Cycle Identification

Yizhou Li (Huawei)  
Guanhua Zhuang (Huawei)  
Jiang Li (Huawei)  
Jeong-dong Ryoo (ETRI)  
Li Dong (Shenyang Institute of Automation)  
Wenbin Dai (Shanghai Jiao Tong University)
Introduction

• 802.1Qdv proposes to
  1. Store received frames into multiple cyclic bins based on the time of reception of the frame
  2. Bins are drained in rotation manner at a fixed interval (i.e. cycle/cycle time in the slides)

• Goal of the slides
  • Show the goal to improve the bandwidth utilization in small cycle
  • Discuss the cycle ambiguity problem when making dead time (DT) minimum to improve the utilization
  • Propose to use cycle id based determination in addition to time based one in 802.1Qdv
  • Provide three options to carry cycle id
DT (dead time) is the key to absorb the time variation

- DT (Dead time): time imposed in a cycle to ensure that the last byte sent in node A’s cycle x is fully received and ready to be sent at the start of node B’s cycle y, where cycle y is usually the earliest available cycle to meet such a requirement.

- DT (*) = output delay at node A + link delay + preemption delay + processing delay at node B.
  - Link delay will not contribute to DT if node B’s receiving side offsets the cycle start time by link delay.

- DT << T, so DT is negligible conventionally.
  - Cycle time T is normally ~100x μs.
  - DT is ~10x μs.
  - DT/T < 5% generally.

- Frame reception time at node B determines to which bin the frame should be put.

(**): draft-ietf-detnet-bounded-latency
Utilization unacceptably low in small cycle

- E2e latency and jitter is proportional to cycle time $T$ and #of hops $h$
  - E2e delay $\approx h * T$; jitter = $2T$

- Desire to use smaller $T$ for better latency and jitter
  - To achieve e2e latency (e.g. < 1ms) or large # of hops (e.g. 10+)
  - A known app cycle 31.25 $\mu$s; cycle $T < 31.25$ would be desired
  - Should support $T$ in the order of $\sim 10x \mu s$
  - DT is in the order of $\sim 10x \mu s$ as well
  - DT and $T$ are in the same order then

- Utilization decreases when cycle time $T$ decreases
  - DT eats $T$, e.g. $\approx 50\%$ when $T=40 \mu s$ & DT=20 $\mu s$

Cycle $T_1 = 100x \mu s$  \hspace{1cm} $T_2 = 10x \mu s$  \hspace{1cm} Unacceptably low utilization
Hardware and Software frame reception time used for output bin determination

**Hardware reception time**
- Timestamp every data frame with the 1st bit reception time at phy layer
- Output delay at node A contributes to time variation
- Each frame offset by link delay at minimum
- Not always available in practice

**Software reception time**
- Time at the moment that the frame starts the processing
- Generally available in implementations, e.g. high mac layer in programmable NP
- Frames in a cycle experience the variable delay in Node B.

**Why?**
- Store the variable frame size
- Variable residence time before frame started being processed, e.g. by PBA (packet bus arbiter) or buffering

**Dead time imposed at each cycle**
- Time window for sending or receiving frames (not a single frame)

**Node A** (transmitter)
- Cycle time T
  - T1, T2, T3, T4
- Link delay

**Node B** (receiver, i.e. hardware reception time)
- Cycle time T
  - T1, T2, T3, T4

**Node B** (processing, i.e. software reception time)
- Cycle time T
  - T1, T2, T3, T4
- Time at the moment that the frame starts the processing
- Generally available in implementations, e.g. high mac layer in programmable NP
- Frames in a cycle experience the variable delay in Node B.

**Why?**
- Store the variable frame size
- Variable residence time before frame started being processed, e.g. by PBA (packet bus arbiter) or buffering

**The following slides uses the software reception time to determine to which bin the frame should be put at node B**
Goal - Improve the utilization in small cycle $T$

- **Why low utilization?** DT is relatively too large.
- **A straightforward way to improve utilization:** make DT minimum
  - Absorb only the preemption delay instead of the full time variation, i.e. curve out output delay and processing delay
- **A remaining problem:** cycle ambiguity in reception time based bin determination
Propose to use the explicit cycle identification

- Carry cycle id and change per hop
- Use cycle id based output bin determination instead of time based
- Remove the ambiguity
- Achieve the good utilization in small cycles by making DT minimum
How to carry a cycle id

1. R-tag (defined in 802.1CB)

Define a subtype flag and use the last 4-bit in Reserved field for cycle id.

<table>
<thead>
<tr>
<th>Ethertype (F1-C1)</th>
<th>Reserved (16-bit)</th>
<th>Sequence number (16-bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>flag(1)</td>
<td>Rsvd (11)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ethertype (F1-C1)</th>
<th>Reserved (16-bit)</th>
<th>Sequence number (16-bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1-C1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How to carry a cycle id (cont’d)

2. Define a new cycle-tag

<table>
<thead>
<tr>
<th>Ethertype (cycle-tag)</th>
<th>Subtype (4-bit)</th>
<th>Reserved (4-bit)</th>
<th>Cycle ID (8-bit)</th>
</tr>
</thead>
</table>
How to carry a cycle id (cont’d)

3. Use vlan stacking + vlan mapping function
   • Inner vlan is used for cycle id, use ACL to map from ingress cycle id to egress cycle id
   • Outer vlan is used as normal vlan based mac learning and forwarding
   • Used in a controlled domain, may not be compatible with some existing s-vlan + c-vlan usage

<table>
<thead>
<tr>
<th>Ethertype (s-vlan)</th>
<th>vlan-tag (16-bit)</th>
<th>Ethertype (c-vlan)</th>
<th>Cycle id (16-bit)</th>
</tr>
</thead>
</table>

ACL example: if-match cvlan-id cycle-id-in
remark cvlan-id cycle-id-out
backup
Cycle id used in the extension to traditional 2-buffer CQF

- Demand to have the restricted extension to traditional 2-buffer CQF usage
- Re-use traditional CQF as much as possible
- Nodes sync their cycles as usual, gate open/close controlled by sync’ed time
- Utilization gets worse as DT can not curve out the link delay

**Restricted extension:**
- Use 3 buffer rotation with the smaller buffer space each
- Set cycle T smaller, e.g. half of the original in pic above
- Cycle id based determination works well to solve the ambiguity in this case and improve the utilization effectively
Procedures when using frame 1st bit hardware timestamping

- Time variation before hardware timestamping has to be put into DT, e.g. output delay, clock/timestamp accuracy
- Time variation after hardware timestamping has no more impact
- Check timestamp for each frame at processing phase to determine the output bin in order to remove cycle ambiguity
- Requires:
  - Native hardware support. Can not be done in programmable NP.
  - Time variation before timestamping has to be negligible since it contributes to DT.
    - DT here is always larger than the value used in cycle-id case since it can not curve out this part of variation.