Deadlocks in Datacenter Networks: Why Do They Form, and How to Avoid Them

Shuihai Hu[1,2] Yibo Zhu[1] Peng Cheng[1] Chuanxiong Guo[1] Kun Tan[1]
Jitendra Padhye[1] Kai Chen[2]
[1]Microsoft [2]Hong Kong University of Science and Technology

ABSTRACT

Driven by the need for ultra-low latency, high throughput and low CPU overhead, Remote Direct Memory Access (RDMA) is being deployed by many cloud providers. To deploy RDMA in Ethernet networks, Priority-based Flow Control (PFC) must be used. PFC, however, makes Ethernet networks prone to deadlocks. Prior work on deadlock avoidance has focused on *necessary* condition for deadlock formation, which leads to rather onerous and expensive solutions for deadlock avoidance. In this paper, we investigate *sufficient* conditions for deadlock formation, conjecturing that avoiding *sufficient* conditions might be less onerous.

1. INTRODUCTION

In this rather atypical hotnets submission we discuss a problem that is quite (c)old, albeit one that has re-emerged in a new context, and admit that we have no idea how to solve it completely. Our hope is to draw the community's attention to this problem, and re-ignite research in this area.

The problem is deadlock formation in lossless networks.

Driven by the need for ultra-low latency, high throughput and low CPU overhead, major cloud service providers are deploying Remote Direct Memory Access (RDMA) in their datacenter networks [17, 24]. Among the available RDMA technologies, RDMA over Converged Ethernet (RoCE) [1] is a promising one as it is compatible with current IP and Ethernet based datacenter networks.

The deployment of RoCE requires Priority-based Flow Control (PFC) [2] to provide a lossless L2 network. With PFC, packet loss can be avoided by letting a switch pause its immediate upstream device before buffer overflow occurs. However, PFC can cause deadlock problem. Deadlock refers to a standstill situation: there is a cyclic buffer dependency among a set of switches. Any switch in the cycle holds all the buffer needed by its upstream switch, and meanwhile is waiting for its downstream switch to release some buffer and resume its packet transmission. A simple scenario is illustrated in Figure 1.

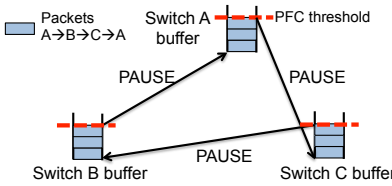


Figure 1: PFC-induced deadlock: simple illustration

It is easy to see that when deadlock occurs, no switch in the cycle can proceed. Further, throughput of the whole network or part of the network will go to zero due to the back-pressure effect of PFC pause.

It is often believed that such deadlocks cannot occur in clos-structured datacenter networks, since a loop cannot form in such networks with valley-free routing [24]. However, Guo et al. [10] has shown that deadlocks can indeed occur in such network. We now believe that deadlocks can also occur when transient loops form in clos structured networks. In our datacenters, these can happen as BGP[1] re-routes around link failures. In SDN-based datacenters, transient loops can occur during updates [12]. While transient loops will disappear by themselves soon, deadlocks caused by them are not transient. Deadlocks cannot recover automatically even after the problems (misconfiguration, failures/updates, etc.) that cause them have been fixed.

Hence some mechanism for handling the deadlock problem must be used when deploying RDMA in datacenter networks. These mechanisms fall in two broad categories. *Reactive* mechanisms/systems detect that a deadlock has formed, and then try to break it by resetting links/ports/hosts etc. These mechanisms are inelegant, disruptive, and should be used only as a last resort. We do not consider them further in this paper. *Proactive* deadlock prevention is a more principled approach to this problem. Prior work on deadlock prevention can be classified into two categories including 1) *Routing restriction-based approach* [7, 21]. The idea of this

[1]In our datacenters, we use BGP for routing, with each switch being a private AS.

HotNets-XV, November 09-10, 2016, Atlanta, GA, USA

 ISBN 978-1-4503-4661-0/16/11…$15.00
DOI: http://dx.doi.org/10.1145/3005745.3005760

approach is to ensure that no cyclic buffer dependency exists in the network by limiting the routing paths used in each priority class; 2) *buffer management (structured buffer pool) based approach* [8, 14]. This approach divides switch buffer into several buffer classes. A packet is allowed to access more buffer classes as it travels greater distance in the network. It can be proved that as long as the number of buffer classes is no smaller than the hop count of the longest routing path, there will be no cyclic buffer dependency.

Both approaches to deadlock prevention have some important drawbacks. Approaches based on routing restriction usually waste link bandwidth and limit throughput performance. Buffer management based approach may require a lot of priority classes and switch buffer for networks of large diameters, While the commodity switches with shallow buffer can support at most 2 lossless traffic classes [10]. These drawbacks can be viewed as the cost of eliminating cyclic buffer dependency in the network. While avoiding cycle buffer dependency guarantees a deadlock-free network, it may not be always feasible to pay these costs.

Thus, we take step back and ask: Is cyclic buffer dependency a necessary condition for deadlock formation, or is it a sufficient condition? If it is only a *necessary* condition, can we focus on *sufficient* conditions, and guarantee deadlock freedom, without eliminating cyclic buffer dependency?

To answer the above questions, we studied several representative deadlock cases. First, we find that cyclic buffer dependency is just a necessary condition for deadlock. In some sense, this is trivially true: if no flow is sending any data, there will obviously be no deadlock, regardless of cyclic buffer dependency. However, there are also several non-trivial cases where cyclic buffer dependency is met, all flows are active, but there is no deadlock. Second, we find that even if all the links in a switch cycle are paused simultaneously, deadlock may still not occur. These findings indicate that prior solutions are *too* conservative.

So, in this paper, we shall try to understand the *sufficient* conditions for deadlock formation, which we conjecture to be far easier to ameliorate than the *necessary* conditions. As mentioned earlier, we do not yet have a precise characterization of the sufficient conditions. Yet, we have made some headway, which allows us to sketch a few possible solutions to the problem.

2. DEADLOCK IN LOSSLESS NETWORK

We now briefly discuss how deadlocks form in lossless networks, and why we must study the necessary and sufficient conditions of deadlock formation.

Lossless Ethernet relies on PFC. RoCE needs PFC to provide a lossless L2 network. With the PFC PAUSE mechanism, a switch can pause an incoming link when its ingress buffer occupancy reaches a preset threshold. Properly tuned, no packet will be dropped due to insufficient buffer space. Unfortunately, deadlock may occur in such lossless networks.

PFC may lead to deadlock, if paused links form a cycle. In a PFC-enabled network, if a subset of links simultaneously paused by PFC happen to form a directed cycle, no packets in the cycle can move even if there is no more new traffic injected into this cycle.

To avoid such deadlock, *deadlock-free routing* [21] has been proposed. It guarantees that (if the routing configuration is correct,) any traffic does not cause deadlock.

Unfortunately, achieving deadlock-free routing is inefficient, and may not even be viable. Deadlock-free routing is achieved by eliminating cyclic buffer dependency [5]. However, ensuring that there is never any cyclic buffer dependency is challenging.

First, deadlock-free routing largely limits the choice of topology. For example, Stephens et al. [21] proposes to only use tree-based topology and routing, and shows that it is deadlock-free. However, there are a number of other datacenter topologies and routing schemes that are not tree-based [3, 9, 19], and do not have deadlock-free guarantee.

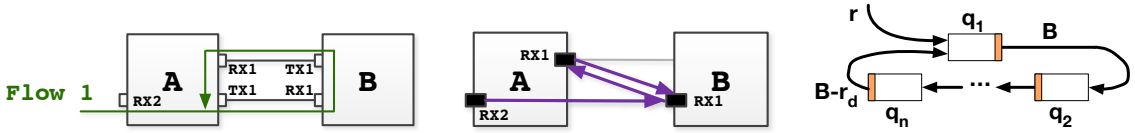
Second, due to bugs or misconfiguration, deadlock-free routing configuration may turn into deadlock-vulnerable. In fact, recent work has observed a PFC deadlock case in real-world tree-based datacenter [10], caused by the (unexpected) flooding of lossless class traffic. This case is a concrete example to show that even for tree-based topology, cyclic buffer dependency can still occur if up-down routing is not strictly followed. Furthermore, there are multiple reports of routing loops due to misconfiguration in today's production datacenters [23, 25]. If lossless traffic encounters any of these loops, cyclic buffer dependency is unavoidable.

In this paper, we argue that we should accept the fact that cyclic buffer dependency cannot be completely avoided,[2] and instead try to understand more precise deadlock conditions. Our findings show that even if there is cyclic buffer dependency, deadlock may not occur (see Section 3). This means that cyclic buffer dependency is only a necessary but not sufficient condition for deadlock. We show the occurrence of deadlock is affected by the packet TTL, the traffic matrix, as well as flow rate. Based on these findings, we propose several ways to avoid deadlock even in face of cyclic buffer dependency.

3. CASE STUDIES: CYCLIC BUFFER DEPENDENCY IS INSUFFICIENT FOR DEADLOCK

Although cyclic buffer dependency is a necessary condition for deadlock, it is not a sufficient condition. In this section, we present our case studies in which cyclic buffer dependency is present, but deadlock formation still depends on other factors. We demonstrate that 1) a looping flow that generates cyclic buffer dependency does not always lead to deadlock. The length of loop, flow rates and packet Time-To-Live (TTL) affects whether the deadlock forms. 2) Multiple flows may cause cyclic buffer dependency, but slightly

[2]In other words, deadlock-free routing may not always apply or work correctly.



(a) Topology and flows (b) Buffer dependency graph (c) Stable state model

Figure 2: Single looping flow creates cyclic buffer dependency but may not create deadlock.

different flow sets lead to different deadlock results. 3) Rate-limiting can prevent deadlock from happening.

3.1 Case 1: Flow Rate and TTL Determine Deadlock in a Routing Loop

A routing loop can create cyclic buffer dependency if a flow is trapped in the loop. The simplest example is a two-hop loop between two switches, as shown in Figure 2(a) (RX represents input queue (or port), and TX represents output queue (or port)). We then plot buffer dependency graph (Figure 2(b)). Each directed line represents a buffer dependency from the source RX to the destination RX. For example, packets buffered in $RX1$ of switch A will be sent to $RX1$ of B, and vice versa. So in Figure 2(b), two directed lines are drawn between A and B. Switch A's dependency on switch B means whether switch A can move the packets in its receiving buffer *RX1* to egress depends on switch B's buffer *RX1*.[3] The switches can send packets to the other side only when the other side's buffer utilization is under PFC PAUSE threshold. The deadlock happens when both of the involved buffers reach the PFC threshold at the same time and PAUSE the links.

However, this cyclic buffer dependency may not always turn into deadlock state. The flow rate, the TTL (time-to-live) of packets and the length of the loop together determine whether there will be deadlock. In our testbed, we run a simple experiment on two switches that are connected by a 40Gbps link and configured with a routing loop. All packets have initial TTL of 16 and are injected into one of the switches. We find that, only if the packet injection rate exceeds 5Gbps, there can form deadlock.

In order to analyze deadlock formation in the cases of routing loop, we develop a mechanism called *boundary state analysis*. It yields accurate prediction of whether deadlock forms, as shown below.

Boundary state analysis. On any of the switches in the loop, packets are injected by the previous hop and drained by the next hop. If the draining rate is smaller than the injecting rate, packets will continuously queue up in the switch buffers. In a loop, once one switch buffers enough packets and triggers PFC, the PFC will soon cascade through the whole loop and forms deadlock. We define *boundary state*, in which the injecting rate and draining rate are balanced on every switch, and any larger injecting rate leads to deadlock because draining rate cannot catch up.

We build the boundary state model as illustrated in Figure 2(c). The variables are described in Table 1.

Table 1: Stable state analysis variables

Variable	Description
r	Inject rate of new packets.
B	Link bandwidth.
r_d	Packets drain rate caused by TTL expiration.
TTL	Initial Time-To-Live value.
n	The length of the routing loop.

According to the boundary state definition, the injecting rate and draining rate must be equal on the first switch:

$$r + B - r_d = B \tag{1}$$

In addition, we consider the sum of TTL values of all packets in the system. During the boundary state, it should remain stable. Therefore, the increase rate and decrease rate of the sum of TTL should be the same:

$$n * B = TTL * r \tag{2}$$

Combining Equation 1, Equation 2, and the fact that deadlock requires larger injecting rate than that in boundary state, we derive the *necessary and sufficient condition* of deadlock in a routing loop scenario:

$$r > r_d = \frac{nB}{TTL} \tag{3}$$

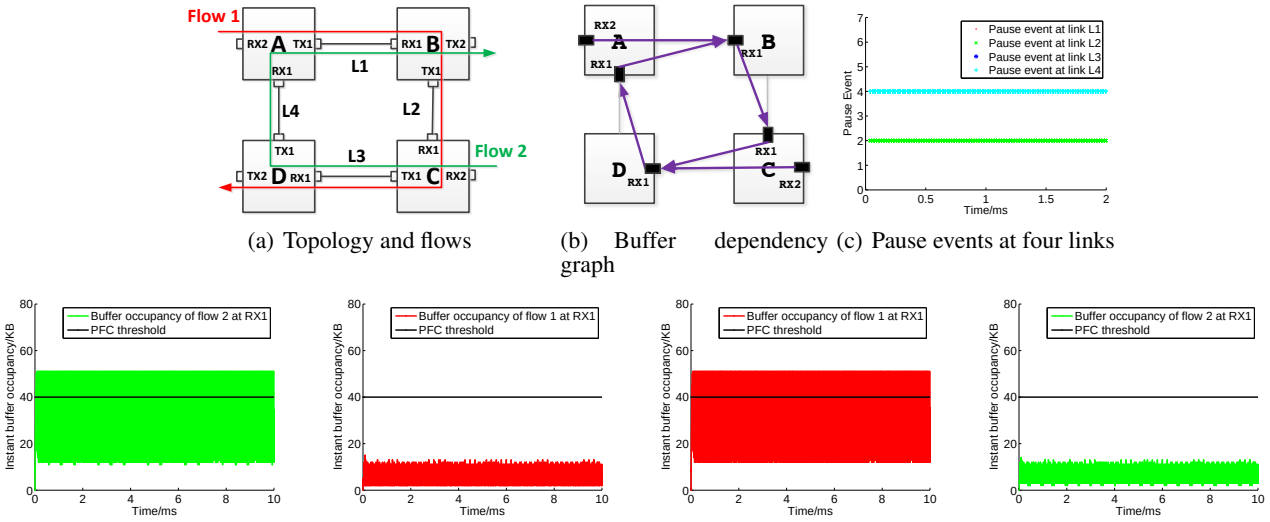
This matches what we observe in testbed experiment: with $B = 40Gbps$, $n = 2$ and $TTL = 16$, the flow injecting rate must be at least $5Gbps$ to cause deadlock. With larger bandwidth, shorter loop length or smaller initial TTL values, the threshold of r can be higher. As long as the flow rate is smaller than the threshold, no deadlock will form. As shown in Section 4, we may utilize this property to avoid deadlock, even if routing loop occurs.

3.2 Case 2: Traffic Matrix Affects Deadlock

Multiple flows may create a cyclic buffer dependency even if there is no routing loop. Figure 3(a) shows a simple example with four switches A, B, C and D. Flow 1 starts at a host (not shown) attached to A, passes through B and C, and ends at a host attached to D. Flow 2 starts at a host attached to C, passes through D and A, and ends at a host attached to B. Similar to the previous case, we can draw the dependency lines between switches. As shown in Figure 3(b), there is a cyclic buffer dependency among the four switches, i.e., dependencies from RX1 of A to RX1 of B, then to RX1 of C, then to RX1 of D, and finally back to RX1 of A.

In this example, the boundary state analysis does not yield meaningful results. Because the flows do not have any rate limiting, one can easily analyze that the stable throughput of each flow is $B/2$. However, it is not easy to tell whether

[3]We focus on receiving buffer because PFC PAUSE triggers based on the occupancy of receiving buffer.



(a) Topology and flows (b) Buffer dependency graph (c) Pause events at four links

(d) Buffer occupancy at switch A (e) Buffer occupancy at switch B (f) Buffer occupancy at switch C (g) Buffer occupancy at switch D

Figure 3: There is no deadlock even though two flows create cyclic buffer dependency among four switches.

deadlock will form. For example, one may suspect that switch A's $RX1$ will generate PFC PAUSE, and these PAUSE frames may cascade from A to D, then to C, B and finally back to A, thus creating deadlock. To understand such scenarios, we must analyze and simulate them at packet level.

Simulation setup: To create a well-controlled experimental environment, we simulate the scenario in Figure 3(a) using packet-level NS-3 simulations. In our NS-3 simulator, we implement the PFC protocol (*i.e.,* IEEE 802.1 Qbb protocol). For each ingress queue, the switch maintains a counter to track the bytes of buffered packets received by this ingress queue. Once the queue length exceeds the preset PFC threshold, the corresponding incoming link will be paused.

In our simulations, we configure static routing on all switches so that flow paths are enforced. Both flows are UDP flows with infinite traffic demand. Link capacity of all links is 40Gbps. All the switches have 12MB buffer. PFC threshold is statically set to 40KB for each ingress queue. These parameters affect how fast deadlock forms (if any), but do not affect whether deadlock forms.

In Figure 3(c), we plot the PFC pause events at four links L1, L2, L3 and L4. If link $Li, (i = 1, 2, 3, 4)$ is paused at time t, we plot a point at location (t, i). Pause events at different links are plotted with different colors and of different heights. As we can observe, links L2 and L4 are paused continuously, while the other two links L1 and L3 never get paused. In this case, deadlock will never form as no packet will be paused permanently.

To understand the pause pattern, we sample the instantaneous buffer occupancy of both flows at RX1 queues of A, B, C and D every 1us. In Figure 3(d), we draw the instant buffer occupancy of flow 2 at RX1 of A. Similarly, in Figure 3(e), Figure 3(f) and Figure 3(g), we draw the instant buffer occupancy of interested flows at RX1 queues of B, C and D, respectively. As flow 1 and flow 2 are symmetric, we only present the analysis for Figure 3(d) and Figure 3(e) to show why Link L4 is paused continuously but link L1 never gets paused. As shown in the figures, buffer occupancy of flow 2 at RX1 of A fluctuates between 10KB and 55KB around the PFC threshold, so link L4 will get paused intermittently. In contrast, buffer occupancy of flow 1 at RX1 of B is well below the PFC threshold (fluctuates between 0KB and 18KB), hence link L1 never gets paused.

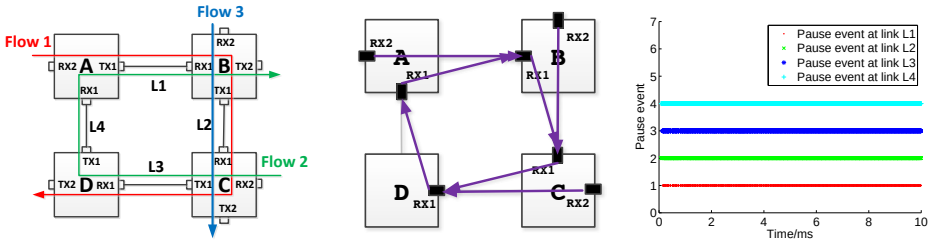
The takeaway is that, we cannot simply predict deadlock based on the existence of cyclic buffer dependency and flow-level stable state analysis. This is because we cannot predict the instantaneous buffer occupancy (and whether PFC is triggered) from flow-level analysis that only focuses on *average* flow throughput.

Slightly different traffic matrix leads to deadlock: as shown in Figure 4(a), based on the previous scenario, We add another flow (flow 3) which enters the network at switch B and leaves at switch C. All the three flows are UDP flows with infinite traffic demand. Buffer dependency graph is drawn in Figure 4(b). Compared with previous scenario, one additional dependency from RX2 of B to RX1 of C is added, but it is outside the cyclic buffer dependency. The cyclic buffer dependency itself remains unchanged.

Pause events at four links L1, L2, L3 and L4 are plotted in Figure 4(c). As we can see, in this case four links are all paused. To check whether deadlock will form in this case, we stop the three flows after a sufficient long period (1000ms). We find that pause events are continuously generated at all the four links even after three flows stop sending new packets. This means that a deadlock has been created among the four switches.

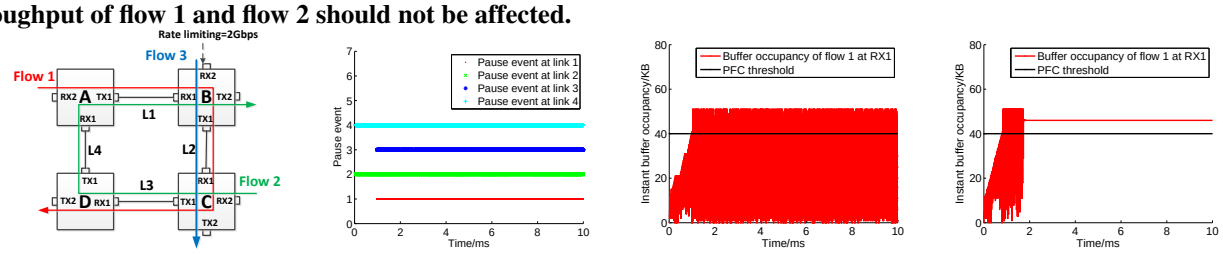
The bizarre thing is, if we apply the stable state flow analysis based on PFC fairness,[4] it is easy to see that all flows should have 20Gbps throughput. Particular at switch D, the

[4]PFC ensures per-hop per-port fairness. If packets from two ingress ports go to the same egress port, each ingress port gets half of the egress bandwidth.



(a) Topology and flows (b) Buffer dependency graph (c) Pause events at four links

Figure 4: Slightly different traffic matrix leads to deadlock, even though the flow-level analysis shows that the average throughput of flow 1 and flow 2 should not be affected.



(a) Topology and flows (b) Pause events at four links (c) 2Gbps rate limiter, no deadlock (d) 3Gbps rate limiter, deadlock

Figure 5: Different rate limiting determines whether the deadlock forms.

stable ingress and egress rate of flow 1 and flow 2 should remain the same as the previous case (Figure 3). However, now switch D starts to generate PFC at RX1 towards switch C, as opposed to no PFC generated in the previous scenario.

Flow-level stable state analysis cannot capture such behavior. We can only get answers from the packet-level analysis. Unfortunately, we have not yet found any analytic tools that can precisely describe the PFC behaviors in these two examples. Looking at the packet traces, we only roughly know that after adding flow 3, flow 1 has to share the bandwidth of link L2 with flow 3 and this may cause different PFC patterns on link L1 without affecting the average throughput of flow 1 and flow 2. But this change in pattern makes PFC cascade towards L4 and finally L3.

Once all the four links are paused simultaneously, there is a chance that no link can get resumed. For example, it is possible that when simultaneous pause happens, at switch A and switch B, the first packet buffered in the head is a packet of flow 1, and meanwhile, at switch C and switch D, the first packet buffered in the head is a packet of flow 2. Once this condition is met, PFC deadlock occurs.

Summary: In the above multi-flow scenarios, cyclic buffer dependency can be created without a routing loop. However, it is again not a sufficient condition for deadlock. The analysis of sufficient condition is complicated. Stable flow state analysis does not apply. A slightly different matrix that does not significantly affect stable flow state may lead to very different packet-level behavior, thus different deadlock results. Though packet-level simulations help us understand these scenarios, we so far do not find any analytic tools that are at packet-level and work for above examples.

3.3 Case 3: Rate Limiting Mitigates Deadlock

In the last deadlock example (Figure 4), if we additionally limit the rate of flow 3, deadlock may be avoided. As shown in Figure 5(a), we add a rate limiter to switch B's ingress port RX2. While the buffer dependency graph remains the same as Figure 4(b), slower flow 3 means that the congestion on switch B is reduced, PFC is less frequent and deadlock may be avoided. The question is, what is the maximum rate that can avoids deadlock?

Again, using flow-level stable state analysis, we cannot get the answer. We use packet-level simulator to test different rate limiting values. We find that when the rate of flow 3 is no more than 2Gbps, there is no deadlock even though all links have frequent PAUSE (Figure 5(b)). Note that, after zoom in Figure 5(b), we can see that four links are *never* paused simultaneously at packet level. Why is 2Gbps different from higher rate, like 3Gbps?

We plot the buffer occupancy of RX1 at switch B, and compare when we limit the rate of RX2 to 2Gbps (Figure 5(c)) and 3Gbps (Figure 5(d)). Interestingly, the buffer occupancy always fluctuates between 0 and a little above PFC threshold[5] with 2Gbps rate limiter. While with 3Gbps rate limiter, after some fluctuation the buffer goes into deadlock, even though the peak buffer usage is the same as 2Gbps case. Unfortunately, we cannot find any existing analysis tools that explain what we have observed.

In short, while rate limiting mitigates deadlock, packet-level analysis is required for understanding the actual threshold. We are currently working on analysis tools, *e.g.,* a fluid model that can describe PFC behavior, and will report it in future work.

Summary: From all the examples in this section, we summarize that cyclic buffer dependency is a loose condition for

[5]It takes some time for PFC PAUSE to arrive the other side and become effective after PFC threshold is reached. The switch buffers additional packets due to this delay.

deadlock. The traffic demand matrix, TTL and flow rates all affect the deadlock formation. While we cannot obtain the tightest condition (*i.e., necessary and sufficient condition*), we know that a tighter condition should include those factors, and that these factors can be utilized for deadlock mitigation. In Section 4, we discuss potential deadlock mitigations in addition to avoiding cyclic buffer dependency.

4. POTENTIAL DEADLOCK MITIGATIONS

Since cyclic buffer dependency is just a necessary condition for deadlock, there are mitigation mechanisms that avoid deadlock even if cyclic buffer dependency is present. The examples and analysis in Section 3 inspire us with some of the following potential deadlock mitigations.

TTL-based mitigation for deadlock caused by loops. In a routing loop, deadlock formation becomes less likely with smaller TTL (see Equation 3). Thus, the most straightforward mitigation is to reduce packets' initial TTL values. For example, in an N-hop routing loop, if the initial TTL is not larger than N, no deadlock will form because the deadlock threshold for r is B, which can never be exceeded.

In practice, we may not be able to guarantee that initial TTL values are always smaller than the size of the loop. However, by proper switch buffer management, we may make *class-specific* TTL much smaller than the actual TTL values. For example, if we assign packets that have different TTL values by at least X to different priority classes, the effective TTL becomes X within a priority class. Since PFC PAUSE operates based on priority classes, the deadlock threshold of injecting rate r is effectively increased.

In worst-case scenarios, the effective TTL may still be larger than the size of loop, meaning that some r smaller than B leads to deadlock. We may consider rate limiting to keep r below the threshold NB/TTL, as discussed below.

Rate limiting. Commodity switches support bandwidth shaping for each priority class or even particular flows. This can mitigate deadlock caused by both routing loops and multi-flow buffer dependency, as shown in Section 3. If we are able to predict the rate threshold for deadlock, we may bound the individual flow rate by that threshold on switches that are involved in cyclic buffer dependency. However, this requires intelligent rate limiting schemes to avoid over-punishing innocent flows. We leave this to future work.

Limiting PFC pause frames propagation: PFC is well known for its HoL blocking problem. The damage of HoL and the potential deadlock caused by PFC is significant because the pause frames are generated near the destination or in the middle of the network, where network congestion usually happen. Hence if we can limit the PFC pause frame propagation – or just generate them near the source, we can reduce the damage of both deadlock and HoL blocking.

Here we describe several possible ways of doing so: first, we can assign different PFC thresholds to the ports of a switch based on their position in the topology. Ports connecting to the downstream (i.e towards leaf) get smaller threshold, whereas ports connecting to the upstream get larger threshold. Second, we can use switches with larger threshold values at the higher tiers so that they can absorb small bursts instead of generating PFC pause frames. Third, again, we may classify packets with different TTL into different classes and assign them different PFC thresholds. Unfortunately, these solutions may lead to other issues including the unfairness between long (across different high tier switches) and short (*e.g.,* within the same rack) flows. This trade-off requires further study.

Preventing PFC from been generated. The recent transport protocols, DCQCN [24] and TIMELY [17] are designed to reduce the possibility of PFC generation. But due to the feedback latency of end-to-end delay, neither algorithm can detect congestion instantaneously, and thus they cannot completely prevent PFC from been generated.

One possible way to further reduce PFC is to integrate DCQCN together with phantom queuing, like [4]. By reacting to the phantom queues that assume lower link speed, congestion signals are generated much earlier.

Other possible mechanisms. In future work, a deeper understanding of tighter conditions for deadlock may lead to more deadlock mitigation mechanisms – this is the focus of our ongoing work.

5. RELATED WORK

RDMA in datacenters. RDMA has been used for improving distributed application performance, like in-memory key value store [6, 13, 16], Hadoop RPC [15] and HBase [11]. It has been recently deployed inside modern datacenters [10, 17, 24], based on RoCE (RDMA over Converged Ethernet), which relies on PFC to create a lossless Ethernet. Recent work [17, 24] discusses congestion control for RoCEv2 networks. The issue of deadlock is mentioned in these papers, but not directly addressed. In this paper, we aim to get deeper understanding on deadlock and possible mitigations.

Deadlock-free routing. To avoid deadlock in lossless networks, previous work [14, 18, 20–22] has focused on deadlock-free routing: i.e. deadlock freedom regardless of traffic pattern etc. It has also been proven that that eliminating cyclic buffer dependency is a necessary and sufficient condition for deadlock-free routing [5]. However, deadlock-free routing is difficult to implement in practice – since it is challenging to eliminate cyclic buffer dependency in face of arbitrary bugs and failures. Our work explores how we may control the flows, packet formats and switch configurations to avoid deadlock even if routing is not deadlock-free.

6. CONCLUSION

In this paper, we study the problem of deadlock in datacenter network. We demonstrate a few case studies in which deadlock formation depends on factors other than cyclic buffer dependency. In light of these examples, we discuss potential deadlock mitigation mechanisms including TTL-based schemes, rate limiting and reducing PFC propagation.

7. REFERENCES

[1] RDMA over Converged Ethernet (RoCE). http://www.mellanox.com/page/products_dyn?product_family=79.

[2] IEEE. 802.11qbb. Priority-based flow control, 2011.

[3] Hussam Abu-Libdeh et al. Symbiotic routing in future data centers. In *Proc. of SIGCOMM*, 2010.

[4] Mohammad Alizadeh et al. Less is more: Trading a little bandwidth for ultra-low latency in the data center. In *Proc. of NSDI*, 2012.

[5] W.J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. C-36(5):547 –553, may 1987.

[6] Aleksandar Dragojevic, Dushyanth Narayanan, Orion Hodson, and Miguel Castro. FaRM: Fast remote memory. In *Proc. of NSDI*, 2014.

[7] Jose Flich, Tor Skeie, Andres Mejia, Olav Lysne, Pierre Lopez, Antonio Robles, Jose Duato, Michihiro Koibuchi, Tomas Rokicki, and Jose Carlos Sancho. A survey and evaluation of topology-agnostic deterministic routing algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 2012.

[8] Mario Gerla and Leonard Kleinrock. Flow control: A comparative survey. *IEEE Transactions on Communications*, 1980.

[9] Chuanxiong Guo et al. BCube: a high performance, server-centric network architecture for modular data centers. In *Proc. of SIGCOMM*, 2009.

[10] Chuanxiong Guo et al. RDMA over commodity ethernet at scale. In *Proc. of SIGCOMM*, 2016.

[11] Jian Huang et al. High-performance design of hbase with rdma over infiniband. In *Proc. of IPDPS*, 2012.

[12] Xin Jin, Hongqiang Harry Liu, Rohan Gandhi, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Jennifer Rexford, and Roger Wattenhofer. Dynamic scheduling of network updates. In *Proc. of SIGCOMM*, 2014.

[13] Anuj Kalia, Michael Kaminsky, and David G. Andersen. Using rdma efficiently for key-value services. In *Proc. of SIGCOMM*, 2014.

[14] Mark Karol, S Jamaloddin Golestani, and David Lee. Prevention of deadlocks and livelocks in lossless backpressured packet networks. *IEEE/ACM Transactions on Networking*, 2003.

[15] Xiaoyi Lu, Nusrat S. Islam, Md. Wasi-Ur-Rahman, Jithin Jose, Hari Subramoni, Hao Wang, and Dhabaleswar K. Panda. High-performance design of hadoop rpc with rdma over infiniband. In *Proc. of ICPP*, 2013.

[16] Christopher Mitchell, Yifeng Geng, and Jinyang Li. Using one-sided rdma reads to build a fast, cpu-efficient key-value store. In *Proc. of ATC*, 2013.

[17] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. Timely: Rtt-based congestion control for the datacenter. In *Proc. of SIGCOMM*, 2015.

[18] Jose Carlos Sancho, Antonio Robles, and Jose Duato. An effective methodology to improve the performance of the up*/down* routing algorithm. *IEEE Transactions on Parallel and Distributed Systems*.

[19] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P. Brighten Godfrey. Jellyfish: Networking data centers randomly. In *Proc. of NSDI*, 2012.

[20] Tor Skeie, Olav Lysne, and Ingebjørg Theiss. Layered shortest path (lash) routing in irregular system area networks. In *Prof. of IPDPS*, 2012.

[21] Brent Stephens, Alan L. Cox, Ankit Singla, John Carter, Colin Dixon, and Wesley Felter. Practical dcb for improved data center networks. In *Prof. of INFOCOM*, 2014.

[22] Jie Wu. A fault-tolerant and deadlock-free routing protocol in 2D meshes based on odd-even turn model. *IEEE Transactions on Computers*.

[23] Hongyi Zeng, Shidong Zhang, Fei Ye, Vimalkumar Jeyakumar, Mickey Ju, Junda Liu, Nick McKeown, and Amin Vahdat. Libra: Divide and conquer to verify forwarding tables in huge networks. In *Proc. of NSDI*, 2014.

[24] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. Congestion control for large-scale RDMA deployments. In *Proc. of SIGCOMM*, 2015.

[25] Yibo Zhu, Nanxi Kang, Jiaxin Cao, Albert Greenberg, Guohan Lu, Ratul Mahajan, Dave Maltz, Lihua Yuan, Ming Zhang, Ben Y. Zhao, and Haitao Zheng. Packet-level telemetry in large datacenter networks. In *Proc. of SIGCOMM*, 2015.