Further Details on Cycle Identification

Yizhou Li (Huawei)
Guanhua Zhuang (Huawei)
Shoushou Ren (Huawei)
Jiang Li (Huawei)

Jeong-dong Ryoo (ETRI)
Li Dong (Shenyang Institute of Automation)
Wenbin Dai (Shanghai Jiao Tong University)
Recap and Goal

• **dv-yizhou-cycle-identification-0922-v02** in September meeting
  • 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

• Slides 3-7 recaps the basic concept of using cycle id. Please refer to the previous slides for more details if needed.

• Goal of this deck of slides is to answer the questions received:
  • How to initialize the cycle id based system?
  • How to compute cycle and buffer IDs as they have different ID spaces?
  • How the mapping works when the #of bins(buffers) are different on two neighbor nodes?
  • How large should the cycle ID space be? What are the considerations when choosing the field length of it?
Goal - Improve the utilization in small cycle T

- **Why low utilization?** DT is relatively too large to a cycle.
- **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)

<table>
<thead>
<tr>
<th>Ethertype (F1-C1)</th>
<th>Reserved (16-bit)</th>
<th>Sequence number (16-bit)</th>
</tr>
</thead>
</table>

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>
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>
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
Further details on Cycle id

• Conceptually Cycle id = timestamp of cycle start time of the frame’s transmission buffer on upstream node’s output port.

• Try to reuse terms defined in new-finn-multiple-CQF-0921-v02 as much as possible.
Some assumptions in this deck to simplify the illustration

• Only one priority class uses Qdv mechanisms at a physical port.
• The configuration and calculation is for that priority.
• Cycle time $T_c$ remains the same for the same priority in the domain.
• The slides provide a practical way in implementation as example, but it is not the only way.
Terms – buffer related

• n: the number of physical ports. Range: 0 ~ n-1

• $B_{i,o}$: the number of buffers required for the input and output port pair $(i, o)$

• $B_o$: the number of physical output buffers on port o.
  
  $B_o = \text{Max}(B_{i,o}), \ i = 0 ~ n-1$ on a given port o

• buf_id:
  
  • id assigned to the physical buffer in which a frame should be placed at output port o when the frame is processed.
  
  • $buf_id \in [0, B_o -1]$
Terms – latency related

- **TV**: sum of the frame-based time variation
  - Include output delay, processing delay and link delay
  - Link delay is the first-bit-out to last-bit-in delay that varies with packet size
- **TV\textsubscript{i,o}**: TV value for the input and output port pair \((i, o)\).
- Calculate \(B_{i,o}\) as follows:

\[
B_{i,o} = \text{floor}(\frac{TV_{i,o}}{T_c}) + 4 \quad --- \quad \text{baseline}
\]
- Max receiving window time interval \(T_c + TV_{i,o}\)
- # of receiving buffer to accommodate the max receiving window time:
  \[
  \text{floor}\left(\frac{\left(T_c + TV_{i,o}\right)}{T_c}\right) + 2 = \text{floor}\left(\frac{TV_{i,o}}{T_c}\right) + 3
  \]
- # of sending buffer \(= 1\)
- Hence, \(B_{i,o} = \) receiving buffer + sending buffer
  \[
  = \text{floor}\left(\frac{TV_{i,o}}{T_c}\right) + 4
  \]
- Special cases:
  - use one less buffer, i.e. \(B_{i,o} = \text{floor}\left(\frac{TV_{i,o}}{T_c}\right) + 3\)
    - rev window spans over 2 cycles in case of lucky or intentional cycle phase shift between A & B. We assume A&B independently run cycles here, hence always use at least the baseline calculation
  - Use more buffers, i.e. \(B_{i,o} = \text{floor}\left(\frac{TV_{i,o}}{T_c}\right) + \frac{5}{6}/.../n\)
    - Introduce intentional extra delay, e.g. to balance replicated frames over diff paths
Terms – cycle related

- **cycle_id**: cycle id assigned to a frame when the frame is placed to a buffer with buf_id at an output port o
  - Max value of cycle_id C is limited by the field length, e.g. if length is 4, then C = 16, so that cycle_id ∈ [0, 15]
  - C is the same for all ports on all nodes in a domain
  - Increments (modulo C) each cycle time

- **N**: the least common multiple over all $B_o$ and C in a node. Value of N can be different on the nodes.

- **$S_i$**: Each input port i assigns each received frame a logical buffer selector $S_i$, which is an integer in the range 0 through N-1, and which increments (modulo N) each cycle. The value of $S_i$ is $s_i$.
  - $s_i$ is directly used by data frames, not for (cycle) mapping determination frames.

Note: All values are integers
4 steps in initialization and determination in the following slides

• Step 1: Base parameter provisioning
• Step 2: Initialization of buffer selector and output buffers
• Step 3: Determination of cycle id mapping relation for port pair (i,o)

• Step 4: Determination of output cycle id and buffer id for data frame
  • This step is for real data traffic, not part of initialization and provisioning.
Example Step 1—base parameter provisioning

- The node has n=8 port.
- Cycle time $T_c$ is set and time variation $TV_{i,o}$ for port pair $(i,o)$ is known.
- Calculate $B_{i,o} = \text{floor}(TV_{i,o} / T_c) + 4$ and $B_o$ is the max($B_{i,o}$) for a port $o$.
- Assume the calculation results of the number of physical buffers $B_o$ on each port:
  - $B_5 = B_7 = 6$, $B_6 = B_8 = 4$
  - $B_1 – B_4$ not shown, assume all are 4
- Assume Max # of cycle_id $C = 8$, cycle_id $\in [0, 7]$
  - i.e. length of cycle_id = 3 bit. (This is for picture simplicity. In reality, it should be larger)
- Then $N = 24$ (the least common multiple of 4,6,8)
- Logical buffer selector $S_i$ makes value $s_i$ rotate between 0 – 23.
  - Note: It is the index assignment for calculation simplicity. No real buffer attached.
Example Step 2 – initialization of buffer selector and output buffers

- All the ports of a node use the same system time T. T is the time elapsed (in nanosecond) from the latest system startup. All the components in the node can use T as a time source.

- **System Synchronized Initialization:**
  - Initialize the starting time for $S_i$ and $buf_id$ on all ports to be a multiple of $N*Tc$.
  - Logically make the position and shifting for all $S_i$ the same, thus a single selector $S$ can be used to simplify the implementation. Let $s$ be the value of the selector $S$.
  - Facilitate halfway port enabling
  - Initialize $s = 0$ and the transmitting buffer id $= 0$ on all ports
  - Increments (modulo $N$) each cycle
  - At any time $T$, the following computations hold
    - $s = (T \mod (N*Tc)) / Tc$ ---- “/” is floor division
    - buffer id currently transmitting frames on port $o$:
      - $Tx$-buf-id = $s \mod B_o$
    - cycle id used by currently transmitting buffer:
      - $Tx$-cycle-id = $s \mod C$
Example Step 3 – Determination of cycle id mapping relation for port pair (i,o)

Pre-requisite:
• Node A & B have run their cycles independently with cycle $T_c$

Purpose: For the input and output port pair (i, o) on B, determine mapping parameter $M_{i,o}$ so that a stable cycle mapping equation $cycle\_id\_out = (cycle\_id\_in + M_{i,o}) \mod C$ can be used for future data frames over port pair (i,o) no matter what time variation are experienced by that frame.

Mapping determination frame: special frame to determine the cycle mapping relation between two neighbors during system initialization and auditing. It experiences the least time variation
• Shortest frame – 64B
• Highest priority

Cycle mapping relation computation example:
• B receives the mapping determination frame from port 2
• Assume B’s system time is T. Compute the potential worst case $s$ for the frame over each output port $o$ as:
  $s_o = ((T + TV_{2,o}) \mod (N*T_c)) / T_c$
• Compute cycle_id_out for the potential worst case frame over each output port $o$ as:
  $cycle\_id\_out_o = (s_o + 1) \mod C$
• Compute $M_{i,o} = (cycle\_id\_out_o - cycle\_id\_in_i + C) \mod C$
  • Assume the frame carries cycle_id_in = 7 from the incoming port 2 and the computed cycle_id_out is 2 on output port 5, then $M_{2,5} = (2-7+8) \mod 8 = 3$
  • $M_{i,o}$ can be simplified to $M_i$ for a port $i$ if $M_i$ takes Max($M_{i,o}$) for a specific $i$. That will make the # of mapping parameters O(n) instead of O($n^2$), n=# of ports.
Example Step 4 – Data frames: determination of output cycle id and buffer id

**Output cycle id:**
- \(\text{cycle}_\text{id}_\text{out}_o = (\text{cycle}_\text{id}_\text{in}_i + M_{i,o}) \mod C\)
- e.g. as \(M_{2.5} = 3\) for port pair (2,5), the cycle_id determination is defined as \(\text{cycle}_\text{id}_\text{out} = (\text{cycle}_\text{id}_\text{in} + 3) \mod 8\)

**Output buffer id:**
- Cycle id used by currently transmitting buffer: \(\text{Tx-cycle-id} = s \mod C\)
- Compute the id offset between targeting placement buffer and current transmitting buffer as: \(\text{offset} = (\text{cycle}_{\text{id}}_\text{out}_o - \text{Tx-cycle-id} + C) \mod C\)
- Compute the \(\text{buf}_\text{id}_o\) on port \(o\) to place the frame as: \(\text{buf}_\text{id}_o = (s + \text{offset}) \mod B_o\)
Summary – on each node, for each priority

1. Set cycle time $T_c$ and time variation $TV_{i,o}$ for port pair $(i,o)$
2. Calculate number of physical buffers $B_o$ on each port $o$ as:
   - $B_{i,o} = \text{floor}(TV_{i,o} / T_c) + 4$
   - $B_o = \max(B_{i,o})$ for port $o$
3. Calculate $N = \text{the least common multiple of } B_o$ and $C$
4. Start system synchronized initialization on every node for buffer selector $S$ and ports
5. Use Mapping Determination Frame to calculate the mapping relation value $M_{i,o}$ for port pair $(i,o)$ as follows:
   - $s_o = \left[\left((T + TV_{i,o}) \mod (N*T_c)\right) / T_c\right]$
   - cycle_id_out_o = $(s_o + 1) \mod C$
   - $M_{i,o} = (\text{cycle_id_out}_o - \text{cycle_id_in}_i + C) \mod C$
6. For an incoming frame for port pair $(i,o)$, calculate the output cycle id $\text{cycle_id_out}_o$ and output buffer id $\text{buf_id}_o$
   - output cycle id: $\text{cycle_id_out}_o = (\text{cycle_id_in}_i + M_{i,o}) \mod C$
   - output buffer id:
     - $Tx$-cycle-id = $s$ mod $C$
     - offset = $(\text{cycle_id_out}_o - Tx$-cycle-id + $C) \mod C$
     - $\text{buf_id}_o = (s + \text{offset}) \mod B_o$

One-time static calculation and provisioning

Step 1: Base parameter provisioning
Step 2: Initialization of buffer selector and output buffers
Step 3: Determination of cycle id mapping relation
Step 4: Data frame mapping

for each data frame
Choice of cycle id length

- Number of cycle id $C = 2^L$ where $L$ is the length in bits of cycle id.
- $C$ should be larger than the number of physical buffer $B_o$ on any port in step 1 base parameter provisioning.
- Discuss: $L$ to be at least 5?
  - $C = 32$ ($L = 5$) should be sufficiently large as it can support a 32-buffer port.
  - If we make $L=8$, that would allow 256-buffer on any port.