Tagger: Practical PFC Deadlock Prevention in Data Center Networks

Shuihai Hu*(HKUST), Yibo Zhu, Peng Cheng, Chuanxiong Guo* (Toutiao), Kun Tan*(Huawei), Jitendra Padhye, Kai Chen (HKUST) Microsoft

CoNEXT 2017, Incheon, South Korea

* Work done while at Microsoft
RDMA is Being Widely Deployed

RDMA: Remote Direct Memory Access

- High throughput, low latency with low CPU overhead
- Microsoft, Google, etc. are deploying RDMA

RDMA NIC

Kernel bypass

Lossless Network (With PFC)
Priority Flow Control (PFC)

PAUSE upstream switch when PFC threshold reached

- Avoid packet drop due to buffer overflow
Due to Cyclic Buffer Dependency (CBD) A->B->C->A
Not just a theoretical problem, we have seen it in our datacenters too!
CBD in the Clos Network
CBD in the Clos Network

consider two flows initially follow shortest UP-DOWN paths
CBD in the Clos Network

due to link failures, both flows are locally rerouted to non-shortest paths
CBD in the Clos Network

these two **DOWN-UP bounced** flows create CBD

CBD: L2->S1->L3->S2->L2
Real in Production Data Centers?

Packet reroute measurements in more than 20 data centers:

~100,000 DOWN-UP reroutes!
Handling Deadlock is Important

• **#1:** transient problem $\rightarrow$ **PERMANENT** deadlock
  - Transient loops due to link failures
  - Packet flooding
  - ...

• **#2:** small deadlock can cause large deadlock
Three Key Challenges

What are the challenges in designing a practical deadlock prevention solution?

- No change to existing routing protocols or hardware
- Link failures & routing errors are unavoidable at scale
- Switches support at most 8 limited lossless priorities (and typically only two can be used)
The Existing Deadlock Prevention Solutions

- **#1: deadlock-free routing protocols**
  - not supported by commodity switches *(fail challenge #1)*
  - not work with link failures or routing errors *(fail challenge #2)*

- **#2: buffer management schemes**
  - require a lot of lossless priorities *(fail challenge #3)*

Our answer: **Tagger**
TAGGER DESIGN
Important Observation

**Takeaway:** In a data center, we can ask operator to supply a set of **expected lossless paths (ELP)**!
Basic Idea of Tagger

1. Ask operators to provide:
   - topology & expected lossless paths (ELP)

2. Packets carrying tags when in the network

3. Pre-install match-action rules at switches for tag manipulation and packet queueing
   - packets travel over ELP: lossless queues & CBD never forms
   - packets deviate ELP: lossy queue, thus PFC not triggered
Illustrating Tagger for Clos Topology

Root cause of CBD: packets deviate UP-DOWN routing!

ELP = all shortest paths (CBD-free)
Under Tagger, packets carry tags when travelling in the network
Initially, tag value = NoBounce
At switches, Tagger pre-install match-action rules for tag manipulation
Packet received by switch L3

tag = NoBounce

match-action rules installed at switches

Illustrating Tagger for Clos Topology
Illustrating Tagger for Clos Topology

tag = NoBounce  Bounced

flow 1

down-up bounce observed!

rewrite tag once DOWN-UP bounce detected
• S2 knows it is a **bounced packet** that deviates ELP → placed in the lossy queue
• No PFC PAUSE sent from S2 to L3 → buffer dependency from L3 to S2 removed
Tagger will do the same for packets of flow 2
2 buffer dependency edges are removed \( \rightarrow \) CBD is eliminated
What If ELP Has CBD?

ELP = shortest paths + 1-bounce paths

(ELP has CBD now!)
Segmenting ELP into CBD-free Subsets

Path segments before bounce
(only have UP-DOWN paths, no CBD)

Path segments after bounce
(only have UP-DOWN paths, no CBD)

two bounced paths are in ELP now
Isolating Path Segments with Tags

**tag 1** $\rightarrow$ path segments before bounce

**tag 2** $\rightarrow$ path segments after bounce
Isolating Path Segments with Tags

Adding a rule at switch L3: (Tag = 1, Inport=S1, OutPort = S2) -> NewTag = 2
No CBD after Segmentation

buffer dependency graph

CBD: \( L2 \rightarrow S1 \rightarrow L3 \rightarrow S2 \rightarrow L2 \)

packets with tag \( i \) \( \rightarrow \) \( i \)-th lossless queue
What If $k$-bounce Paths all in ELP?

**solution**: just segmenting ELP into $k$ CBD-free subsets based on number of bounced times!

**ELP** = shortest up-down paths + 1 bounce paths

- $k$-bounce paths
Summary: Tagger Design for Clos Topology

1. Initially, packets carry with tag = 1

2. pre-install match-action rules at switches:
   • DOWN-UP bounce: increase tag by 1
   • Enqueue packets with tag \( i \) to \( i \)-th lossless queue (\( i \leq k+1 \))
   • Enqueue packets with tag \( i \) to lossy queue(\( i > k+1 \))

For Clos topology, Tagger is optimal in terms of # of lossless priorities.
How to Implement Tagger?

- DSCP field in the IP header as the tag carried in the packets
- build 3-step match-action pipeline with basic ACL rules available in commodity switches
Tagger Meets All the Three Challenges

1. Work with existing routing protocols & hardware
2. Work with link failures & routing errors
3. Work with limited number of lossless queues
More Details in the Paper

- Proof of Deadlock freedom

- Analysis & Discussions
  - Algorithm complexity
  - Optimality
  - Compression of match-action rules
  - ...

Scenario: two flows forms CBD

Tagger avoids CBD caused by bounced flows, and prevents deadlock!
Evaluation-2: Scalability of Tagger

<table>
<thead>
<tr>
<th>Switches</th>
<th>Ports</th>
<th>Longest ELP</th>
<th>Lossless Priorities</th>
<th>Max Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>32</td>
<td>5</td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>500</td>
<td>64</td>
<td>6</td>
<td>3</td>
<td>76</td>
</tr>
<tr>
<td>1,000</td>
<td>64</td>
<td>6</td>
<td>3</td>
<td>88</td>
</tr>
<tr>
<td>2,000</td>
<td>64</td>
<td>7</td>
<td>3</td>
<td>98</td>
</tr>
<tr>
<td>2,000 (*)</td>
<td>64</td>
<td>7</td>
<td>4</td>
<td>135</td>
</tr>
</tbody>
</table>

* last entry includes additional 20,000 random paths.

Match-action rules and priorities required for Jellyfish topology

Tagger is scalable in terms of number of lossless priorities and ACL rules.
Evaluation-3: Overhead of Tagger

Tagger rules have no impact on throughput and latency
Conclusion

• Tagger: a tagging system guarantees deadlock-freedom
  – **Practical:**
    - require no change to existing routing protocols
    - implementable with existing commodity switching ASICs
    - work with limited number of lossless priorities
  – **General:**
    - work with any topologies
    - work with any ELPs
Thanks!