

# Implementation of the Algorithm for Congestion control in the Dynamic Circuit Network (DCN)

H S Nalamwar<sup>1</sup>, M A Ivanov<sup>1</sup> and G U Buddhawar<sup>2</sup>

<sup>1</sup> Tomsk Polytechnic University, 30, Lenina Avenue, Tomsk, 634050, Russia

<sup>2</sup> NCET, Mouza Bodli, Dhanora Road, Gadchiroli, MS, 442605, India

E-mail: hitesh@tpu.ru

**Abstract.** Transport Control Protocol (TCP) incast congestion happens when a number of senders work in parallel with the same server where the high bandwidth and low latency network problem occurs. For many data center network applications such as a search engine, heavy traffic is present on such a server. Incast congestion degrades the entire performance as packets are lost at a server side due to buffer overflow, and as a result, the response time becomes longer. In this work, we focus on TCP throughput, round-trip time (RTT), receive window and retransmission. Our method is based on the proactive adjust of the TCP receive window before the packet loss occurs. We aim to avoid the wastage of the bandwidth by adjusting its size as per the number of packets. To avoid the packet loss, the ICTCP algorithm has been implemented in the data center network (ToR).

## 1. Introduction

In this work, our discussion is mainly focused on the congestion problem that usually occurs in incast, a condition which is the opposite of broadcast. In broadcast, one node sends messages to multiple nodes; in incast - on the contrary - multiple nodes send messages to one and the same node. The TCP (Transport Control Protocol) is a reliable protocol and has been widely used in the Internet. The congestion happens when many synchronized servers work under the same switch and TCP is not very reliable for high-bandwidth and low latency networks. Previous solutions focused on either reducing the wait time for packet loss recovery with faster retransmissions [1, 2, 3], or controlling switch buffer occupation to avoid overflow by using ECN and modified TCP on both the sender and receiver sides [4, 5]. This report focuses on avoiding the packet loss before incast congestion, which seems more appealing. Our idea is to perform the incast congestion avoidance at the receiver side by preventing incast congestion. The receiver side is a natural choice since it knows the throughput of all TCP connections and the available bandwidth. The receiver side can adjust the receive window size of each TCP connection so that the aggregate bursts of all the synchronized senders are kept under control. We suggest calling this design Incast congestion Control for TCP (ICCTCP). However, adequately controlling the receiver window is challenging. The receiver window should be small enough to avoid incast congestion, but also large enough for good performance and other non-incast cases. The technical novelties of this work are as follows:

- 1) To perform congestion control on the receiver side, we use the available bandwidth on the network interface as a quota to coordinate the receiver window increases of all incoming connections.
- 2) Our per flow congestion control is performed independently of the slotted time of RTT of each



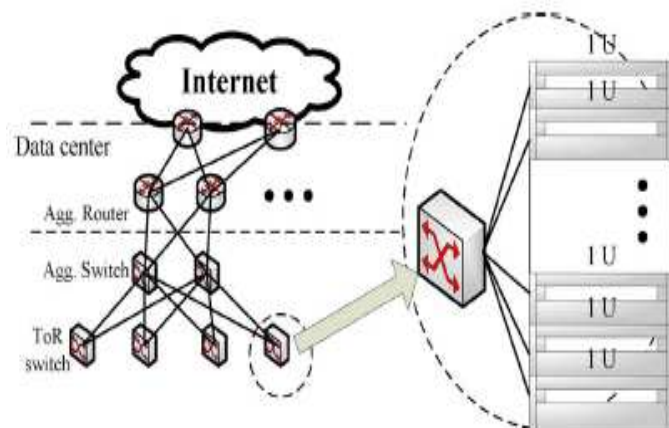
connection, which is also the control latency in its feedback loop.

3) Our receiver window adjustment is based on the ratio of the difference between the measured and expected throughput over the expected.

The live RTT is necessary for throughput estimation as we have observed that TCP RTT in a high bandwidth of low latency network increases with throughput, even if the link capacity is not reached. TCP incast has been identified and described in a distributed storage cluster [5]. In distributed file systems, the files are deliberately stored in multiple servers. However, TCP incast congestion occurs when multiple blocks of a file are fetched from multiple servers at the same time. Several applications of specific solutions have been proposed in the context of parallel file systems.

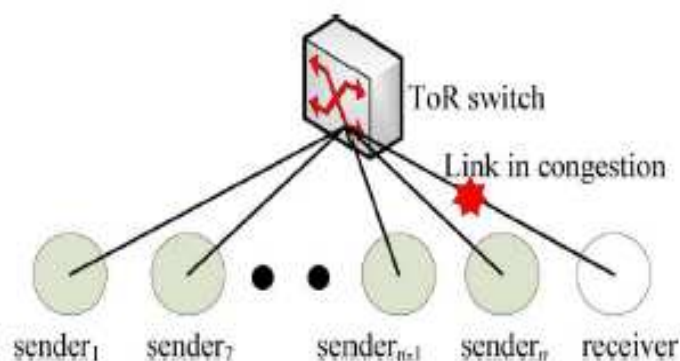
## 2. TCP Incast Congestion

Fig. 1 shows a typical data-center network structure. There are three layers of switches/routers, in particular the ToR switch, the Aggregate switch, and the Aggregate router. The figure also illustrates a detailed case for a ToR switch connected to dozens of servers.



**Figure 1.** Data-center networks and a detailed illustration of a ToR switch connected to multiple rack-mounted servers.

Incast congestion happens when multiple sending servers under the same ToR switch send data to one receiver server simultaneously. This process is demonstrated in Fig. 2.



**Figure 2.** The scenario of incast congestion in data-center networks, where multiple TCP senders transmit data to the same receiver under the same ToR switch.

The amount of data transmitted by each connection is relatively small, e.g. 64kb. The term goodput is effective throughput obtained and observed at the application layer. The multiple TCP connections are barrier-synchronized. First, these connections are established between all senders and the receiver, respectively. Then, the receiver sends out a (very small) request packet to ask each sender to transmit data, that is, multiple requests packets are sent using multiple threads. The TCP connections are issued

round by round, and one round ends when all connections on that round have finished their data transfer to the receiver. Here, we can observe similar goodput trends for three different traffic amounts per server, but with slightly different transition points.

### 3. Motivation

As we have mentioned, in broadcast, one node sends messages to multiple nodes, while in incast the process is quite opposite, that is multiple nodes send messages to one and the same node. When a number of servers work in parallel with a single server, network congestion occurs. Network congestion is the situation in which an increase in data transmission results in proportionately smaller increase, or even a reduction, in throughput. Congestion degrades the performance of the network. Packet loss is the major reason for the congestion.

Packets are having two numbers that are associated with each packet.

1. A sequence number indicates the ID of a packet.
2. An acknowledgement number (ACK) indicates the expected acknowledge (ACK) number of the packet that is associated with it.

### 4. TCP Timeout

Timeout happens when the sender does not receive an ACK for a long period (retransmission or RTO). There are several reasons for the timeout due to congestion given below:

- a) In distributed file systems, as the files are deliberately stored in multiple servers at the time of fetching, congestion occurs.
- b) With the recent progress in data center networking, TCP incast problems in the data center network have become a practical issue.
- c) TCP throughput is severely degraded by incast congestion since one or more TCP connections can experience timeout caused by packet drops.
- d) TCP variants sometimes improve performance, but fail to prevent incast congestion collapse since most of the timeouts are caused by buffer overflow.

Based on the analysis of the characteristics of the data center network, the common pattern and the TCP congestion control algorithm, the results of TCP incast are as follows:

1. Since the top of rack switches is shallow buffered, highly bursty, fast and simultaneous, data transmissions overflow the switch buffer to make packet losses.
2. Mass Packet losses will lead to TCP congestion control, reducing the sending window by half and decreasing the sending rate.
3. The intense packet loss results in TCP timeouts. The TCP timeout lasts for 100 milliseconds, but the round trip time of the data center network is around 100 microseconds. Coarse grained RTO reduces the application throughput by 90%.

### 5. Problem Analysis

The root cause of TCP incast collapse is that the highly bursty traffic of multiple TCP connections overflows the Ethernet switch buffer in a short period of time, causing the intense packet loss resulting in the TCP retransmission and timeouts. In distributed file systems, TCP incast congestion occurs when multiple blocks of a file are fetched from multiple servers, as files are stored at multiple servers. To resolve issue, several application specific solutions have been proposed in the context of the parallel file system. Due to recent progresses on DCN, the TCP incast problem is a common problem. Since there are various applications running in DCN, the best solution is to keep the application free and keep responsibility at a transport layer for building effective solutions. It is a preferred way to avoid congestion and timeout for TCP incast. Our idea is to perform incast congestion avoidance at the receiver side by preventing incast congestion. The receiver side is a natural choice since it knows the throughput of all TCP connections and the available bandwidth. The receiver side can adjust the receiver window size of each TCP connection.

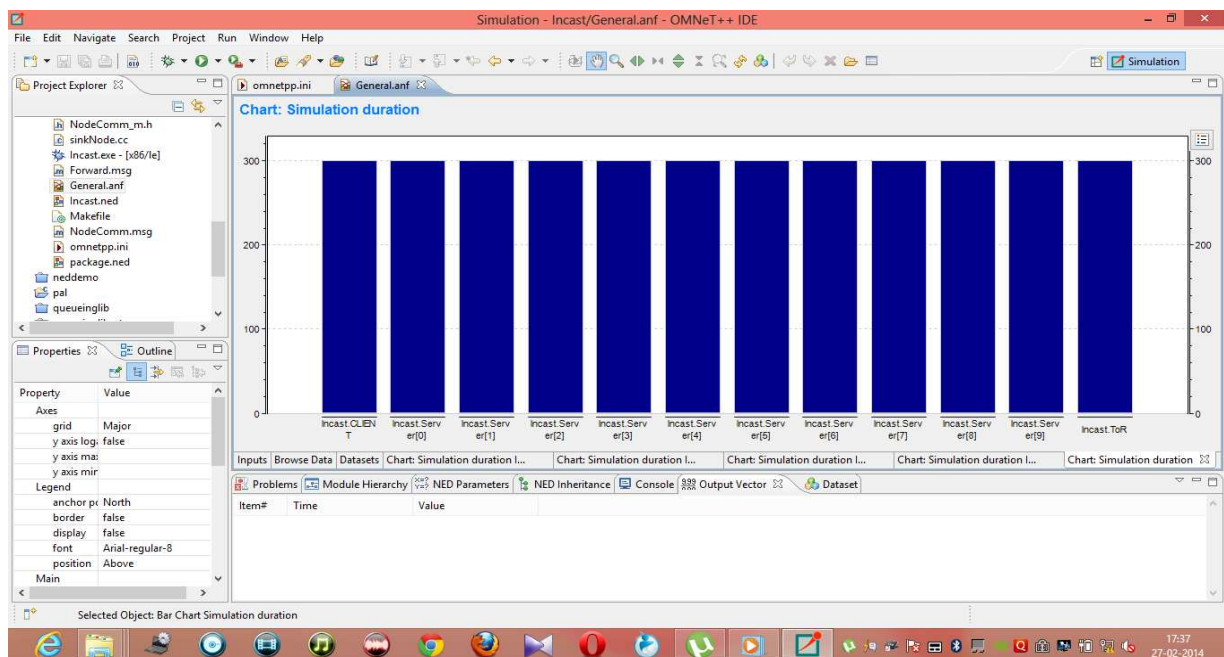
## 6. Working Modules

Here, we have the TOR switch which is a datacenter network.

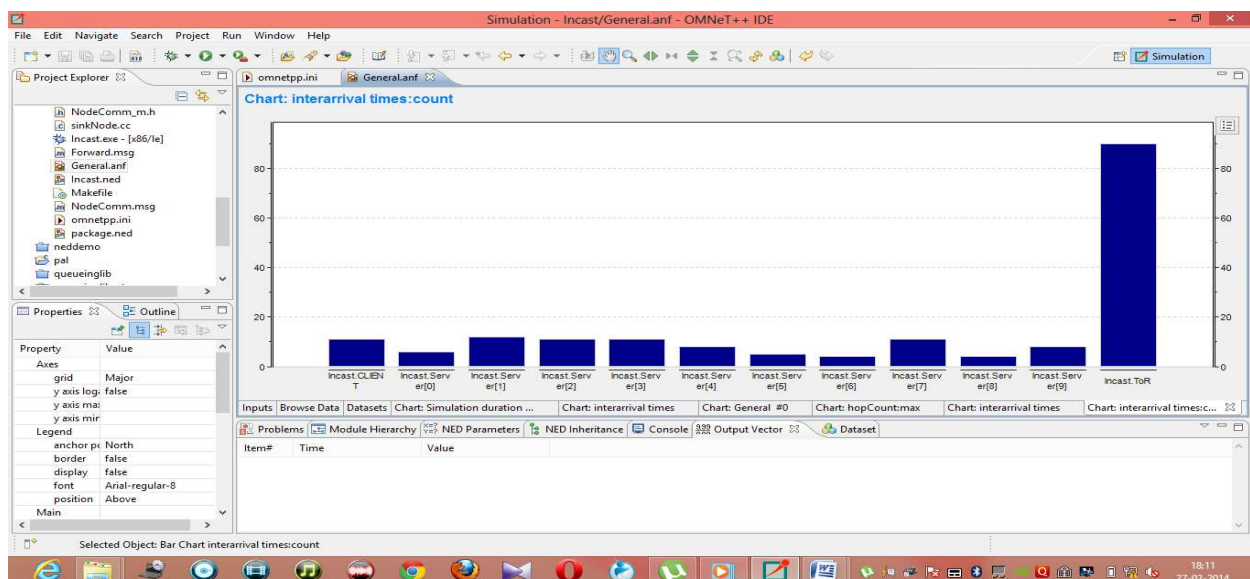
- The ToR switch is acting as a data center network.
- All traffic is managed at the ToR Switch.
- All the data/Packets from a number of servers are collected at ToR and then transferred to a client.
- ToR manages all the traffic from different servers and transfers it to a client to avoid the packet loss.

## 7. Experimental Results and Discussions

Figures 3 and 4 show the results obtained after the implementation of the ICTCP algorithm in the platform.



**Figure 3.** Simulation duration of the packet on the server, which is uniform in the system.



**Figure 4.** The inter-arrival time representing time between each arrival of the packet on the receiver side

## 8. Advantages

Table 1 compares the efficiency of the retransmission algorithm to the efficiency of the ICTCP algorithm, with the latter demonstrating obvious advantages

No	Retransmission algorithm	ICTCP algorithm
1	Buffer Overflow Problem	Buffer overflow is impossible
2	Timeout problem	Time out problem is less possible
3	Load increase on the receiver side	Load does not increase on the receiver side
4	Both windows need to be adjusted	Only receiver windows need to be adjusted
5	Wait time reduced by retransmission	Not in the ICTCP algorithm
6	Bandwidth is not preserved	Bandwidth is preserved to a large extent as a means of retransmission avoidance

**Table 1.** Efficiency of the ICTCP algorithm over the retransmission algorithm

## 9. Conclusion

The paper describes the design and implementation of the ICTCP algorithm to improve TCP performance for TCP incast. The previous approaches based on a fine tuned timer for faster retransmission or controlling of a switch buffer are different from ICTCP. The paper focuses on the receiver based congestion control algorithm where ICTCP proactively adjusts the TCP receiver window based on the ratio of difference of achieved and expected throughputs per connection over expected throughputs, as well as the available bandwidth at the receiver end. Our experimental results demonstrate that ICTCP is effective to avoid congestion by achieving almost zero timeout for TCP incast, and packet losses are reduced to a large extent, and thus the bandwidth is considerably preserved. The ICTCP method is also used for the preservation of congestion in the network.

## References

- [1] Vasudevan V, Phanishayee A, Shah H, Krevat E, Andersen D, Ganger G, Gibson G, and Mueller B 2009 Safe and effective fine-grained TCP retransmissions for datacenter communication *Proc. ACM SIGCOMM* 303–314
- [2] Kandula S, Sengupta S, Greenberg A, Patel P, and Chaiken R 2009 The nature of data center traffic: Measurements & analysis *Proc. IMC* 202–208
- [3] Alizadeh M, Greenberg A, Maltz D, Padhye J, Patel P, Prabhakar B, Sengupta S and Sridharan M 2010 Data center TCP (DCTCP) *Proc. SIGCOMM* 63–74
- [4] Krevat E, Vasudevan V, Phanishayee A, Andersen D, Ganger G, Gibson G and Seshan S 2007 On application-level approaches to avoiding TCP throughput collapse in cluster-based storage systems *Proc. Supercomput.* 1–4
- [5] Al-Fares M, Loukissas A and Vahdat A 2008 A scalable, commodity data center network architecture *Proc. ACM SIGCOMM* 63–74
- [6] Guo C, Lu G, Li D, Wu H, Zhang X, Shi Y, Tian C, Zhang Y and Lu S 2009 BCube: A high performance, server-centric network architecture for modular data centers *Proc. ACM SIGCOMM* 63–74
- [7] Brakmo B and Peterson L 1995 TCP Vegas: End to end congestion avoidance on a global internet *IEEE J. Sel. Areas Commun.* **13** 1465–1480