# Analysis for Unfairness of TCP Outcast Problem in Data Center Networks

Yang Qin[+]*, Yao Shi, Qiwei Sun, Liqun Zhao
[+] Key Laboratory of Network Oriented Intelligent Computation, Shenzhen Graduate School,
Harbin Institute of Technology .Shenzhen, P. R. China
*corresponding author: yqinsg@gmail.com

***Abstract*—There is a form of unfairness phenomenon found in data center networks,which was identified by P. Prakash. This phenomenon shows that the throughput of a flow with smaller RTT is less than the throughput of a flow with large RTT. This is completely contrary to the classic TCP protocol. P. Prakash gives an explanation based on port blackout theory. However, after careful analysis, we have found another reason for this unusual phenomenon, which may be more important. This is because of an unfair distribution of the flows with different RTT on the physical link and the differences in congestion window size when finishing the current block transmission. In this paper, we develop an analytical model for the goodput in the outcast behavior. Then, we design a window notification based protocol to address this phenomenon. By using the mean value of the congestion window measurement, we unified the congestion window size for flows with different RTT, which improves the goodput of the flows with small RTT. By designing several experiments using the ns-2 simulator, we demonstrate that our explanation of outcast phenomenon is correct. Furthermore, we verify the effectiveness of the proposed algorithms.**

**Keywords—data center network; congestion window; outcast**

## I. INTRODUCTION

There isn't a standard definition for a data center. Generally speaking, a data center consists of facilities for placement of computer systems and related components, such as storage systems, the environment and security equipment [1]. In recent years, the BCube architecture[2] and the Fat-tree architecture shown in Fig. 1 are widely used in the data center [3] .

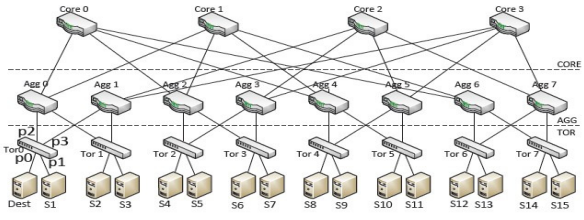


Fig. 1. Fat-tree framework topology diagram[6].

Fat-tree framework for data center was presented at the ACM SIGCOMM 2008 conference. In data center networks, fat-tree architecture is widely used and researched for its scalable characteristics [8]. Fat-tree framework has a special routing strategy. This strategy allows receivers to get data from senders with fixed links [3].

The main features of a data center network comprises: high-speed links, the lower switch propagation delay and limited cache. The TCP protocol has not only become the most successful transport layer protocol on the Internet, but also widely used in data center networks. However, traffic load and link environments are different between data center network and the Internet, the TCP protocol is not fully applicable for data center networks because the performance of TCP in data centers is different from that in the Internet. It is worth while researching the corresponding issues. Outcast is one of the special problems from using the TCP protocol in data center network.

P. Prakash first described the outcast phenomenon[6]. In that paper, he also validated the outcast problem in different data center testbeds and analyzed the reason for this phenomenon. The main reason he believed is the port blackout phenomenon. Port blackout can be essentially explained as follows. When a series of packets enters the switch from different input ports, they must compete for the only output port. In these different input ports, some of them may luckily get into the switch cache, while the rest of them could only be discarded, since the queue in the output port is full.

## II. THE REASON OF THE OUTCAST PHENOMENON

In order to look inside for the essential reason for this outcast, we set up a network in Fig. 2, which consisted of three senders, three switches and one receiver.

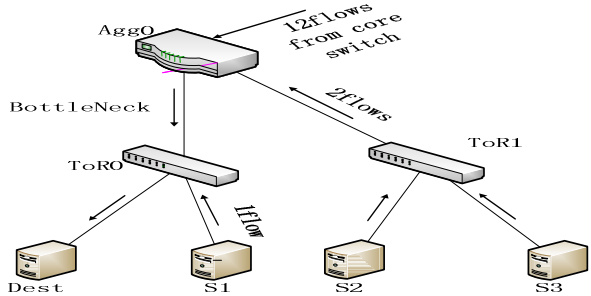


Fig. 2. The unfairness on the distribution of flows on different links.[4].

One of the physical links carries 12 flows, one carries 2 flows and another one carries only one flow. This simplified experiment is based on the routing algorithm which is used by the fat-tree topology[3]. We adopt the routing algorithm which is used by the fat-tree topology[3]. We also know that the receiver cannot ask for the next data block until it receives all current blocks. Based on the information above, we can determine the reason for this outcast phenomenon.

(1). When the client requests data blocks from the servers, we can confirm that the flows with small RTT (round trip time) can complete earlier than the flows with large RTT. This is mainly because when a connection is newly created, the congestion cannot immediately appear, and according to the characteristics of data center applications, only when all senders complete the transmission of the current data block, can the receiver request the next data block. Thus, the senders with small RTT can only wait until the other senders with large RTT finish the transmission of their current data block, then they can send the next data block. In this case, the number of flows on the bottleneck decreases. That means each sender can obtain more bandwidth. Such a direct consequence of this event is that when these senders finish sending the data packets, they would have a larger congestion window.

Conclusion: After the end of the first data block transmission, the situation is that flows with small RTT have a smaller congestion window, while flows with large RTT have a larger congestion window.

(2). We assume that when the second data block starts sending, the network suffers packet loss. From the overall observation, each input port has the same probability of whether their packets are discarded, since we assume that the incoming data packets follows a Poisson distribution. Assume else that each input port loses k packets, due to the cache for the bottleneck switch is full. We can see that if there are more flows on the link, then less packets, on average, get lost on each connection; conversely, the fewer flows on a link, the more packets on average get lost on each connection.

Conclusion: When packets loss occurs, flows with small RTT lose more packets than the flows with large RTT.

(3). Combining the fast transmission event with the first and second conclusions, we can see that because the congestion window of the flows with small RTT is small and they lose more packets, they are more likely to lose the packets whose corresponding ACK is not received in the current congestion window. This will obviously lead to timeout. At the same time, flows with large RTT are less likely to suffer timeout except they just lose one of the last three packets, since their congestion windows are large while they suffer less in packet loss. According to the analysis in section Ⅲ, timeout and RTOmin time have the largest impact on the whole throughput of the network.

Conclusion: When flows with small RTT suffer packet loss, they are more likely to be affected, which will result in a decrease of throughput. Conversely, flows with large RTT will not show similar behavior.

From the three conclusions above, we can see that the substantial causes of the outcast phenomenon are the uneven distribution of flows with different RTT on the physical links and the characteristics of the applications which run on the data center network. These two reasons work together to make the final result -- the throughput of flows with small RTT is less than the flows with large RTT. In the next section, we will give a simple mathematical model on the outcast phenomenon.

## III. THROUGHPUT MODEL OF THE OUTCAST PHENOMENON

In this section, we shall respectively develop an analytical model for throughput of flows with small RTT and flows with large RTT to reinterpret the outcast phenomenon. In our model, we only describe the congestion avoidance and fast retransmit states. We assume that these are the only flows with large RTT and small RTT we defined existing in the bottleneck link, without any other data flow. Our model uses the fat-tree architecture[3]. The upper part of the fat-tree architecture is used the core switches and the middle part is used for the aggregation switches. The flows with larger RTT come from servers that cannot communicate with the target node through the same aggregation switch, so they must go through the upper core switch . The flows with small RTT come from servers which communicate with the same aggregation switch with the target node ,so they do not need to use the upper core switch. There are lots of servers connected to the upper core servers and few servers can connect to the same aggregation switch with the target node. So the number of the flows with large RTT is greater than the number of the flows with small RTT.

Assume that there are $N$ senders, including $N_s$ senders with small RTT and $N_l$ senders with large RTT. Since the differences of RTT are mainly reflected on the time from the senders to the bottleneck switch, we define the round trip delay of the flows with small RTT as $RTT_s$ and delay of the flows with large RTT as $RTT_l$. The default retransmission timeout value is $RTO$ . The bandwidth of the bottleneck is $C$ $pkts/s$ , and the buffer of the bottleneck switch is $B$ $pkts$ . The payload of each packet is $S_p$ $bytes$ . Since we only consider the congestion avoidance state, we define $W_i^s$ as the congestion window size of the flows with small RTT in the $i^{th}$ round, and $W_i^l$ as the congestion window size of the flows with large RTT in the $i^{th}$ round. We further define $R_i$ as the bottleneck delay on the $i^{th}$ round. $Q_i$ is the queue length of bottleneck buffer when the $i^{th}$ round is finished.

First we calculate the dynamic changes for the queue length. During the congestion avoidance phase, when the transmission is in the $i^{th}$ round, $N_s$ flows with small RTT and $N_l$ flows with large RTT send data packets to a single receiver. So we can calculate the number of packets when the $i^{th}$ round is finished.

$$Q_i = \min\left\{(Q_{i-1} + N_s \times W_i^s + N_l \times W_i^l - C \times R_i)^+, B\right\} \quad (1)$$

Where $a^+$ means if $a > 0$, $a^+ = a$; else $a^+ = 0$.

Since the switch uses the first in first out method to manage its queue in the buffer and the propagation delay is a fixed value, we define it as D. We can show that the propagation delay in the $i^{th}$ round plus the queuing delay is the bottleneck delay in the $i^{th}$ round $R_i$.

$$R_i = D + (Q_{i-1} + \varphi) / C \tag{2}$$

where $\phi$ means a random variable when the queuing delay exceeds $Q_{i-1} / C$.

### A. Throughput model of flows with small RTT

In this section, we calculate the goodput of the flows with small RTT. First, we calculate the maximum size of the congestion window in the congestion avoidance state. As we all know, after the end of each round, the congestion window size will increase by one maximum segment size. This situation repeats until there is one packet getting lost in the $n^{th}$ round. Then we can consider that the time from the first round to the $n^{th}$ rounds makes up the congestion avoidance phase. We define $W_n^s$ as the maximum congestion window size of flows with small RTT in this phase, $W_n^l$ as the maximum congestion window size of flows with large RTT. When the window reaches to $W_n^s$, there are $d_n^s$ packets getting lost. We take a maximum possible value for the number of lost packets, i.e. the whole window $W_n^s$. The number of packets lost we calculate is from the period once the congestion window size is halved to the next time the congestion window halved. In this way, we can calculate the total number of packets sent in this stage $S_n^s$.

$$S_n^s = \sum_{j=0}^{\frac{W_n^s}{2}} \left( \frac{W_n^s}{2} + j \right) = \frac{3}{8} \left( W_n^s \right)^2 + \frac{3}{4} W_n^s \tag{3}$$

So we can get the number of successfully transmitted data packets $Y_n^s$ during the congestion avoidance phase.

$$Y_n^s = S_n^s - d_n^s = \frac{3}{8} \left( W_n^s \right)^2 - \frac{1}{4} W_n^s \tag{4}$$

From the equation 1 and equation 2, we can obtain

$$Q_i = \min \left\{ (N_s \times W_i^s + N_l \times W_i^l - C \times D - \varphi)^+, B \right\} \tag{5}$$

We can calculate the difference between $Q_i$ and $Q_{i-1}$ to get the following result.

$$Q_i - Q_{i-1} = N_s + N_l = N \tag{6}$$

So in the $i^{th}$ round, the first packet suffers the delay as $Q_{i-1} / C$. And the last packet suffers the delay as $(Q_{i-1} + N) / C$. Since the arrival and departure rate of the buffer is the same, we can be obtain the expectation of $\varphi$ as: $E(\varphi) = N / 2$.

$$Q_i = \min \left\{ \left( N_s \times W_i^s + N_l \times W_i^l - C \times D - \frac{N}{2} \right)^+, B \right\} \tag{7}$$

$$R_i = (N_s \times W_{i-1}^s + N_l \times W_{i-1}^l) / C \tag{8}$$

Next, we calculate the duration of the congestion avoidance phase. Equation (8) represents the delay time of the bottleneck link. Since the respective slow start thresholds[6] at this time are $W_n^s / 2$ and $W_n^l / 2$, we can see that the congestion windows in the last round of the slow start phase are $W_0^s = W_n^s / 4$ and $W_0^l = W_n^l / 4$. Therefore, the bottleneck link congestion avoidance phase delay time is shown below:

$$T_n = \sum_{i=1}^{n+1} R_i = \frac{N_s}{C} \left( W_n^s + \frac{3}{8} \left( W_n^s \right)^2 \right) + \frac{N_L}{C} \left( W_n^l + \frac{3}{8} \left( W_n^l \right)^2 \right) \tag{9}$$

Since the retransmission timeout is RTO, we can get the total time as below.

$$T_s = T_n + RTO + n \times (RTT_s + RTT_l) \tag{10}$$

We can obtain the goodput equation.

$$G_s = \frac{Y_n^s}{T_s} = \frac{\left( \frac{3}{8} \left( W_n^s \right)^2 - \frac{1}{4} W_n^s \right) \times N_s \times S_p}{\frac{N_s}{C} \left( W_n^s + \frac{3}{8} \left( W_n^s \right)^2 \right) + \frac{N_l}{C} \left( W_n^l + \frac{3}{8} \left( W_n^l \right)^2 \right) + RTO + n \times (RTT_s + RTT_l)} \tag{11}$$

### B. Throughput model of flows with large RTT

The situation of the flows with large RTT is similar to the flows with small RTT. However, when packet loss occurs, the flows with large RTT would not get into a timeout event. Because the congestion window size is large and it won't lose all the packets. Conservatively, they turn into the fast retransmission state. So we choose the minimum number of packets lost, i.e. one data packet, when these flows suffer packet loss. We know in the fast retransmission phase, $W_n^l / 2$ packets are resent. $d_n^l$ is the number of lost packets.

$$\begin{aligned} Y_n^l &= S_n^l - d_n^l \\ &= \frac{3}{8} \left( W_n^l \right)^2 + \frac{3}{4} W_n^l + \frac{W_n^l}{2} - 1 = \frac{3}{8} \left( W_n^l \right)^2 + \frac{5}{4} W_n^l - 1 \end{aligned} \tag{12}$$

The total congestion avoidance duration of the flows with large RTT can be calculated as the flows with small RTT. The difference is that the flows with large RTT turn into fast retransmission instead of timeout. The extra time that the flows with large RTT need is the time for retransmitting $W_n^l / 2$ packets and the propagation delay D. Then we can obtain the total duration time as below.

$$T_l = T_n + D + n \times (RTT_s + RTT_l) \tag{13}$$

We can get the goodput equation of flows with large RTT.

$$G_l = \frac{Y_n^l}{T_l} = \frac{\left( \frac{3}{8} \left( W_n^l \right)^2 + \frac{5}{4} W_n^l - 1 \right) \times N_l \times S_p}{\frac{N_s}{C} \left( W_n^s + \frac{3}{8} \left( W_n^s \right)^2 \right) + \frac{N_l}{C} \left( W_n^l + \frac{3}{8} \left( W_n^l \right)^2 \right) + D + n \times (RTT_s + RTT_l)} \tag{14}$$

Now we can find out that the main differences are on the window size, the number of flows, and the waiting time. In the analysis of the previous section, we already know that $W_n^s < W_n^l$ and $N_s < N_l$. Obviously, we have $RTO > D$. Finally, we have the conclusion that the goodput of flows with small RTT is smaller than flows with large RTT.

## IV. A WINDOW SIZE NOTIFICATION BASED ALGORITHM

From the analysis above, it can be seen, in order to solve the outcast phenomenon, we must make the congestion window of the data flows as consistent as possible. Now we

introduce a window size notification based algorithm TCP-CWR (TCP congestion window replacement). The senders send their congestion window size to the receiver when they finish the transmission of the current data block. The receiver calculates the average window size according to the size provided by the senders and sends back to each sender. The senders adjust their congestion window to the value calculated by the receiver. This would make all the senders have the same congestion windows when a new data block starts to send. The following discussion describes the process.

*1)*. After the transmission of the current data block, the applications running on the senders will send the size of the current congestion window to the receiver. Before the receiver gets this message, it does not request the next data block.

*2)*. When the receiver gets all the messages carrying the window size, it calculates the mean value of the congestion window. Then the receiver sends the new data block request packet attached with this value to the senders.

$$cwnd = (\sum_{i=1}^{N} cwnd_i) / N \tag{15}$$

*3)* After receiving this message, the senders set their congestion window value to the target value, and then send data packets normally.

When congestion occurs in the network and it results in the packet loss, we have adjusted the congestion window, flows with small RTT would not lose the entire unACKed packets. This means that they could enter into the fast retransmission phase, which won't affect performance seriously.

## V. SIMULATION RESULTS

We used the NS-2 simulation platform to do the simulation experiments. We set up a simplified topology which is shown in Fig. 2.

When we chose our experiment parameters, we strictly followed the parameters in Prakash's experiment. So we set 3 flows with RTT time 400 $\mu s$ , and 12 flows with RTT time 800 $\mu s$ .In the experiment, we observed their goodput and the number of timeouts with their RTT time. The results are shown in Fig. 3 and Fig. 4 .

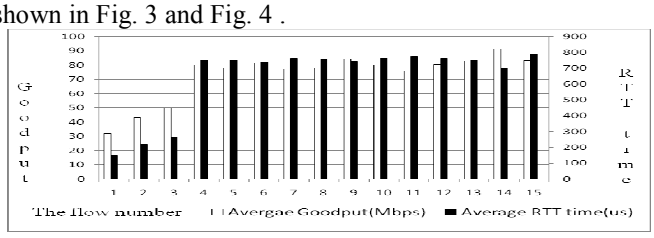


Fig. 3. The relationship between RTT and Goodput.

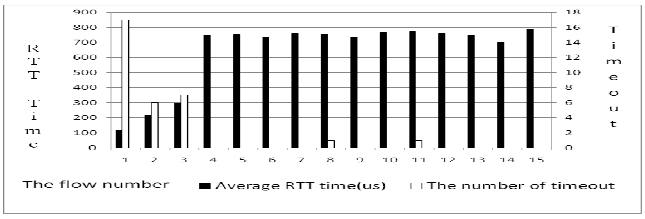


Fig. 4. The relationship between RTT and the number of timeout

From Fig. 3, we find that a quantitative relationship between goodput and RTT, i.e. the goodput of flows with small RTT is less than for flows with large RTT. This means the outcast phenomenon occurs in our simplified testbed. From Fig. 4, we also can see that flows with small RTT suffer 30 timeouts, while flows with large RTT only suffer 2 timeouts.

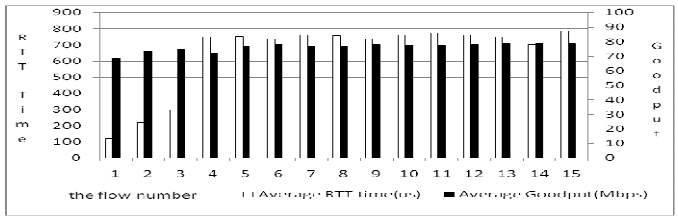


Fig. 5. The relationship between RTT and goodput.

From the results in Fig. 5, we can find that, after the adjustment of our algorithm, the goodput of flows with small RTT is almost the same with as flows with large RTT.

## VI. CONCLUSION

In this paper, we discussed the essential reasons for the outcast phenomenon. Through analysis using our simplified topology, we find that outcast is mainly because, when there are some packets getting lost, the flows with different RTT are affected differently. It has a more serious impact on the flows with small RTT, and the flows with large RTT are less affected. We then gave a mathematical model for goodput to further interpret the outcast phenomenon. Finally, we proposed window size notification based algorithm to mitigate the outcast phenomenon. With these experiments, we clearly validated our work.

## REFERENCES


[1] Arregoces M, Portolani M. Data Center Fundamentals. San Jose, Cisco Press, 2003:56-58.

[2] Guo C, Lu G, Dan L. BCube: A High Performance, Server-Centric Network Architecture for Modular Data Centers. ACM Special Interest Group on Communication, 2009:63-74.

[3] Al-Fares M, Loukissas A, Vahdat A. A Scalable, Commodity Data Center Network Archietecture. ACM Special Interest Group on Communication, 2008:63-74.

[4] Prakash P, Dixit A, Hu Y. The TCP Outcast Problem: Exposing Throughput Unfairness in Data Center Networks. 9th USENIX Conference on Networked System Design and Implementation, 2012:30-41.

[5] Zhang J, Ren F, Lin C. Modeling and Understanding TCP Incast in Data Center Networks. IEEE Conference on Computer Communication, 2011:1377-1385.

[6] Tam A, Xi K, Xu Y. Preventing TCP Incast Throughput Collapse at the Initiation, Continuation, and Termination. 20th International Workshop on Quality of Service, 2012:1-9.

[7] Floyd S, Henderson T, Gurtov A. RFC3782: The NewReno Modification to TCP's Fast Recovery Algorithm[S/OL]. www.ieee.org.

[8] Greenberg A, Lahiri P, Maltz D A, et al. Towards a next generation data center architecture: scalability and commoditization[C] //Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow. ACM, 2008: 57-62.