Don't Drop, Detour!

Matt Calder
Univ. of Southern California
calderm@usc.edu

Rui Miao
Univ. of Southern California
rmiao@usc.edu

Kyriakos Zarifis
Univ. of Southern California
kyriakos@usc.edu

Ethan Katz-Bassett
Univ. of Southern California
ethan.kb@usc.edu

Minlan Yu
Univ. of Southern California
minlanyu@usc.edu

Jitendra Padhye
Microsoft Research
padhye@microsoft.com

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Performance

Keywords

data center, buffers, packet loss

1. INTRODUCTION

Modern data center networks (DCNs) are built with shallow-buffered switches. These switches are cheaper than their deep-buffered counterparts, and they also reduce the maximum queuing delays packets may suffer. However, shallow buffers can lead to high packet loss under bursty traffic conditions (often caused by the "incast" traffic pattern [7]). DCTCP [3] attempts to solve this problem by using ECN markings to throttle the flows early, to avoid buffer overflow. However, DCTCP cannot prevent packet loss if the traffic bursts are severe, and short-lived. One extreme example is a large number of senders each sending one or two packets to a single receiver. No feedback-based congestion control protocol can prevent buffer overflow and packet losses under such extreme conditions. One reason for this is an assumption ingrained in today's DCN design.

The assumption is that when a switch needs to buffer a packet, it must do so only in its own available buffer. If its buffer is full, it must drop that packet. This assumption is so obvious that no one states it explicitly. We believe that in a data center environment this assumption is not warranted. All switches in a DCN are under a single administrative control. Thus, when faced with extreme congestion, the switches should pool their buffers into a large virtual buffer to absorb the traffic burst.

To share buffers among switches, we propose that a switch detour excess packets to other switches – instead of dropping them – thereby temporarily claiming space available in the other switches' buffers. We call this approach detour-induced buffer sharing (DIBS).

SIGCOMM'13, August 12–16, 2013, Hong Kong, China.
ACM 978-1-4503-2056-6/13/08.

The mechanics of DIBS are best explained assuming output-buffered switches, although it can work with any switch type. When a packet arrives at a switch input port, the switch checks to see if the buffer for the destination port is full. If so, instead of dropping the packet, the switch selects one of its other ports at random [1] to forward the packet on. Other switches will buffer and forward the packet, and it will make its way to its destination, possibly coming back through the switch that originally detoured it.

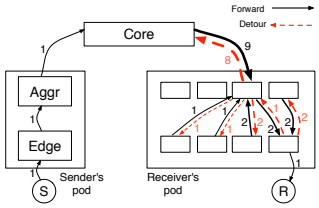

Figure 1: Example path of a packet detoured 15 times in a K=8 fat-tree topology. For simplicity, we only show 1 edge switch and 1 aggregation switch in the sender's pod, and we abstract the 16 core switches into a single node. The numbers near the arcs and the arcs' thicknesses both indicate the number of times the packet traversed that arc.

Figure 1 shows an illustrative example. It depicts the path of a single packet (from one of our simulations) that was detoured multiple times, before reaching destination R. The weight of an arc indicates how often the packet traversed that specific arc. Dashed arcs indicate detours. While the actual order of the hops cannot be inferred, we can see that the packet bounced 8 times back to a core switch because the aggregation switch was congested. The packet was bounced several times within the receiver pod before finally reaching the receiver.

The DIBS idea appears to violate basic principles taught in every Networking 101 class. However, our results show that DIBS works quite well as long as it is coupled with a **congestion control scheme**, and the network has some **spare capacity**.

Congestion control: DIBS is not a replacement for congestion control. Indeed, DIBS *must be paired with a congestion control scheme that does not rely on packet losses to infer congestion*. Otherwise, when faced with sustained congestion (e.g. multiple long-lived TCP flows converging to a single destination), DIBS will build up large queues. To avoid this problem, DIBS must be paired with a scheme like DCTCP [3]. With DCTCP, switches mark packets (using ECN bits) when the queue builds up beyond a certain threshold (typically, well before the queue is full). The end hosts react to these ECN mark by reducing their sending rate.

[1]We avoid ports whose buffers are full, and also those that are connected to end hosts.

Thus, DCTCP avoids long-term congestion, while DIBS ensures that there is no packet loss during large and transient bursts created by severe incast-like traffic. DIBS can also be paired with other congestion control schemes such as QCN [4] as well as the classic Random Early marking (RED/REM).

Spare capacity: If all links in the network are fully utilized, DIBS will not work, since there will be no capacity to handle the extra load caused by detoured packets. However, in DCNs, congestion is usually transient and local [5]. Thus, when we detour packets away from one congestion hotspot, the network will typically have capacity elsewhere to handle the traffic. We are working to derive a limit on network utilization, beyond which detouring can be detrimental.

We note that DIBS is particularly suited for deployment in DCNs. Many popular DCN topologies offer multiple paths [2], which detouring can effectively leverage. The link bandwidths in DCNs are very high and the link delays are quite small. Thus, the additional delay of a detour is quite low. Current DCN switches do not implement the random detouring, but the change required to do so is minimal. Indeed, our NetFPGA implementation is less than 25 lines of code. DIBS does not come into play until there is extreme congestion – it has no impact whatsoever when things are "normal".

2. DISCUSSION

Packet reordering: Detouring will cause packet reordering, which can adversely impact TCP's performance. A simple fix is to disable fast retransmission on end hosts.

CIOQ switches: Many modern switches use a combined input-output queued architecture (CIOQ), with shared buffer to store packets. DIBS can be implemented on these switches quite easily by defining a threshold. The switch fabric controller keeps track of packets on a per-port basis. When the number of packets buffered for an output port exceeds the threshold, the switch detours subsequent to other ports.

L2 routing: DIBS cannot work with self-learning L2 routing schemes such as the spanning tree protocol. However, most modern DCN topologies such as FatTree rely on pre-computed routing tables.

Rogue flows and congestion collapse: Rogue flows that do not implement congestion control can degrade performance of any network. DIBS may exacerbate this problem by constantly detouring packets of such flow. Today, DCNs use a variety mechanisms to detect and deal with such rogue flows. We expect that these mechanisms will be sufficient for our purposes as well. We are studying this issue in more detail. In the absence of such rogue flows, and as long as DIBS is paired with a congestion control protocol such as DCTCP, DIBS will not lead to congestion collapse.

Collateral damage: When congestion occurs at a particular switch, DIBS detours excess packets to other switches. These detoured packets interfere with other flows that would have otherwise not have been impacted by the traffic at the congested switch. We term this collateral damage. Our simulations show that collateral damage is low as long as a DIBS is used with a scheme like DCTCP and there is sufficient spare capacity. We are currently exploring the extreme conditions where collateral damage can be significant.

Routing loops: DIBS can appear to cause routing loops, as seen in Figure 1. However, these are merely temporary artifacts and do not affect actual routing.

Early results: We have implemented a preliminary version of DIBS in a NetFPGA router, in a Click modular router, and in NS-3. Our initial experiments are quite encouraging. Here we present only a sample result.

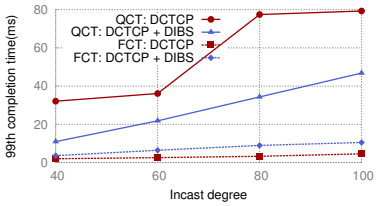

Figure 2: Mixed traffic: Variable incast degree. Compared to DCTCP alone, DIBS+DCTCP improves query completion time (QCT), with little collateral damage to background flows (FCT). (Background inter-arrival time: 120ms; query arrival rate: 300 qps; response size: 20KB)

Figure 2 shows performance of DIBS with a complex mix of traffic, derived from traffic traces from a large search engine [3]. The traffic consists of both query traffic, which causes incast flows, and background flows. As the degree of incast increases, the 99th percentile of query completion time (QCT) is much worse with DCTCP alone. DIBS improves performance by avoiding packet losses. At the same time, the collateral damage to background flows (FCT) is low, since the network has spare capacity.

Ongoing work: We are currently evaluating the performance of DIBS with a variety of traffic patterns using detailed simulations and experiments. We are also considering modifications to the basic DIBS concept such as limiting the number of times a packet can be detoured, probabilistically detouring, and priority-based detouring.

Related work: DIBS can be implemented alongside traffic spreading schemes such as ECMP (flow or packet level) or MPTCP [6]. These schemes reduce the possibility of extreme congestion, but do not eliminate it. Technologies such as Ethernet flow control and its modern variant, Priority Flow Control (PFC) [1], as well as the Infiniband link layer, guarantee loss-free L2 networks. Unlike these technologies, DIBS does not guarantee a loss-free L2 network. However, compared to these technologies DIBS is significantly easier to configure [1]. Also, unlike Ethernet flow control, DIBS is deadlock free [1]. We are exploring this relationship further. [7] proposed reducing TCP RTOmin value to mitigate the impact of packet losses. This proposal is orthogonal to DIBS, and the two can be implemented together. Centralizing scheduling and rate allocation [8] can *in theory* avoid all packet losses. However, in practice, such systems face scalability issues.

3. REFERENCES

[1] Priority flow control. `http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9670/white_paper_c11-542809.pdf`.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *SIGCOMM*, 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 *SIGCOMM*, 2010.

[4] M. Alizadeh, A. Kabbani, B. Atikoglu, and B. Prabhakar. Stability analysis of QCN: The averaging principle. In *SIGMETRICS*, 2011.

[5] S. Kandula, J. Padhye, and P. Bahl. Flyways to de-congest data center networks. In *HotNets*, 2009.

[6] C. Raiciu, S. Barré, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving datacenter performance and robustness with multipath TCP. In *SIGCOMM*, 2011.

[7] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. Andersen, G. Ganger, G. Gibson, and B. Mueller. Safe and effective fine-grained TCP retransmissions for datacenter communication. In *SIGCOMM*, 2009.

[8] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. H. Katz. DeTail: reducing the flow completion time tail in datacenter networks. In *SIGCOMM*, 2012.