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)

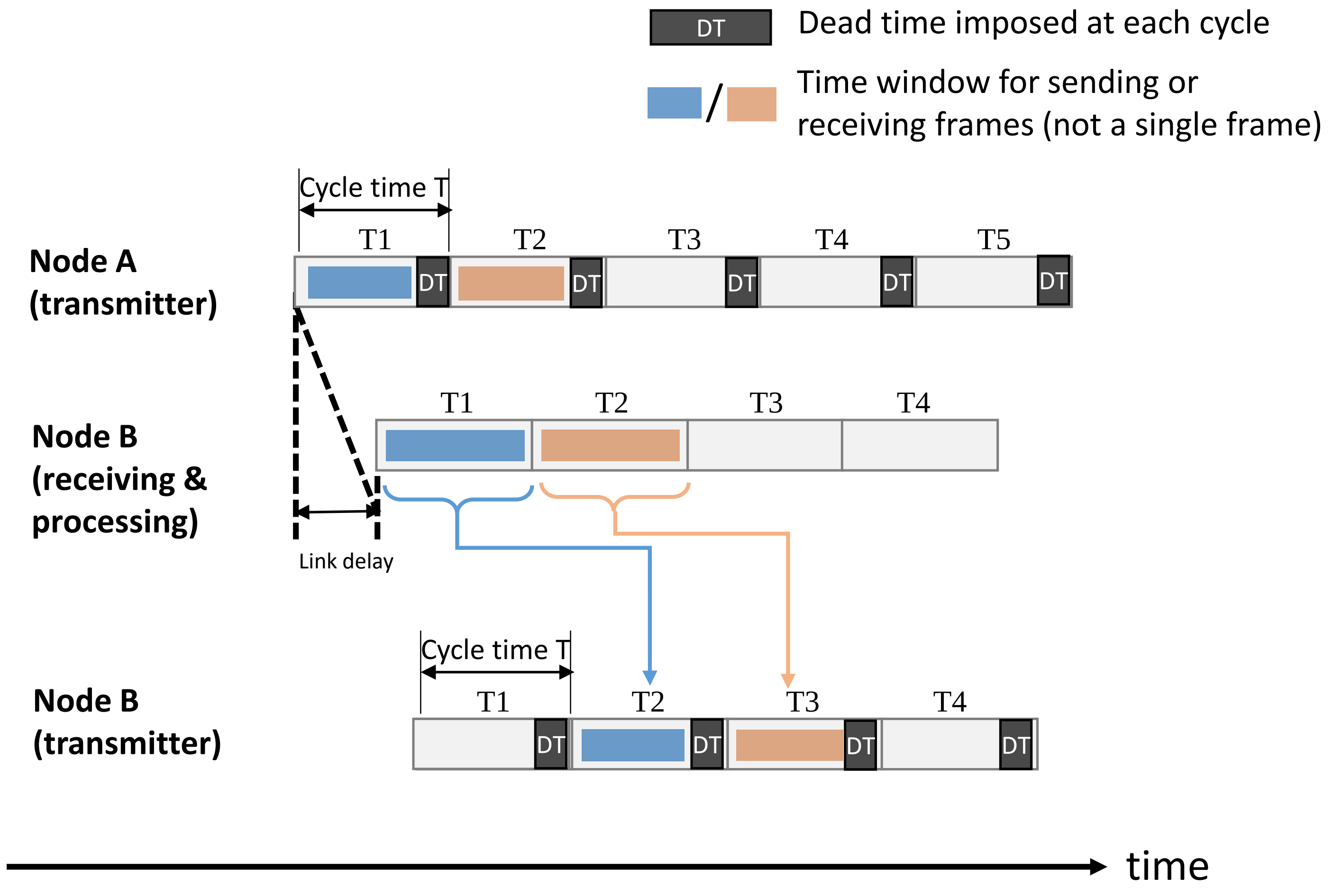
Sept, 2022

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 imposed at each cycle
/
Time window for sending or receiving frames (not a single frame)
Cycle time T
T1
T2
T3
T4
T5
Node A (transmitter)
Node B (receiving & processing)
Link delay
Node B (transmitter)
time

- 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

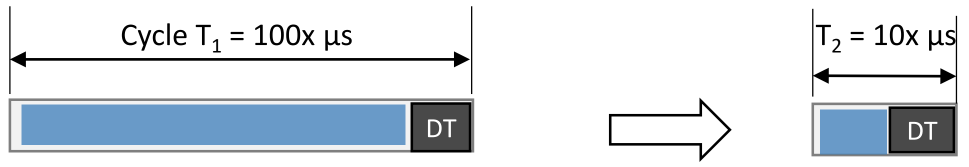
(*): draft-ietf-detnet-bounded-latency

Utilization unacceptably low in small cycle

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

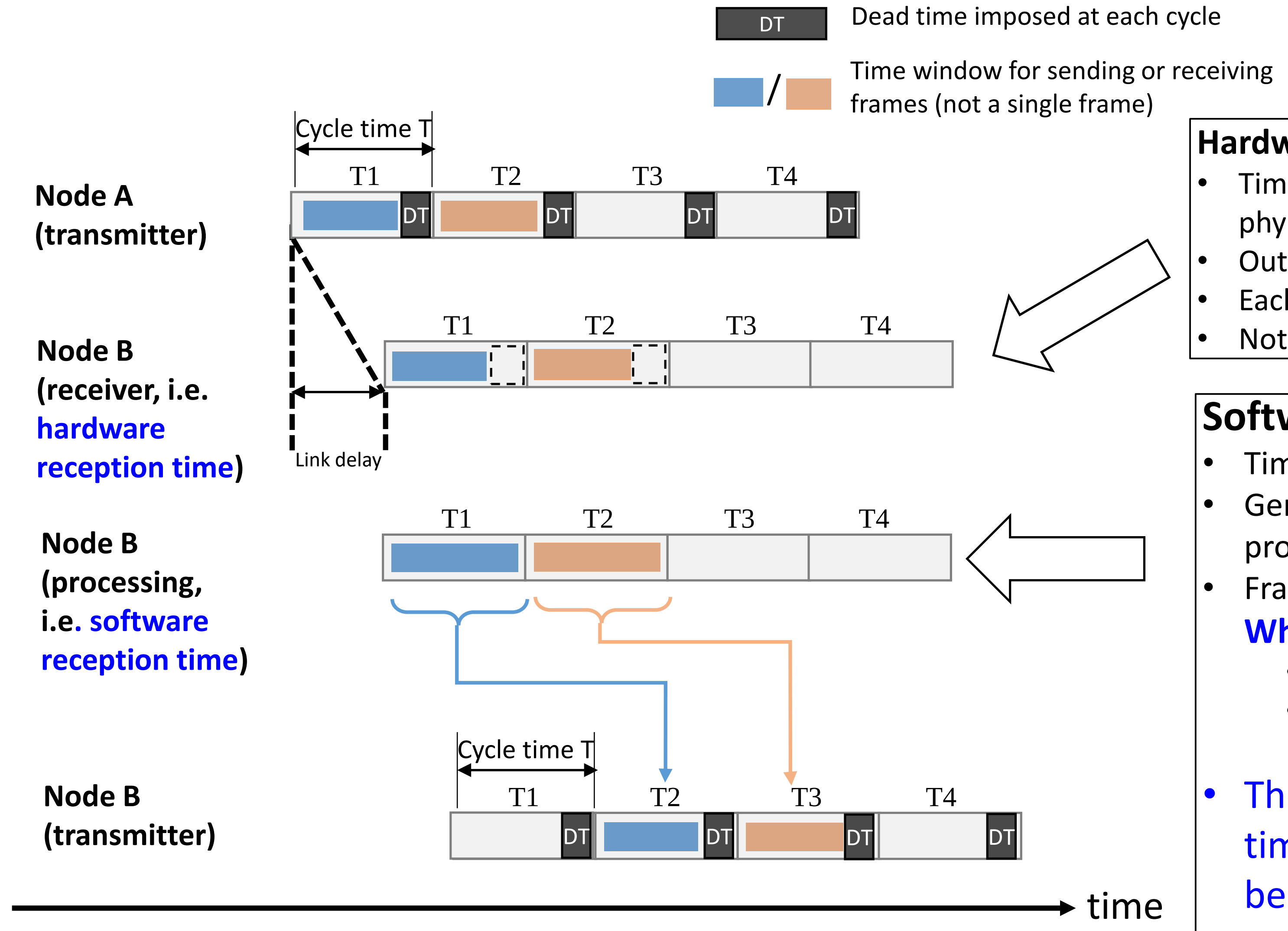
- Desire to use smaller T for better latency and jitter
  - To achieve e2e latency (e.g. < 1ms) or large # of hops (e.g. 10+)
  - A known app cycle 31.25 µs; cycle T < 31.25 would be desired
  - Should support T in the order of ~10x µs
  - DT is in the order of ~10x µs as well
  - DT and T are in the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

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



Unacceptably low utilization

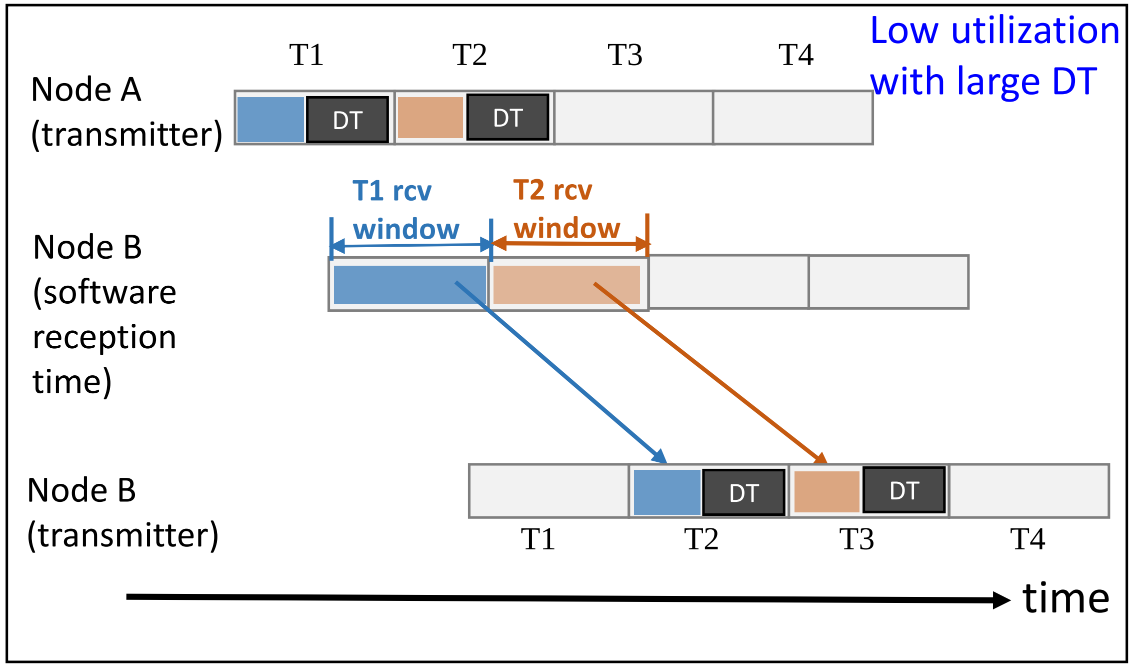
Hardware and Software frame reception time used for output bin determination

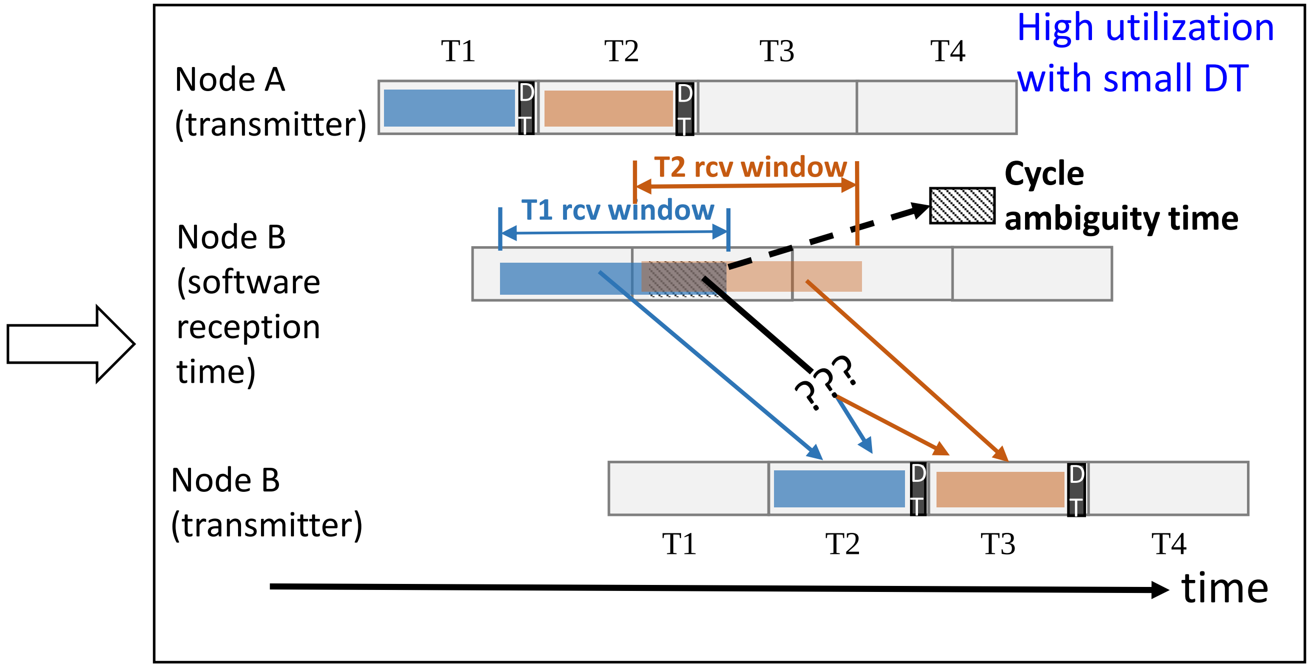

DT
Dead time imposed at each cycle
Time window for sending or receiving frames (not a single frame)
Cycle time T
T1
T2
T3
T4
Node A (transmitter)
DT
Node B (receiver, i.e. hardware reception time)
Link delay
Node B (processing, i.e. software reception time)
Cycle time T
Node B (transmitter)
time

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
- The following slides uses the software reception time to determine to which bin the frame should be put at node B

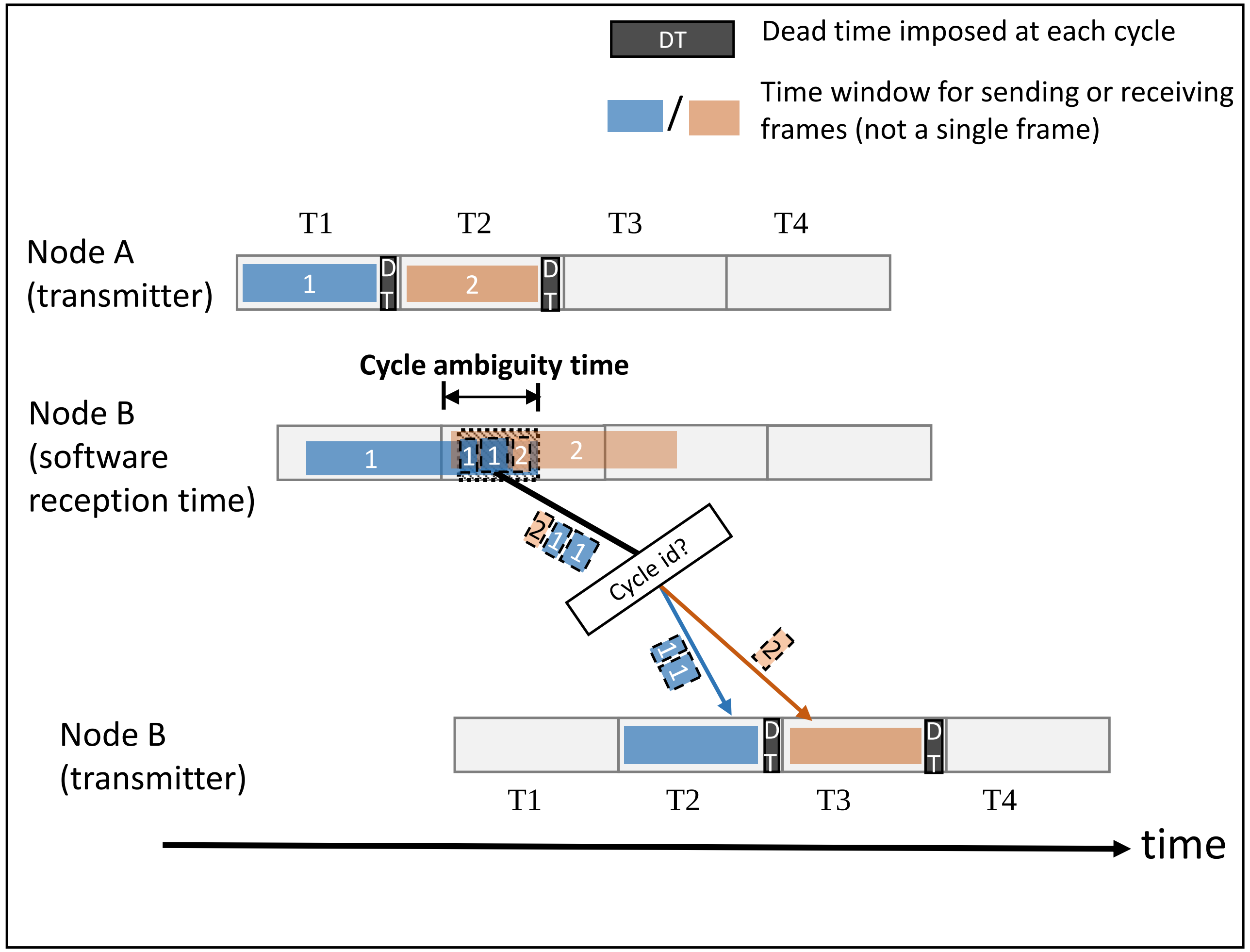
Goal - Improve the utilization in small cycle T





- **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


DT
Dead time imposed at each cycle
Time window for sending or receiving frames (not a single frame)
Node A (transmitter)
T1
T2
T3
T4
1
2
DT
Cycle ambiguity time
Node B (software reception time)
1
1
1
2
2
Cycle id?
Node B (transmitter)
T1
T2
T3
T4
time

- 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)

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

Ethertype (F1-C1)	Reserved (16-bit)			Sequence number (16-bit)
	flag(1)	Rsvd (11)	Cycle id (4)	

How to carry a cycle id (cont'd)

2. Define a new cycle-tag

Ethertype (cycle-tag)	Subtype (4-bit)	Reserved (4-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

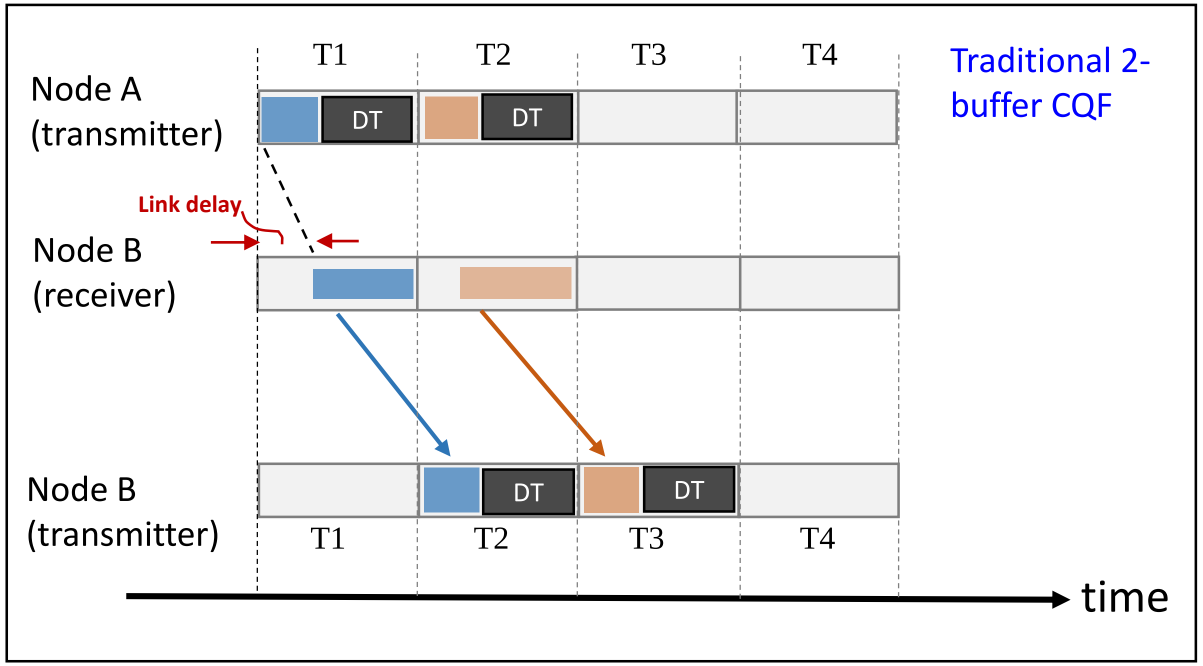
Ethertype (s-vlan)	vlan-tag (16-bit)	Ethertype (c-vlan)	Cycle id (16-bit)

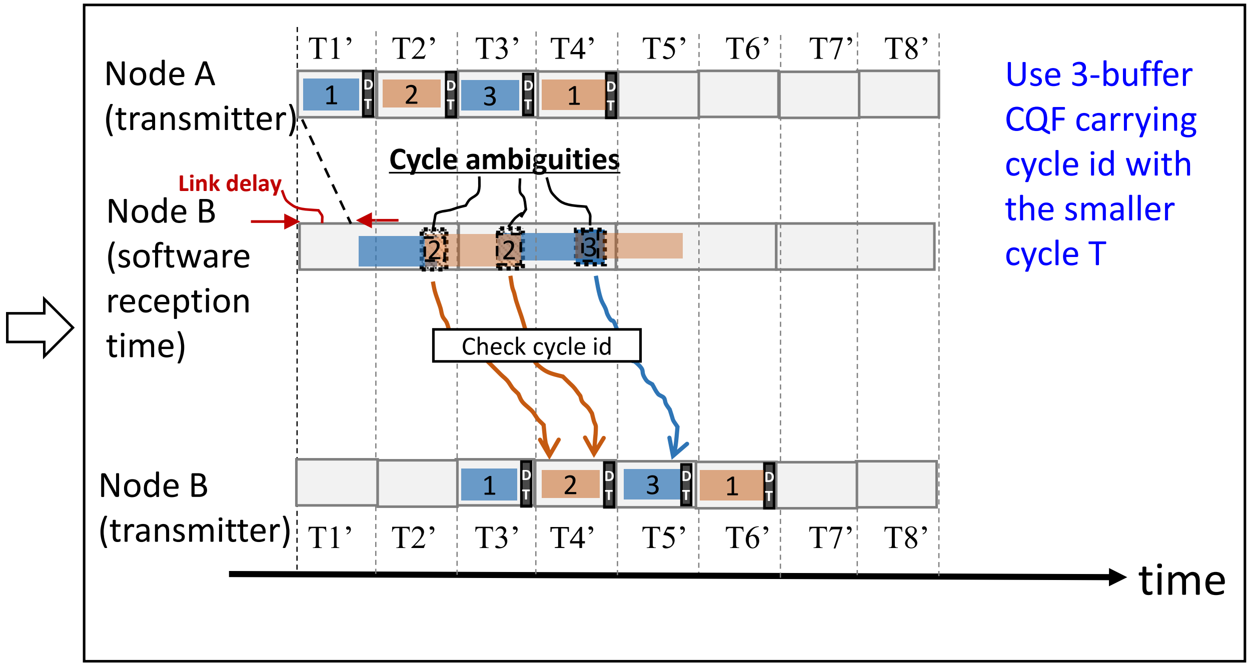
ACL example: if-match cvlan-id *cycle-id-in*

remark cvlan-id *cycle-id-out*

backup

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



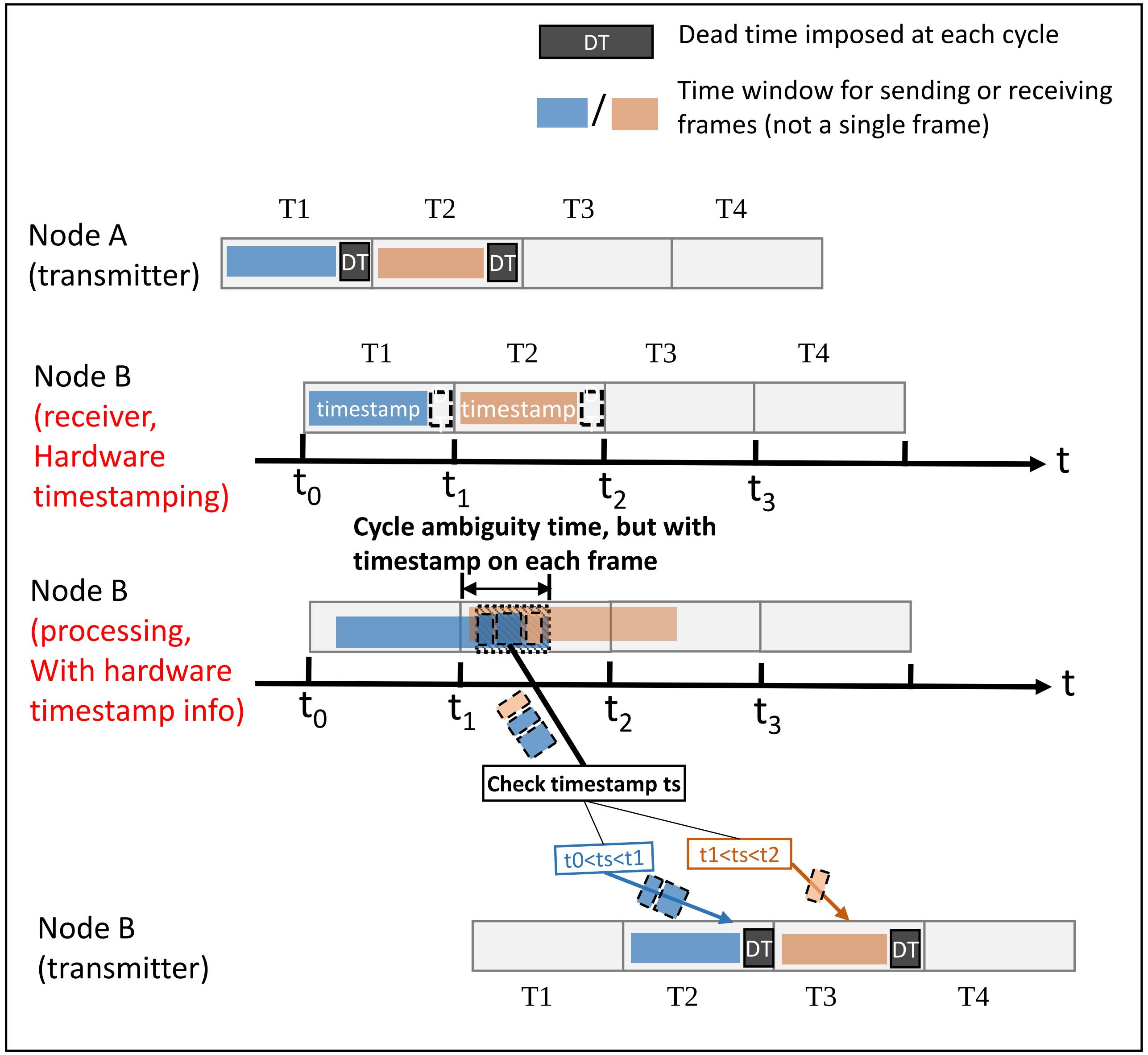


- Demand to have the restricted extension to traditional 2-buffer CQF usage
- Re-use traditional CQF as much as possible
- Nodes sync their cycles as usual, gate open/close controlled by sync'ed time
- Utilization gets worse as DT can not curve out the link delay

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

Procedures when using frame 1st bit hardware timestamping


DT
Dead time imposed at each cycle
Time window for sending or receiving frames (not a single frame)
Node A (transmitter)
T1
T2
T3
T4
DT
DT
Node B (receiver, Hardware timestamping)
timestamp
timestamp
t
t0
t1
t2
t3
Cycle ambiguity time, but with timestamp on each frame
Node B (processing, With hardware timestamp info)
Check timestamp ts
t0<ts<t1
t1<ts<t2
Node B (transmitter)

- 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.