Cycle Identification

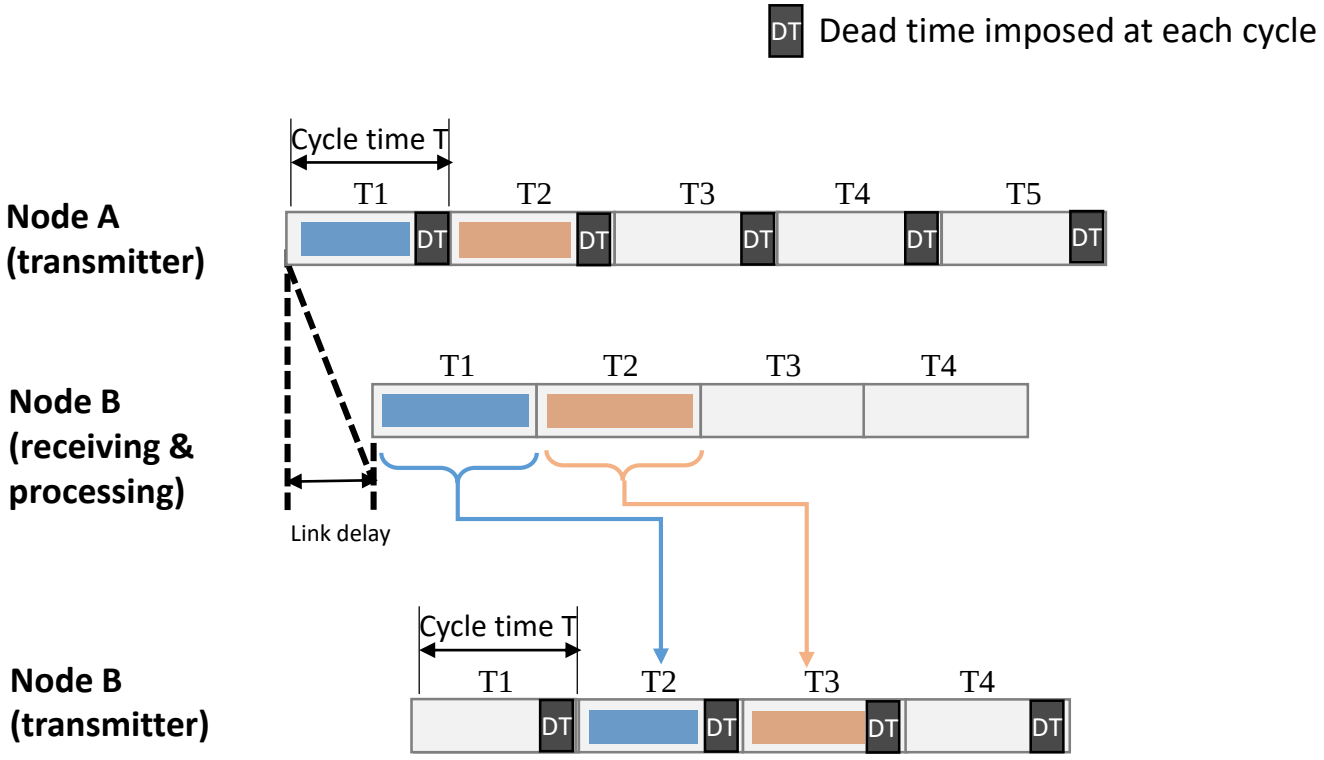
Yizhou Li (Huawei)
Guanhua Zhuang (Huawei)
Jiang Li (Huawei)
Jeong-dong Ryoo (ETRI)
Li Dong (Shenyang Institute of Automation)
Wenbin Dai (Shanghai Jiao Tong University)

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' 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

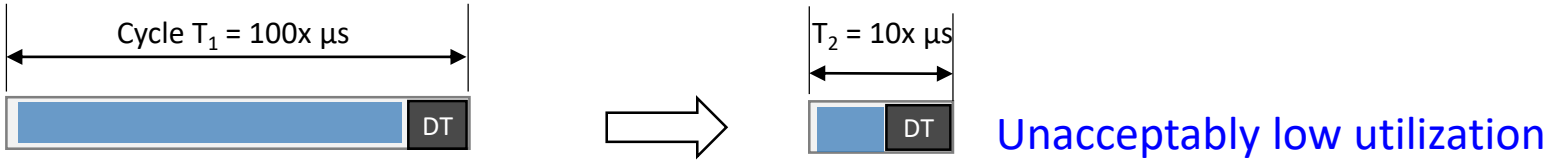
(*): draft-ietf-detnet-bounded-latency

Utilization unacceptably low in small cycle

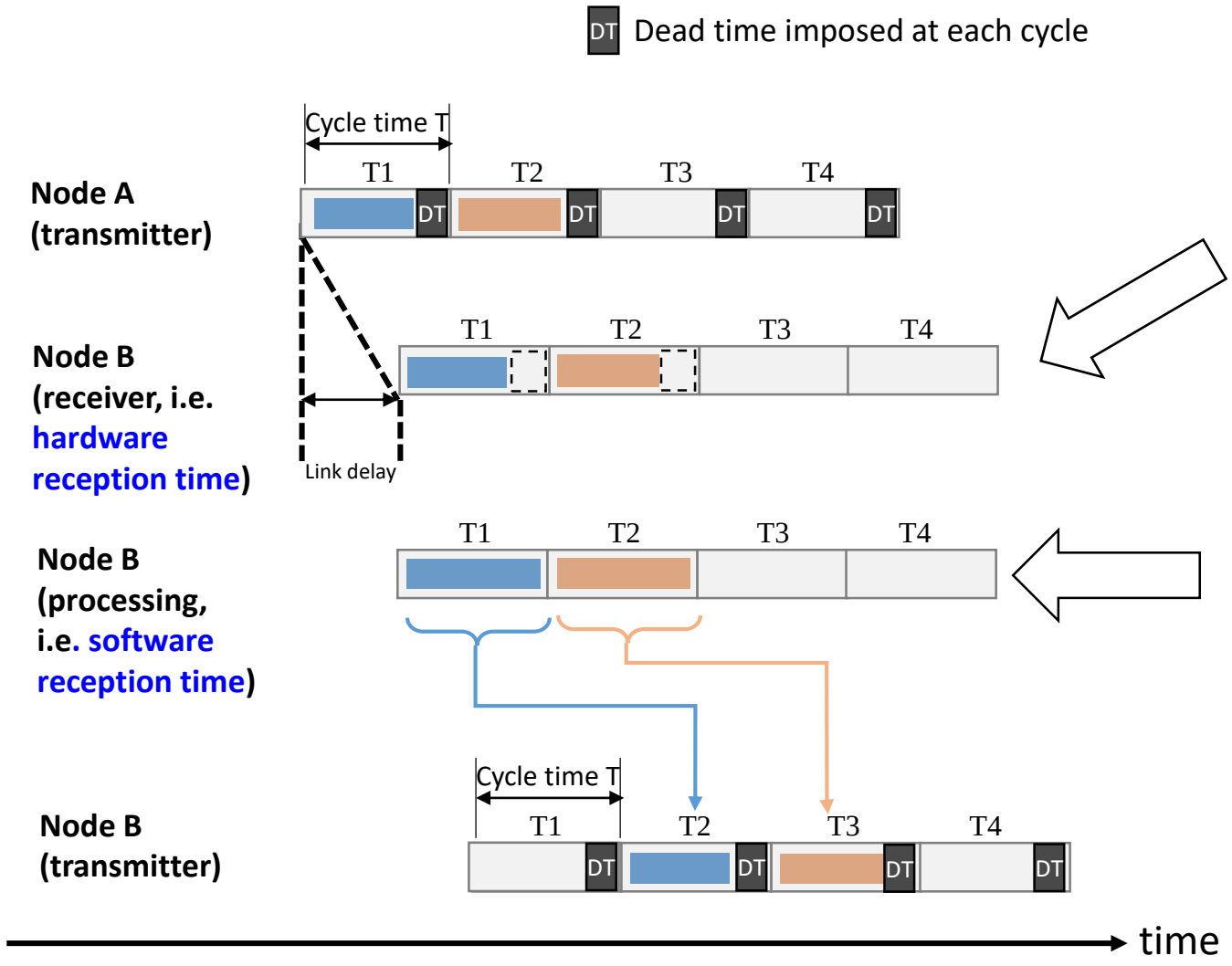
- E2e latency and jitter is proportional to cycle time T and #of hops

- 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 µs; cycle T < 31.25 would be desired
  - Should support T in the order of ~10x µs
  - DT is in the order of ~10x µs as well
  - DT and T are in the same order then

- Utilization decreases when cycle time T decreases
  - DT eats T, e.g. ≈50% when T=40 µs & DT=20 µs



Hardware and Software frame reception time used for output bin determination



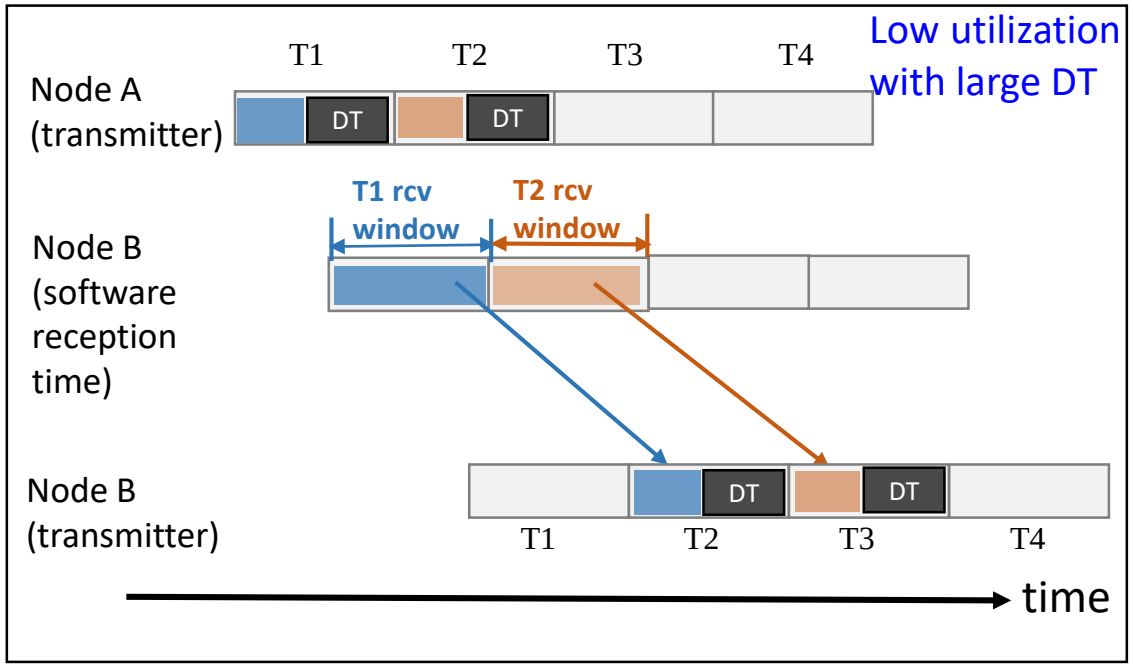
Hardware reception time

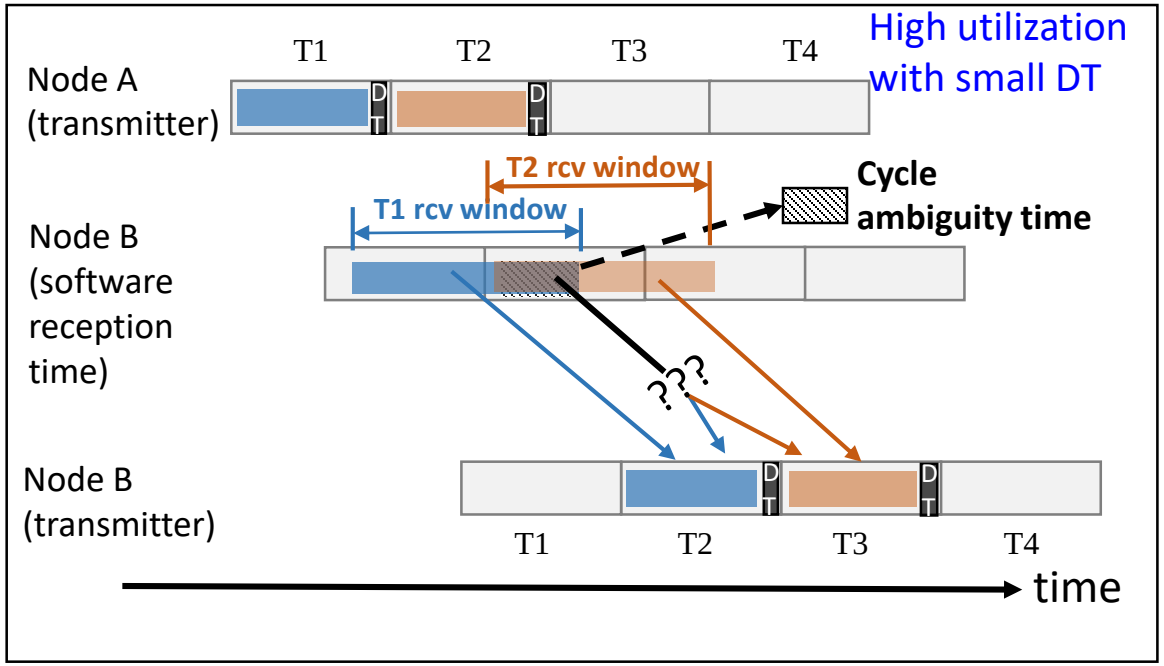
- Timestamp every data frame with the 1st bit reception time at phy layer
- Each frame offset by link delay invariably
- Not always available in practice

Software reception time

- Time at the moment that the frame starts the receiving 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 time before frame started being processed, e.g. by PBA (packet bus arbiter)
- used as frame **reception time** in the following slides to determine to which bin the frame should be put at node B

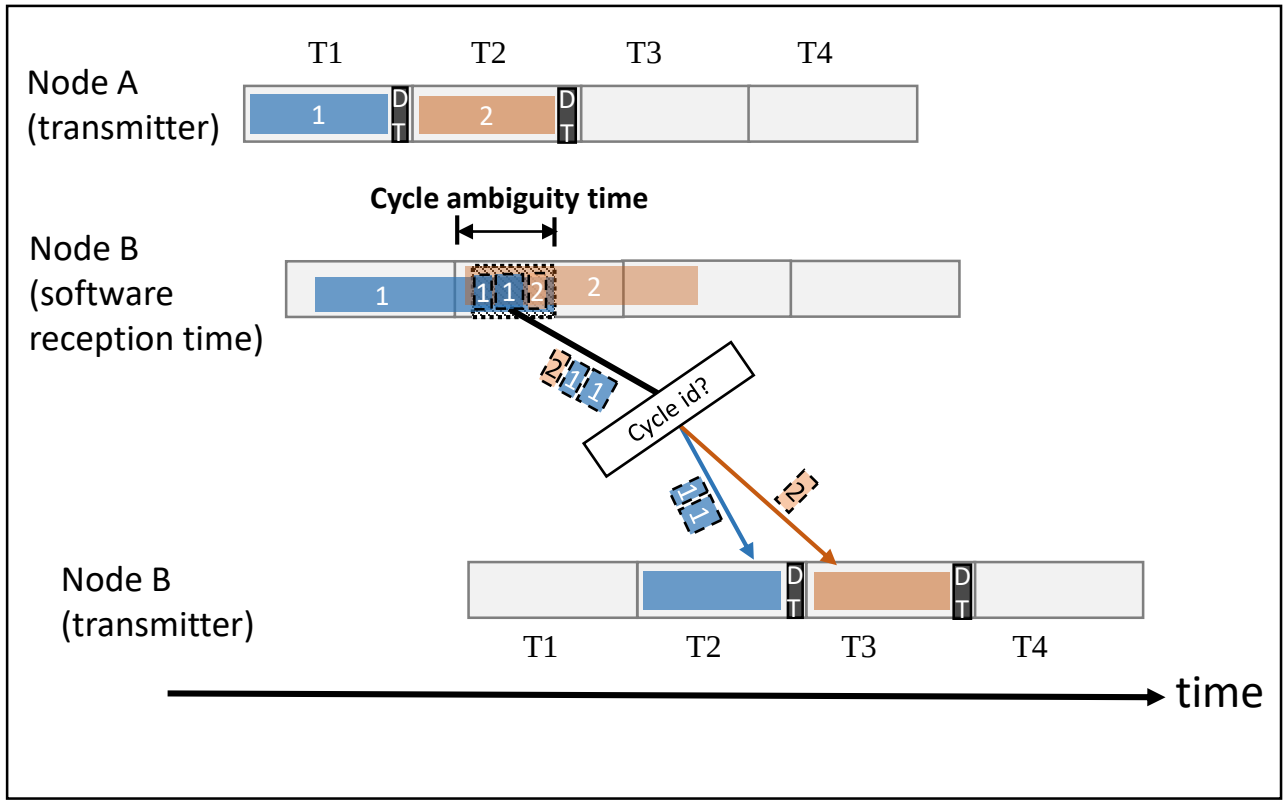
Goal - Improve the utilization in small cycle T





- **Why low utilization?** DT is 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)

Ethertype (F1-C1)	Reserved (16-bit)	Sequence number (16-bit)

- **Option A:** Use the last 3-bit in Reserved field for cycle id. 8 bins are sufficient in all cases?

Ethertype (F1-C1)	Reserved (16-bit)		Sequence number (16-bit)
	Reserved (13)	Cycle id (3)	

- **Option B:** Use the first bit in Reserved field for subtyping of cycle id. When set, field “sequence number” becomes 3-bit cycle id + 13-bit sequence.

Ethertype (F1-C1)	Reserved (16-bit)		Seq# (16-bit)	
	flag(1)	Reserved (15)	Cycle id(3)	Seq# (13)

How to carry a cycle id (cont'd)

2. Define a new cycle-tag

Ethertype (cycle-tag)	Reserved (8-bit)	Cycle ID (8-bit)

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

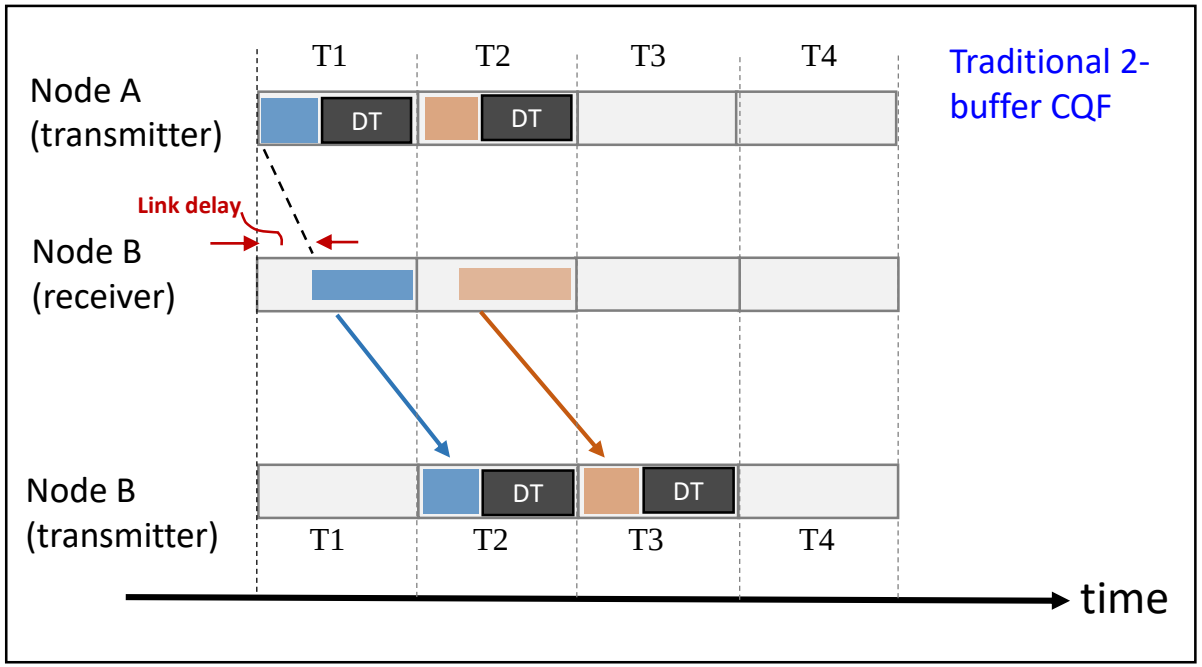
Ethertype (s-vlan)	vlan-tag (16-bit)	Ethertype (c-vlan)	Cycle id (16-bit)

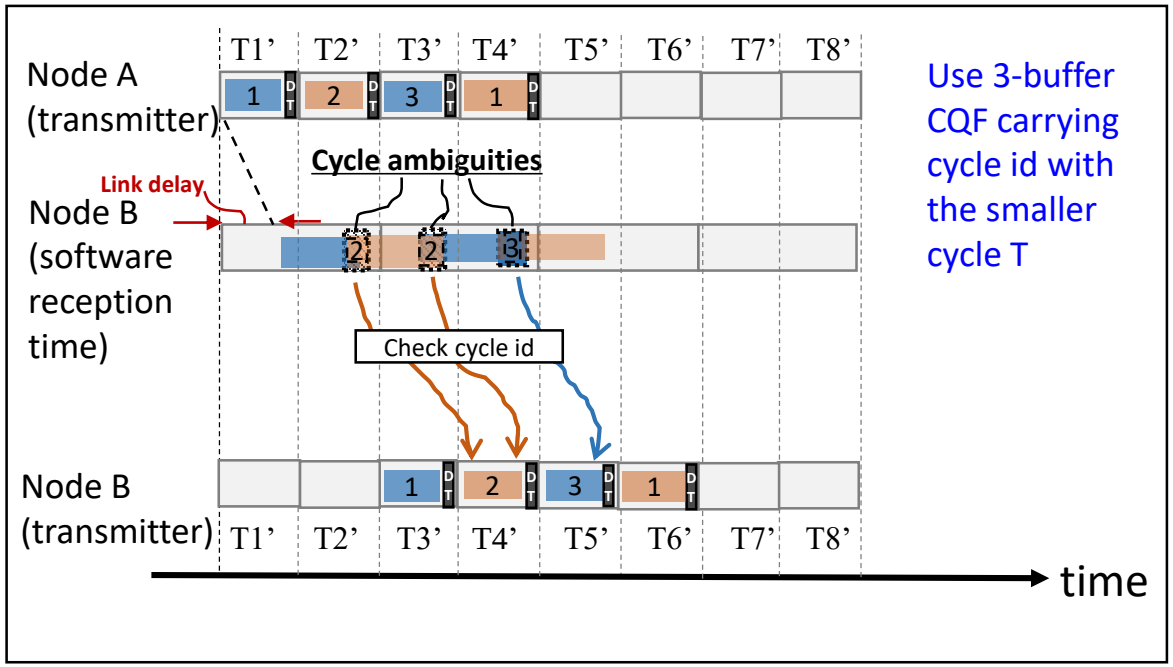
ACL example: if-match cvlan-id *cycle-id-in*

remark cvlan-id *cycle-id-out*

backup

Cycle id used in the extension to traditional 2-buffer CQF



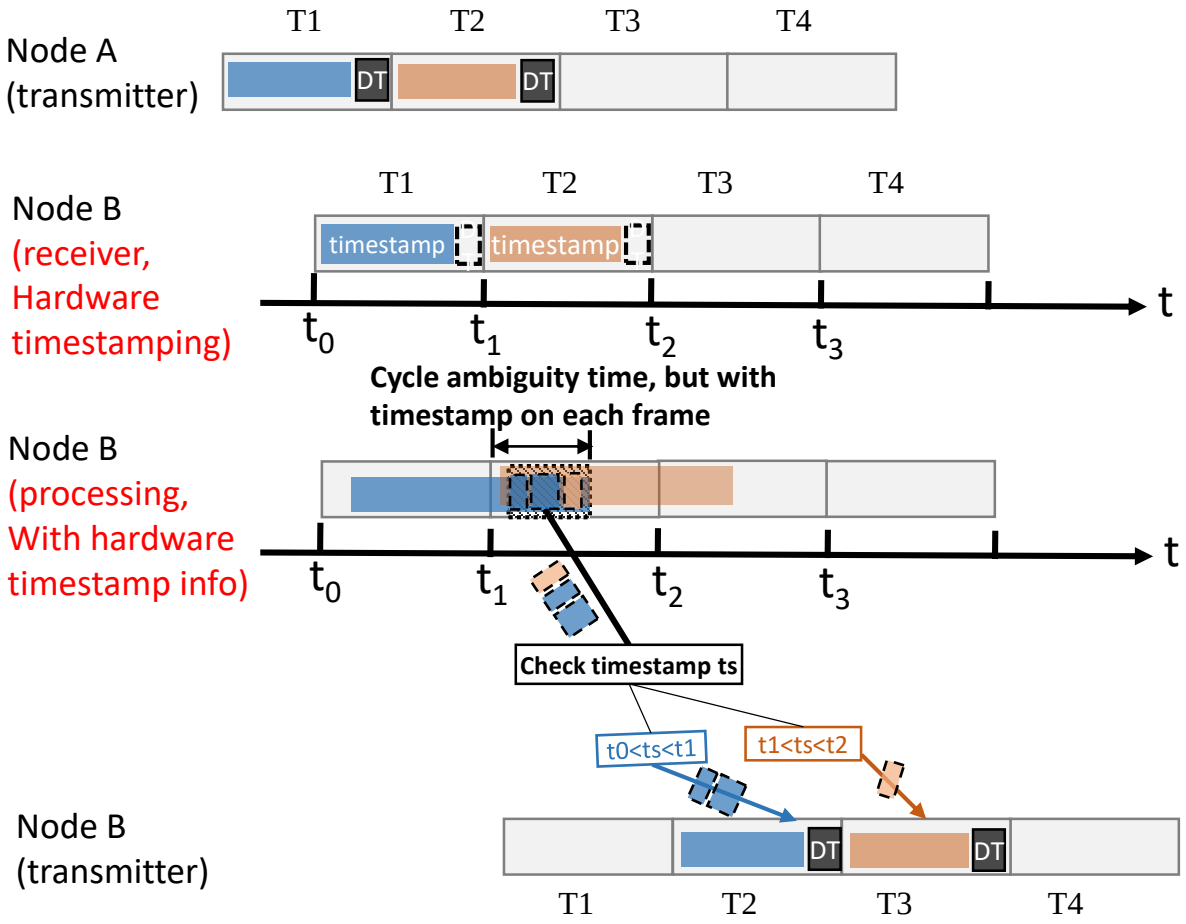


- Demand to have the restricted extension to traditional 2-buffer CQF usage
- Re-use traditional CQF as much as possible
- Nodes sync their cycles as usual, gate open/close controlled by sync'ed time
- Utilization gets worse as DT can not curve out the link delay

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

What if frame 1st bit hardware timestamping is available



- It can remove the cycle ambiguity
- However output delay at A can not be curved out from DT
- A feasible way when
  - Capability of hardware timestamping on the 1st bit of each data frame is available
  - Output delay is negligible