem** in data centers. During this problem, traffic flows experiences heavy packet loss and larger FCTs which in turn leads to industrial capital loss and bad quality of service (QoS) [6,7]. In order to mitigate this problem, effective congestion control frameworks such as sender, receiver, switch-assisted and SDN-based had been proposed to enhance data flows in DCNs [8]. Since incast congestion occurs mostly at the last hop of the switch, congestion control at receiver faces heavy incast in scaling up of data centers. It provides a mean decrease of 51.2% in the 99th percentage latency and 47.5% of mean queue size and in the heavy traffic over TCP [9]. But many researchers feel that a local storage and delivery provider of bandwidth-intensive content as part of a content distribution network can relieve network from congestion and improves streaming of high bandwidth data now and in the future [10].

Revised Manuscript Received on 30 July 2019.
***** Correspondence Author
Ms. K. Arun Selvi*, Assistant Professor Department, of Computer Science and Engineering National Engineering College, Kovilpatti-628 503, Tamil Nadu, India
Dr. K. Kumar, Assistant Professor Department, of Computer Science and Engineering National Engineering College, Kovilpatti-628 503, Tamil Nadu, India
Ms.K. Ramalakshmi Assistant Professor Department, of Computer Science and Engineering National Engineering College, Kovilpatti-628 503, Tamil Nadu, India
Ms. A.Sathiya Assistant Professor Department, of Computer Science and Engineering National Engineering College, Kovilpatti-628 503, Tamil Nadu, India

Therefore,service providers are interconnecting a system of computers on the Internet that caches the datacloser to the user. This makesthe datato be deployed rapidly to multiple users by duplicating the content on different servers and directing the content to users based on proximity. These computers caching content are called edge computers. It gathers information automatically about nodes to monitor traffic for system deployments. Using this information, large scale DCNs can satisfy requirements for traffic management [11].

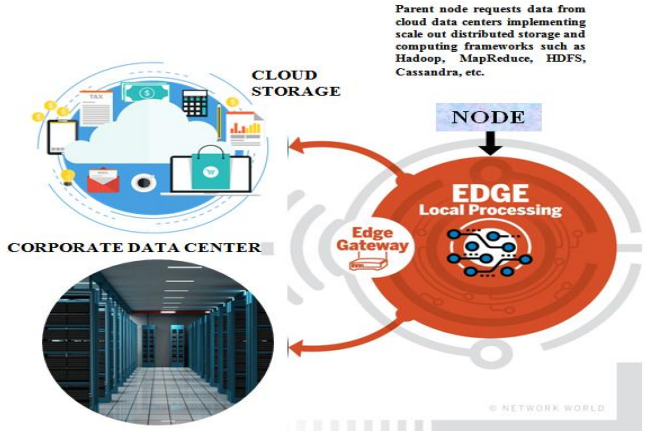


Fig 2. Hybrid Framework of RCC with edge computing

This paper designs and implements a hybrid frameworkas in Fig. 2 which encapsulates receiver-oriented congestion control with edge computing approach for applications having many-to-one traffic pattern. Firstly, the risks of TCP protocols causing incasttaffic in DCNs were identified and factors which cause transport deficiency and performance collapse were provided. Secondly, RCCE in configure-to-order systems (edge systems) in the middle-tier aggregator for reducing latency in the high bandwidth environments were employed. RCC is an integration of open and closed loop congestion controls at receiver and based on TCP. Open loop makes use of centralized scheduler for suppressing the burstiness of incast and employs multi-stage clos networks for balancing the network using spanning trees whereas closed loop makes use of Explicit Congestion Notification (ECN) for achieving normal congestion control in DCNs. The summary of work is as proceeds. Section II deals with research literature works related to receiver-based incast congestion control. Section III deals with motivation of edge in congestion control for DCNs. Section IV reveals prototype of RCCE. Section V deals with implementaion analysis of RCCE factors. Section VI further evaluates performance of RCCE in ns3 simulation environment. Finally section VII concludes overall aspects of the paper.

II. RELATED WORKS

Over the past few years, researchers of network have introduced different effective protocol frameworks based on TCP for its effective usage of bandwidth. The following present typical TCP-related protocols for receiver-oriented congestion control in DCN in brief.

Haitao Wu et al., proposed **"ICTCP: Incast Congestion Control for TCP in Data Center Networks"** [12]**.** It is a window based congestion control algorithm of receiver. The entire reduced Round Trip Time TCP sessions are collectively fine-tuned to stabilisethroughput during incasttraffic. The adjustment depends on the ratioof deviation of expected and obtainedper connection throughputs over conventional ones, as well as the last-hop bandwidth to the receiver. It improves the TCP performance for incast in DCNs.

V. Tsaoussidis et al., proposed **"TCP-Real: receiver-oriented congestion control"** [13]. It uses "wave" communication pattern which allows for congestion avoidance for lowering of unnecessary transmission gaps and advanced error detection and classification mechanisms for providing response to the nature of the errors. The level of contention monitoring allows the receiver to take over on the emergence of packet drops.

Wei Bai et al., proposed **"PAC: Taming TCP Incast Congestion using Proactive ACK Control"** [14]. PAC treats ACK as both the provoke for new packets and as an acknowledgement of received packets and as a. It passes ACKs in such a way that the ACK-provoked incoming data can entirely use the overwhelming link without making incasttraffic collapseeventhe senders are above thousands. It is the first protocol to prove that the receiver can avoidincast traffic.

Lei Xu et al., proposed **"Enhancing TCP Incast Congestion Control Over Large-scale Data Center Networks"** [15]. RDTCP holds bothclosed and open loop congestion controls at receiver i.e., centralized scheduler for suppressing incast and ECN for achieving normal congestion control in data centers. After connection establishment, rwnd i.e.,size of receiver windowwill be simultaneously passed to sender via TCP header advertisement window and the sender fixrwndas its definite congestion window for transmitting data. Evaluation results proved that RDTCP achieves the minimum 99th percentage FCTs.

Lei Xu et al., proposed **"Throughput Optimization of TCP incast congestion control in large-scale data center networks"** [9].This approach designs and implements a transport protocol for heavy incastin congestion control to satisfy criteria from large-scale DCNs. RCC also integrates both closed and open loop congestion controls at receiver. It is implemented in ns3 and the simulation results shows that this approach has an **average reduction of 47.5% of mean queue size and 51.2% in the 99th percentage latency in the heavy traffic over TCP**. But the above discussed protocols do not impose much effect in FCTs and persistent transportation of data. Hence edge takes place in these areas to reduce transport time and promotes effective streaming of packets.

Keqiang He et al., proposed **"Presto: Edge-based Load Balancing for Fast Datacenter Networks"** [16].Presto makes load-optimising functionbe pushed into the soft networking edge e.g., virtual switches such that no modification is required in the transport layer, client VMs or any specific networking hardware.

It balances the network at good granularity and deal with re-positioning without showing much overhead on nodes. But this protocol deals with normal congestion only, hence the hybrid framework of RCC with Edge has been proposed for TCP incast congestion.

III. MOTIVATION OF RCC WITH EDGE COMPUTING

A. *Decentralized cloud and low latency computing*

Centralized data centers does not fit best for bandwidth intensive applications that are distributedgeographically. Computation and data transmission in the cloud or data centers has to perform closer to the origin of the receiver for reducing packet drops and ECN markings [17, 18]. This benefit is generalized for DCNs which implements large scale distributed storage environments and frameworks for computing such as HDFS, Cassandra, Hadoop, MapReduce empowering applications such as web searching, google maps, social networking, data analytics and warehousing as it greatly increases FCTs during heavy incast [19]. Since edge makes the computing environment closer to the receiver, data is decentralized and latency gets reduced during heavy incast even in the 99th percentage over TCP [20].

B. *Support heavy traffic at the edge*

The cloud computing emergence as an efficient means providing computational analysis can already be satisfied with the increase of cloud-based applications. Consequently, the quantity of data generation will also increase, it is estimated that 43 trillion GB of data will be generated in 2020 [21]. This scenario pushes the need for elaborating DCNs to support visualization and analytical workloads. Major consideration with data generation is the amount of network traffic to a centralized data center or cloud. Since edge can automatically gathers information about bandwidth-intensive applications for optimizing traffic environments, it suppresses heavy traffic and decreases the response time of networking devices [22, 23, 24]. This reveals the possibility of data centers to cope up with the data growth as well as traffic distribution in the network.

C. *Integration of multiple senders*

In a partition aggregate workflow, an aggregator node divides computation across multiple worker nodes. Worker nodes perform the computation in parallel and send results to the middle tier aggregators. Edge can take place of these aggregators and performs computation at the nearby locality. Then the aggregators at the edge can combine the results to provide a complete response back to the top-tier aggregator [25]. Generally DCN chooses the server or VM for aggregation that has a large computational and storage capacity. Edge performs aggregation of bandwidth intensive applications like web search and data ranking in a nearby datacenterfor avoiding the heavy incastduring data transmission.Therefore multiple senders integration is done in the edge itself and no long senders can communicate with the receiver at a time. Simultaneously the downstream workers must also immediately communicate with the edge, which requires the DCN topology to reduce their hop distances to be most efficient [26].

IV. DESIGN OF RCCE

RCCE is implemented at the receiver side and the mechanism is as follows. Initially, Edge as a mesh network of micro data centers automatically triages the bandwidth-intensive and latency-sensitive applications locally. During request from the receiver, multiple senders directly from the corporate data center need not to be aggregate with the receiver. Instead, Edge performs transportation of data traffic from the nearby localized form factor. Then, the centralized scheduler at the edge consists of buffer queue, window assignment and load-balancing performs action based on congestion. Maximum window size is allocated by Window assignment for all the flows, load-balancing balances traffic using spanning trees in a near-optimal fashion and queuing bufferconsumes bursting part and calculates available count of windows at flight from the edge area. During data transmission, ECN works at the edge andrwnd will be simultaneously sent off to sender and hereby sender holds rwnd as its definitive congestion window.

A. *Window assignment and load balancing*

RCCE uses window assignment algorithm for assigning suitable window to all the flows when heavy flows are transmitted together in to the NIC of same receiver. The main motivation is to partition the queue_max for all the senders during packet transmission. The load balancing approach uses a complementary mechanism to suppress congestion when connections are too large. In this algorithm, dat is an array which stores all the data flow information and this gets updated when the SYN and FIN i.e., Synchronousand Finish packet is received by the receiver and flow_priorimplies whether the flow priority is used or not. If the packet has flow size information, RCCE uses flow size as priority otherwise it is set to zero. If all the flows has same priority (0 or unavailable), buffer queue degenerates First in First out (FIFO) queue for processing of packets. In line 1 of algorithm, ascend sort flows from higher level priority to lower level and calculates flow's capacity by

$$w_i = \frac{p_i}{\sum_j^n p_j}$$

(1) wherep_i is priority of flow iand n refers to the total number of connections. Further, the first loop of while assigns congestion windowsfor all data flows and the second loop of while attempts to usethe unused bandwidth by the first while loop.When connections are too large, RCCE with the centralized scheduler at the edge can balances the load in the network in an equal manner, excluding an individual switch havingfulfilled data about structure of the network, updated traffic information or uniform cooperation among senders for congestion management and balances the connection at a fine granularity to operate at a high speed. At the upper level, controller part splits the entire network into a pack of various spanning trees. Afterwards, the controller will assign every switch a succeeding label in every tree.

With the switches splitting traffic among the above spanning trees in a well-defined fashion, the network can balance the traffic load in an excellent manner. Creation of spanning treesis possible by putting Clos networks in multi-stagegenerally placed in data centers.

Algorithm 1: Window Assignment for All Flows

Requirement: dat / /an array holds all the flow information flow_prior //A flag for flow priority;

1. Ascend(dat, flow_priority) ; i = 0 ; sum_win = 0 ;
2. ***while****i<dat.length () **do***
3. dat[i].win = dat[i].weight ∗queue_max;
4. ***if****dat[i].win< 1 then dat[i].win =1 ;*
5. ***else if****flow_prior and dat[i].win>dat[i].flowsize**then***
6. dat [i].win = dat[i].flowsize ;
*7.**end if***
8. sum_win + = dat[i+ +].win ;
*9. **end while***
10. i = dat.length () − 1; N_noneed_check = 0 ;
*11. **while** flow_ prior and sum_win<queue_max**do***
*12.**if** | dat[i].win – dat[i].flowsize |< = 1 **then***
*13. **if** + + N_no_check> = dat.length() **then** break;*
*14. **end if***
*15. **else**dat[i].win + + ; sum_win + + ;*
*16.**end if***
*17. **if** −−i==−1 then i = dat.length() − 1; N_no_check =0;*
*18.**end if***
*19. **end while***

B. Buffer queuing mechanism

To reduce packet loss, RCCE uses buffer queuing which is as follows. The mechanism confirms the queue_maxsize is above the sum of flight windows. Since it is a receiver-based congestion control, RCCE at receiver part uses counter for noticing the number of flight windows. If any data packet which is newcomes near receiver side, packet size gets decreased by counter and if that corresponding ACK arrives back, counter gets incremented. Initially the centralized scheduler computes the flight_winsand sum of all the flows within the same NIC of receiver. If the number of flight windows becomes higher than queue_max, buffer queuing will store ACK packets without sending it. In addition, to avoid retransmission of packets, buffer at queue use quick sort for sorting ACKs. The higher priority packets are sorted near queue head. The buffer queue algorithm is implemented when an ACK is received by the receiver during congestion. Line 1 of algorithm assures unique ACK in the queue and returns skiptime (time denotes an ACK is sipped over). Line 2 inserts ACKs into the queue by considering flow_ prior. If the ACKs do not have flow_prior, buffer queue will degenerate FIFO.

Algorithm 2: Sending Back ACKs via Buffer Queue

Requirement: flags //The packet flags to be written in TCP packet header;

1. flow_identity //The identifier of this flow ;
2. flow_priority //The priority of this flow ;
3. skiptime //Times that current ACK is skipped over;
4. skiptime = queue.erase (flow _ identity) ;
5. queue.insert (flow_identity, flow_priority, flags, skiptime) ;
*6. **while**flight_wins ≤queue _ max do*
7. Send (queue.pop()) ;
8. Update (flight _ wins) ;
*9. **end while***

For reducing the waiting period of lower priority flows, the ACK's skiptime is added by 1 during skipping by a larger priority ACKs. The maximum skiptime is $C * \frac{RTO^r}{2}$ here C is the total bandwidth in upcoming packet per second, RTO^r is Retransmission Time Out in receiver side per second and 1/2 denoteshigher priority expectation existence. The while loop sends ACKs if the switch is idle.

C. Delayed ACK mechanism and retransmission

In general, RCCE makes receiver part disable the ACK mechanism which is delayed (i.e., transmitting ACK for each corresponding data packet). Further, it also considers ACK receiver's retransmission and uses $RTO^r$for clocking ACKs. If no packets or corresponding ACKs are received in the buffer queue, RTO is scheduled for firingwith a time sequence of RTO^r. For faster retransmission of packets from sender part, in the TCP header, receiver adds the PSH flag.

Algorithm 3: Retransmission at Receiver part

Requirement: flag //The flag of TCP header

1. fast recovery flag ← 0
2. duplicated Ack ← 0
3. ssthresh ← max {2∗segment, rwnd/ 2 }
4. rwnd ← segment
5. delayAckCount ← delayAckMax −1
6. RTO^r ← RTO^r ∗2
7. SendACK(flag| PSH)

D. ECN response

The ECN mechanism is used by RCCE for detecting the extent of remote congestion. It is enabled when queue length exceeds threshold K. At this time, the switches can mark all the data packets and set the Congestion Experienced (CE) codepoint for avoiding packet losses. RCCE can calculate theextent of congestion by finding α_r, the ratio of data packets having marked and total packets over RTT. Further, it reduces window size with respect to α_r as

$$rwnd = rwnd * (1 - \propto_r/2) \quad (2)$$

Dual state machine is not needed by the receiver to determine setting of ECN-Echo bits because the receiver can directly obtain α_r and fits congestion window accordingly.

V. PARAMETER ANALYSIS OF RCCE

This section analyses the parameters of centralized scheduler, ECN mechanism and FCT for RCCE. The procedure for analysis is based on RCC [9].

A. Analysis of centralized scheduler

Consider N long-lived flows from the edge passes into the receiver with ECN disabled. The threshold is queue_max in unit of packet, link capacity is C packet per second and average RTT without any queue backlog is RTT_a. Then the output queue_sizeof the switch at its last hop is

$Q_1 = queue\_max - RTT_a \times C$ (3)The average RTT at the stable state is

$Q_1/C + RTT^a$(4) If queue_max is larger than N, the size of buffer queue is zero else the queue size is

$Q_2 = N - queue\_max$ (5)

Q_2 takes ACK packet as unit. Since the output behavior of buffer queue is inspired by the input of received data packet, queue delay is still Q_2 / C. Then, the ideal maximum buffer queue size is

$Q_{2\_ideal} = RTO^r \times C$ (6)When N is large and if the buffer queue size is greater than Q_2*_ideal*, spurious retransmission at receiver is induced and is unavoidable. This retransmission has greater impact on RCCE, because the retransmitted ACK is removed in the buffer queue by the centralized scheduler. In Fig. 3 (a), the Q_1 is around 50 packets which is in line with (2) and Q_2 is in line with (4).

B. Analysis of ECN

This section focuses on ECN with the amplitude of queue oscillations (A_r). Consider N long–lived flows which always occur in data centers and in wide area networks. All the flows have same RTT, sharing a single bottleneck link of capacity C gets synchronized with the centralized scheduler disabled. If window size has been increased from *W_1 to W_2 > W_1*, then the number of packets sent by the sender is *P* (*W*_1, *W*_2). The average window size with round-trips *W_2-W_1* is (*W_1 + W_2)/2*,

$P(W\_1, W\_2) = (W\_2^2 - W\_1^2)/2$(7)

The window size *W_k of* each flow if the switch queue size reaches *K* will be

$(C * RTT + K)/N$ (8)After *RTT/2*, the receiver reacts to the ECN marks, during which its window size is increased by 1/2 packet, reaching *W_k+1/2*. Here, the fraction of marked packets $\alpha_r$is

$$\frac{P(W_k, W_k+\frac{1}{2})}{P\left(\left(W_k+\frac{1}{2}\right)\left(1-\frac{\propto_r}{2}\right), W_k+\frac{1}{2}\right)}$$

(9)

Substituting (7) in (9) and assume W_ *k >> 1*,

$$\propto_r^2 \left(1 - \frac{\propto_r}{4}\right) = \frac{W_k + 1/4}{(W_k + \frac{1}{2})^2} \approx \frac{1}{W_k} \quad (10)$$

If $\alpha_r$is small,

$$\propto_r \approx \sqrt{\frac{1}{W\_k}} \quad (11)$$

Therefore, the oscillation amplitude in window size of a single flow *D* is

$D = (W\_k + 1/2) \propto_r /2$(12)

There are *N* flows in total, hence

$A_r = ND = N(W_k + 1/2) \propto_r /2$ (13)

Plugging the value of Equations (7) and (9) and *W_k>> 1*,

$A_r = 1/2\sqrt{N(C \times RTT + K)}$ (14)

The period of oscillations *(Tc)* is calculated as,

$T_C = \mathrm{D} = 1/2\sqrt{(C \times RTT + K)N}\ (in\ RTTs)$

(15)

The maximum size of the queue (Q_{max}) is

$Q_{max} = N\left(W_k + \frac{1}{2}\right) - C * RTT = K + N/2$ (16)

Since $Q_{max} = Q_{max} - A_r$

$= K + \frac{N}{2} - 1/2\sqrt{N(C \times RTT + K)}$

(17)To find a lower bound on K, minimize (17) over N and choose K so that the minimum value is larger than zero. Then we get

$K > (C * RTT)/7$

(18)

Equation (12) reveals that if *N* is small, the queue oscillation amplitude of RCC is in *O(√ C ×RTT)*. In Fig. 3 (a), $A_r$is above 4 packets and $T_c$is around 1 RTT, both of which are in line with (12) and (13).

C. Analysis of RCC near-optimal FCT in incast

For analysis of RCCE near-optimal FCT in heavy incast, we assume that all the hosts are under the same switch sending same flow size flows to one receiver and forms a many-to–one traffic pattern. For simplicity in transmission, the link does not contain any delay and the near-Optimal incast TCP protocol (OITCP) has all flow information in advance. Consider there are N flows of size L in bytes. Let the receiver link capacity is C packet per second, switch queue size is Q in packet, switch queue sizes for OITCP and RCCE is Q_0 and queue_maxsize respectively. OITCP averagely maintains a switch queue size of Q_0, transmit window of arbitrary size and partition Q_0*segment among flows.

Similar to large-scale incast, suppose

$queue\_max \le N \le Q_o$ (19)

so that OITCP dataflows will obtain a window size larger than one byte $Q_o * segment/N$ and RCCE data flows will obtain a one segment window size. Hence, an OITCP flow will transmit

$$\frac{L}{Q_o * \frac{segment}{N}} = NL/(Q_o * segment) \quad (20)$$

packets. Further, it experiences a switch queue delay Q_o/C and transmission delay $L/(segment$ ∗C). Therefore its FCT is

$$\left(\frac{NL}{Q_o * segment}\right) * \left(\frac{Q_o}{C}\right) + \frac{L}{(segment * C)} = NL + \frac{L}{segment} * C$$

(21)

In RCCE, data flow will send$L/segment$ data packets and experiencesdelaysin two queue and a delay in transmission$L/(segment * C)$. Thus, its FCT is

$$\left(\frac{NL}{Q_o * segment}\right) * \left(\frac{Q_o}{C}\right) + \frac{L}{(segment * C)} = NL + \frac{L}{segment} * C \quad (22)$$

This is equal to the perfectly obtained FCT in (21). This conforms that in a stabilized environment, RCC is optimal.

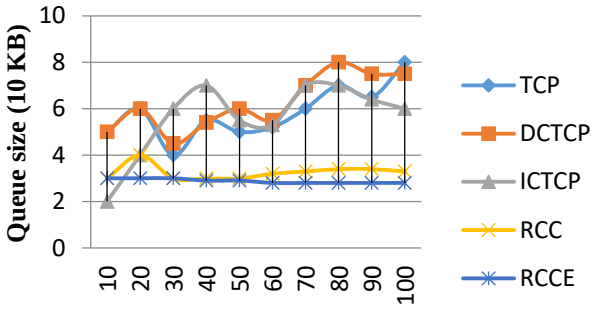
VI. PERFORMANCE EVALUATION OF RCCE

RCCE is evaluated with the factors of goodput, queue buildup, packet-drops, ECN markings and flow completion time. In ns3, transport mechanism for RCCE is implemented with the following criterion.

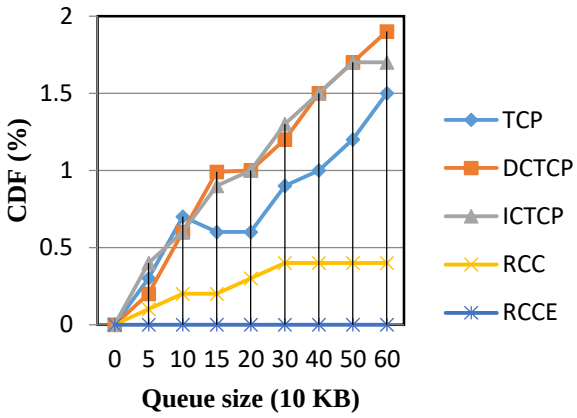
The bandwidth is 1Gbps,link delay is 40 µs and the volume of the output queue at switch is 128 KB. DCTCP's dual state machine is disabled, because all the upcoming flows are accompanied by disabling the delayed ACK mechanism. The Retransmission Time Outs for sender and receiver (RTO^s *and* RTO^r) are set to 10ms, queue_maxsize is 40 packets and hence the threshold for ECN marking is also set to 40 packets. Flow size is used as priority and the switch queue adopts drop tail mechanism.

A. Queue buildup

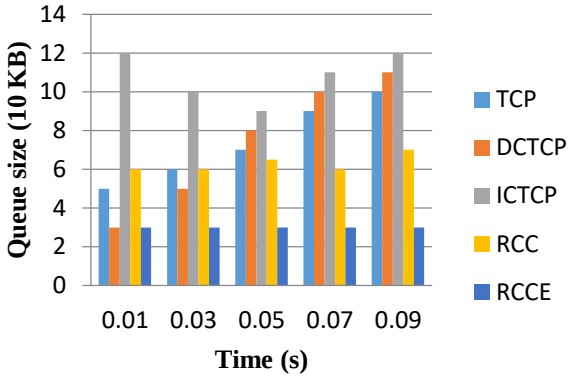
The queue oscillations during packet transmission in the many_to_one traffic pattern are plotted in Fig. 3. With reference from Fig. 3(a), RCCE has the smallest mean queue size when compared withthree different mechanisms excluding 10 senders, as ICTCP handles cause because of networks with low-load. RCCE uses buffer queue to limit the flight windows when the sum is beyond queue_maxsize. It achieves 50.5%, 62.3% and 50.3% reduction in the average queue size over TCP, DCTCP and ICTCP respectively. In addition, it holds low and stable queue in the 95th percentile queue size. These results are taken from the centralized scheduler. Fig. 3(b) and (c) demonstrates the queue size of RCCE with respect to CDF (Cumulative Distribution Function) and queue capacity of 10 KB with 100 senders. Here, it experiences minimum queue length in the major time because the centralized scheduler assures that the flight_windows are under the queue_max of 60 KB. RCCE has stable queue size around 40 KB as the the bandwidth is 1 Gbps and the basic RTT is 160 µs, resulting in 20 KB difference between 60 KB and 40 KB. During transmission, many TCP synchronous packets make coexistence with packets, leading to bigger queue size in RCCE and later it decreases because flows with higher priority havebeen finished.



a) The mean queue size with 5th and 95th percentile of queue size



b) The CDF of queue size with 100 senders



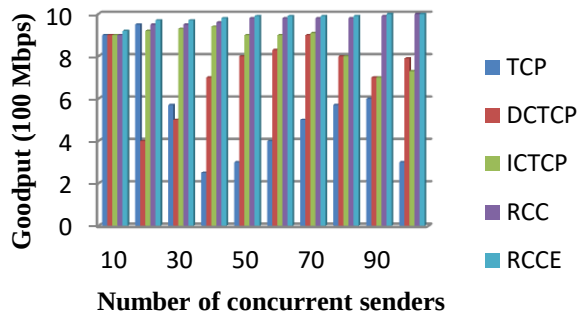
c) Time series of queue size with 100 senders

Fig. 3. Queue builup in the last-hop switch

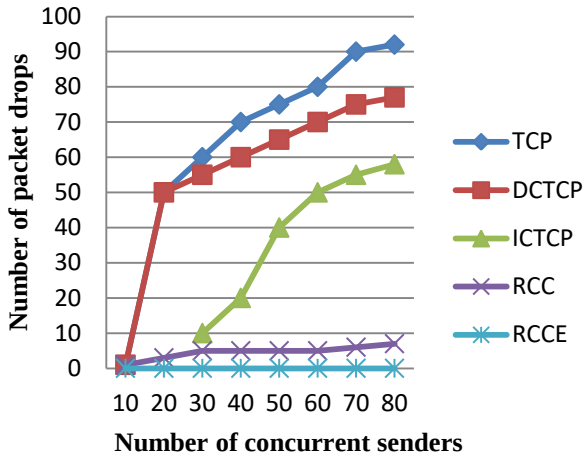
B. Goodput, packet-drops and ECN markings

As implied in Fig. 4(a) all the protocols except RCCE suffers from goodput collapse. RCCE has better goodput due to its stable queue size. At 20 senders, TCP has higher goodput than DCTCP because of ECN which extends congestion to adequate rounds instead of excluding them leading to heavy packet loss. ICTCP suffers goodput collapse when senders are above 50 because its interval of control 2*RTT is not suited for switch queue. For example, at the initial RTT 50 senders sends 50 packets and at the second RTT, it sends 100 packets but the size of the switch queue is only 80.With reference from Fig. 4(b), in all situations RCCE has no packet loss because its buffer queue absorbs burstiness till all flows reach the receiver. DCTCP drops fewer packets than TCP because effective ECN mechanism works well in the sender as well as the receiver. ICTCP also drops fewer packets than TCP because sender window size is adjusted based on the receiver window size rwnd from the receiver. Fig. 4(c) plots ECN markings on the y-axis. Protocols like ICTCP and TCP have no ECN processing to cope up with ECN packets, leading to hundreds of ECN markings. Though DCTCP has ECN mechanism, it does not have window assignment function,

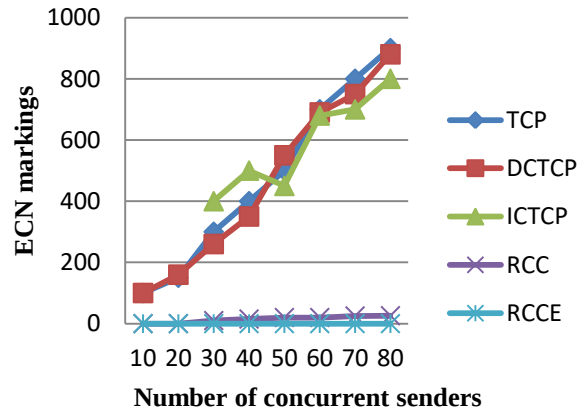
hence the packets beyond switch queue capacity are marked with ECN. RCCE has no ECN markings since it regulates a balanced and small size in switch queue.



a) Goodput



b) Packet-drops

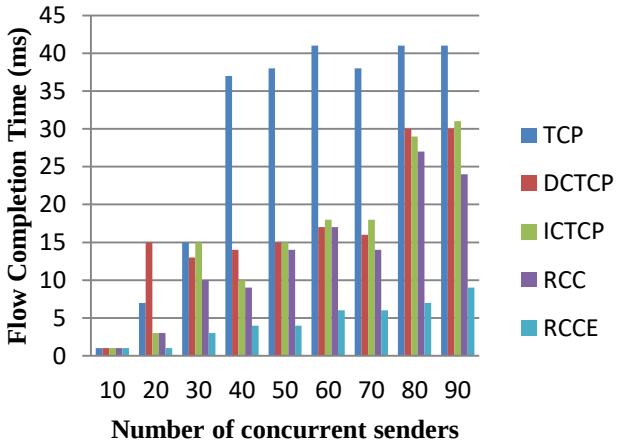


c) ECN markings

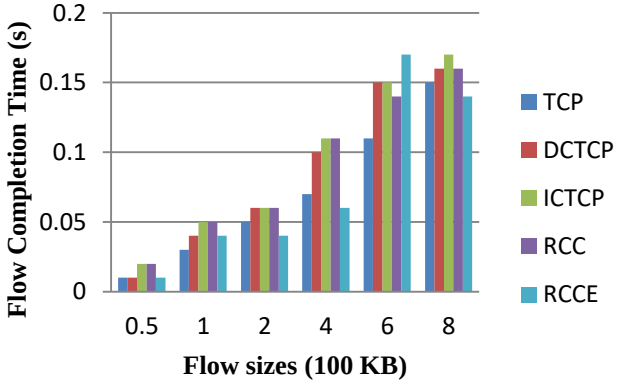
Fig. 4. Goodput, packet-drops and ECN markings of packets in the last hop of switch in heavy incast

C. Flow Completion Times (FCT)

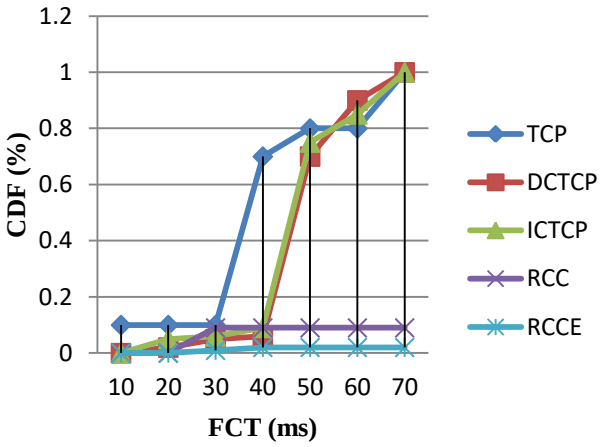
This subsection presents FCTs of RCCE during heavy incast. Fig. 5 (a) plots the 1st,5th and 99th percentile FCTs with different senders each sending 32 KB of data flows. Initially, the entire 50th percentile FCTs increases. But in the 99th percentile FCT, RCCE holds the lowest among all other mechanisms, which indicates that it has better QOS in the partition-aggregate construction. It achieves the 99th percentile completion time over all the mentioned protocols. In addition, itmaintains a stable performance during the entire flow and holds the smallest difference between the 1stand 99th percentile FCTs. In Fig. 5(b) and (d), the numbers of senders are fixed by 50. Fig. 5(b) depicts FCTs of data flows with varying sizes. Initially, TCP has the minimum 1st percentile RTT, but at the 99th percentile RTT, its FCTs are too large due to severe packet drops caused by proactive ACK control. Since 30 senders cause a moderate incast, DCTCP, ICTCP and RCC achieved effective FCT distributions. RCCE though initially has the second smallest 1st percentile FCT, its 99th percentile FCT is minimum compared with other four mechanisms. Fig 5(d) has FCTs similar to 5(b), but DCTCP ICTCP and RCC achieves comparable FCTs with RCCE.



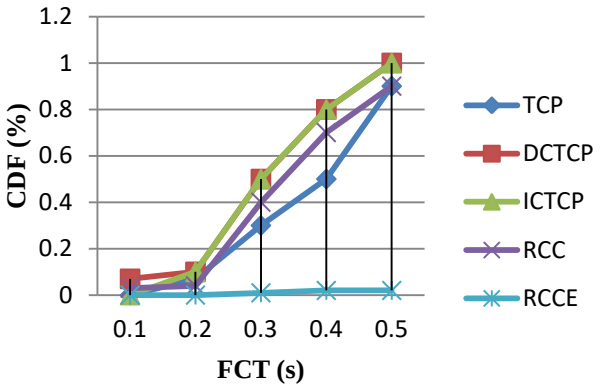
a) The 1st, 5th and 99th percentile of FCT with number of senders sending 32 KB flows



b) The 1st, 5th and 99th percentile of FCT with 100 senders sending flows of different sizes



c) 100 senders with 32 KB each



d) 30 senders with 1000 KB each

Fig. 5. FCTs with varying number of senders and flow sizes

Fig. 5(c) illustrates CDF of FCTs having 100 senders. There, RCCE has accurate distribution of FCTs. This happens because of its load balancing mechanism that balances the network even though there is heavy incast. If any packet drops occur, buffer queue queues up ACK packets based on flow priority. When 100 flows with the same size(same priority) arrives at the receiver, none of the flows must get delayed in the switch queuing buffer at some time. The entire flows have been completed around 28 ms FCTs. If any ACKs have been stored in the buffer queue, the initial flow has around 16 ms FCTs and the flows in the buffer queue has around 30 ms FCTs. RCC, TCP, ICTCP and DCTCP have two, three or four evident distribution respectively.

VII. CONCLUSION

The tendency of big-scale DCNs is indomitable now-a-days. Cloud based applications are attracting large quantity of end customers and escalating their product distribution on data center networks. The many-one-traffic pattern which is ubiquitous make TCP face challenges from incast congestion problem. This paper proposes a hybrid framework using RCC and Edge for avoiding this problem and implementation results imply that it lowers FCTs and ensures availability of traffic in the nearby location of receiver. Further it also reduces packet loss and ECN markings during transmission. The framework considered reflects exact recognition of definitive TCP. It is adaptable for larger-extensibleincast traffic patterns and has efficiently overcome congestion. The contribution is done without modifying the networking hardware to enable quick and true deployment potential in critical DCNs. Future work involves practical aspects of RCCE in at-scale simulations (ns3) in network scenarios with diverse and heavy workloads. Also work will be extended to provide transmission performance analysis including multiple bottleneck topology, flow priority, dynamic traffic as well as convergence and coexistence with TCP.

REFERENCES

1. Al-Fares, Mohammad, Alexander Loukissas, and Amin Vahdat, "A scalable, commodity data center network architecture", In *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63-74, Aug. 2008.
2. Bari, Md Faizul, Raouf Boutaba, Rafael Esteves, Lisandro Zambenedetti Granville, Maxim Podlesny, Md Golam Rabbani, Qi Zhang, and Mohamed FatenZhani, "Data center network virtualization: A survey", *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp: 909-928, Apr. 2013.
3. Al-Fares, Mohammad, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat, "Hedera: Dynamic flow scheduling for data center networks", In *Nsdi*, vol. 10, pp. 19-19, Apr. 2010.
4. How big is facebooks data? 2.5 billion pieces of content and 500+ terabytes ingested every day, http://goo.gl/n8xhq.
5. Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters", *Communications of the ACM* 51, no. 1, pp. 107-113, Jan. 2008.
6. Kandula, Srikanth, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken, "The nature of data center traffic: measurements &analysis", In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pp. 202-208, Nov. 2009.
7. Zhang, Jiao, Fengyuan Ren, and Chuang Lin, "Modeling and understanding TCPincast in data center networks", In *INFOCOM, 2011 Proceedings IEEE*, pp. 1377-1385, Apr. 2011.
8. Dukkipati, Nandita, and Nick McKeown. "Why flow-completion time is the right metric for congestion control." *ACM SIGCOMM Computer Communication Review* 36, no. 1, pp. 59-62, Jan. 2006.
9. Abdelmoniem, M. Ahmed, BrahimBensaou, and AmudaJames Abu, "SICC: SDN-based incast congestion control for data centers", In *Communications (ICC), 2017 IEEE International Conference on*, pp. 1-6, May. 2017.
10. Xu, Lei, Ke Xu, Yong Jiang, Fengyuan Ren, and Haiyang Wang, "Throughput optimization of TCP incast congestion control in large-scale datacenter networks", *Computer Networks* 124, pp. 46-60, Sep. 2017.
11. Rob Nash-Boulden , "The Data Center Journal: Edge Compting: Is it right for your business?", Nov. 2017, [Online].Available:http://www.datacenterjournal.com/edge-computing-right-business/
12. DataCenter Knowledge, "Five Edge Data Center Myths", Dec. 2017, [Online]. Available:http://www.datacenterknowledge.com/edgecomputing/five-edge-data-center-myths
13. Wu, Haitao, Zhenqian Feng, Chuanxiong Guo, and Yongguang Zhang, "ICTCP: Incast congestion control for TCP in data-center networks", *IEEE/ACM transactions on networking* 21, no. 2, pp. 345-358, Apr. 2013.
14. Tsaoussidis, Vassilios, and Chi Zhang. "TCP-Real: receiver-oriented congestion control." *Computer Networks* 40, no. 4, pp. 477-497, Nov. 2002.
15. Bai, Wei, Kai Chen, Haitao Wu, Wuwei Lan, and Yangming Zhao, "Pac: Taming tcpincast congestion using proactive ack control" In *Network Protocols (ICNP), 2014 IEEE 22nd International Conference on*, pp. 385-396, Oct. 2014.
16. Xu, Lei, Ke Xu, Yong Jiang, Fengyuan Ren, and Haiyang Wang, "Enhancing TCP Incast congestion control over large-scale datacenter networks", In *Quality of Service (IWQoS), IEEE 23rd International Symposium on*, pp. 225-230. Jun. 2015.

17. He, Keqiang, Eric Rozner, Kanak Agarwal, Wes Felter, John Carter, and Aditya Akella, "Presto: Edge-based load balancing for fast datacenter networks", In *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 465-478, Aug. 2015.
18. Satyanarayanan, Mahadev, ParamvirBahl, Ramón Caceres, and Nigel Davies "The case for vm-based cloudlets in mobile computing" *IEEE pervasive Computing* 8, no. 4, Oct. 2009.
19. Agarwal, Sharad, MatthaiPhilipose, and ParamvirBahl. "Vision: The case for cellular small cells for cloudlets." In *Proceedings of the fifth international workshop on Mobile cloud computing & services*, pp. 1-5, June. 2014.
20. Brad Hedlund, "TCP Incast and Cloud application performance", May 2011, [Online]. Available:http://bradhedlund.com/2011/05/01/tcp-incast-and-cloud-application-performance/
21. Brandon Butler," What is Edge Computing and how it's changing the network", Sep. 2017, [Online]. Available: https://www.networkworld.com/article/3224893/internet-of-things/what-is-edge-computing-and-how-it-s-changing-the-network.html
22. Big Data and Analytics Hub, "The four V's of Big data", [Online].Available:http://www.ibmbigdatahub.com/infographic/four-vs-big-data
23. Benson, Theophilus, Ashok Anand, Aditya Akella, and Ming Zhang, "MicroTE: Fine grained traffic engineering for data centers", In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, pp. 8, Dec. 2011.
24. Ballani, Hitesh, Paolo Costa, Thomas Karagiannis, and Ant Rowstron, "Towards predictable datacenter networks" In *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 242-253. Aug. 2011.
25. Alizadeh, Mohammad, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, SudiptaSengupta, and Murari Sridharan, "Data center tcp (dctcp)" In *ACM SIGCOMM computer communication review*, vol. 40, no. 4, pp. 63-74, Aug. 2010.
26. Rojas-Cessa, Roberto, YagizKaymak, and Ziqian Dong, "Schemes for fast transmission of flows in data center networks" *IEEE Communications Surveys & Tutorials* 17, no. 3, pp. 1391-1422, Aug. 2015.

AUTHORS PROFILE

Ms.K.ArunSelvi is presently working as an Assistant Professorin National Engineering College, Kovilpatti, Tamil Nadu. She did her B.E (Computer Science and Engineering) Degree in 2016 from Anna University, Chennai and M.E (Computer Science and Engineering) Degree in 2018 from Anna University, Chennai. Her research interests include Data Center Networking, Software Defined Networking, Machine Learning and Image Processing.

Dr.K.Kumaris presently working as an Assistant Professor in Government College of Technology, Coimbatore, Tamil Nadu. He has over 18 years of Teaching Experience. He did his B.E (Computer Science and Engineering) Degree in 1999 from University of Madras and M.E (Computer Science and Engineering) Degree in 2005 from Anna University, Chennai. He did his Ph.D in 2013 from Anna University, Chennai. He has to his credit many International and National journal Publications. Many of his papers had been published in IEEE Explore, Elsevier and Springer. His research interests include Ad Hoc Networks, Wireless Security, Cryptography, Named Data Networking, Software Defined Networking and Machine Learning.

Ms.K.Ramalakshmiis presently workingasan Assistant Professorin National Engineering College, Kovilpatti, Tamil Nadu. She did her B.E (Computer Science and Engineering) Degree in 2009 from Anna University, Chennaiand M.E (Computer Science and Engineering) Degree in 2011 from AnnamalaI University, Chidambaram. Her research interests include Cloud Computing, Speech enhancement, Machine Learning.

Ms.A.Sathiya is presently working as an Assistant Professorin National Engineering College, Kovilpatti, Tamil Nadu. She did her B.E (Computer Science and Engineering) Degree in 2016 from Anna University, Chennai and M.E (Computer Science and Engineering) Degree in 2018 from Anna University, Chennai. Her research interests include Data Mining, Machine Learning and Computer Networks.