Cyclic Queuing and Forwarding with tagging for Deterministic Forwarding

draft-eckert-detnet-tcqf-02
draft-yizhou-detnet-ipv6-options-for-cqf-variant-01

Toerless Eckert, Futurewei USA (tte@cs.fau.de), Yizhou Li <liyizhou@huawei.com>
Stewart Bryant, U. of Surrey ICS (s.bryant@surrey.ac.uk), Andy Malis (agmalis@gmail.com),
Guangpeng Li <liguangpeng@huawei.com>, Shoushou Ren <renshoushou@huawei.com>,
Guangpeng Li <liguangpeng@huawei.com>, Fan Yang <shirley.yangfan@huawei.com>,
Jeong-dong Ryoo <ryoo@etri.re.kr>, Peng Liu <liupengyjy@chinamobile.com>

Interim meeting 04/2023, rev 0.6

https://github.com/toerless/detnet/blob/main/tcqf-slides.pptx

Agenda

- Motivation (this is not vaporware): History / Context
- From CQF to TCQF
- Low Jitter
- Scale
- Packet encapsulations
- Specification Outline

History / Context

What did we do for our proposal.

IETF side: Tagged Cyclic Queuing drafts

- Historic drafts
  - 2018 - 2019: draft-qiang-detnet-large-scale-detnet (00-05)
    - (Christina) Li Qiang, Xuesong Geng, Bingyang Liu, Toerless Eckert, Liang Geng, Guangpeng Li
  - 2019: draft-chen-mpls-cqf-lsp-dp-00
    - Zhe Chen, (Christina) Li Qiang
  - 2021: draft-dang-queuing-with-multiple-cyclic-buffers-00
    - Bingyang Liu, Joanna Dang
  - 2021: draft-eckert-detnet-mpls-tc-tcqf
    - Toerless Eckert , Stewart Bryant , Andrew G. Malis , Guangpeng Li
    - Prior drafts all focussed on mechanism (only)
    - This attempt to write up all aspects considered to be required for standardization – and possible fastest adoption:
      Describe on-wire-behavior for MPLS (only) – matches best developed DetNet data plane.
      Then David Black cleared up confusion about DSCP...
- Current drafts – intent to merge:
  - 2022: draft-eckert-detnet-tcqf-02
    - From –mpls-tc-tcqf, added IP/DSCP encap (intradomain!).
  - 2022: draft-yizhou-detnet-ipv6-options-for-cqf-variant
    - Yizhou Li, Shoushou Ren, Guangpeng Li, Fan Yang, Jeong-dong Ryoo, Peng Liu
    - Proposed / discusses IPv6 new header options – and more deteails of CQF->TCQF evolution to meet the scaling requirement

Development / Validation

Deterministic IP (DIP) “research” project

research ~= advanced PoC engineering

- IFIP 2021 Networking Conference:

  “Towards Large-Scale Deterministic IP Networks”

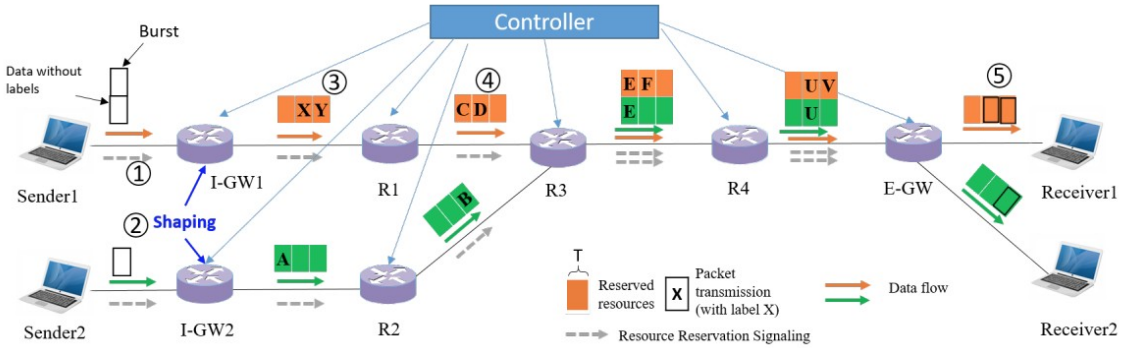
  †Bingyang Liu, †Shoushou Ren, †Chuang Wang, ∗Vincent Angilella,, Paolo Medagliani, ∗Sebastien Martin, ∗, Jeremie Leguay

  https://dl.ifip.org/db/conf/networking/networking2021/1570696888.pdf

Two validation vehicles

- High-speed PoC router implementation
  - Modified off-the-shelf metro-router 100Gbps Ethernets
    - Added FPGA for programmable queuing
    - Packet header processing (cycle-ID) by standard NPU

- Omnet++ Simulation
  - Using published AS 1239 (Sprint) topology of 315 routers and 1944 links



Architecture (from paper)

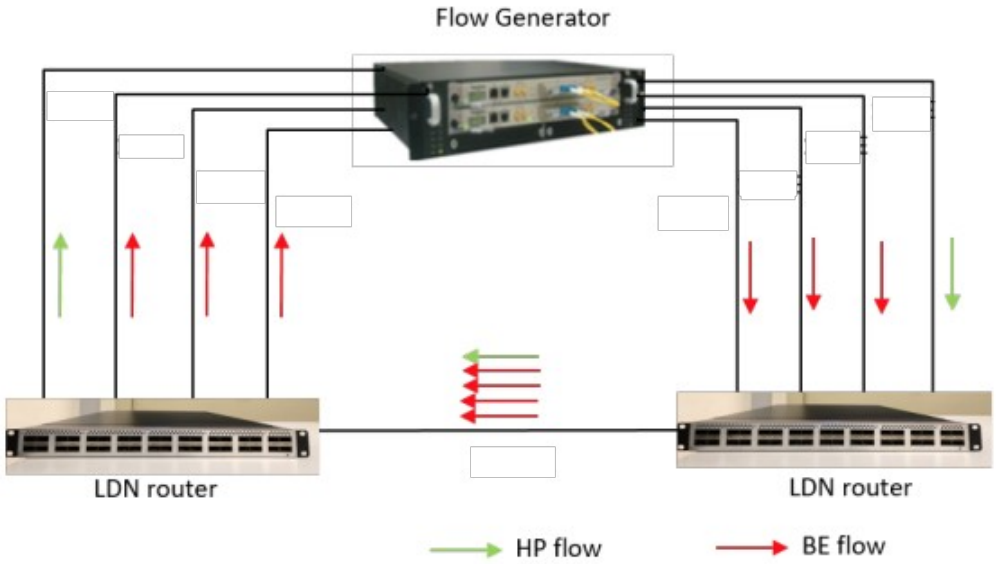


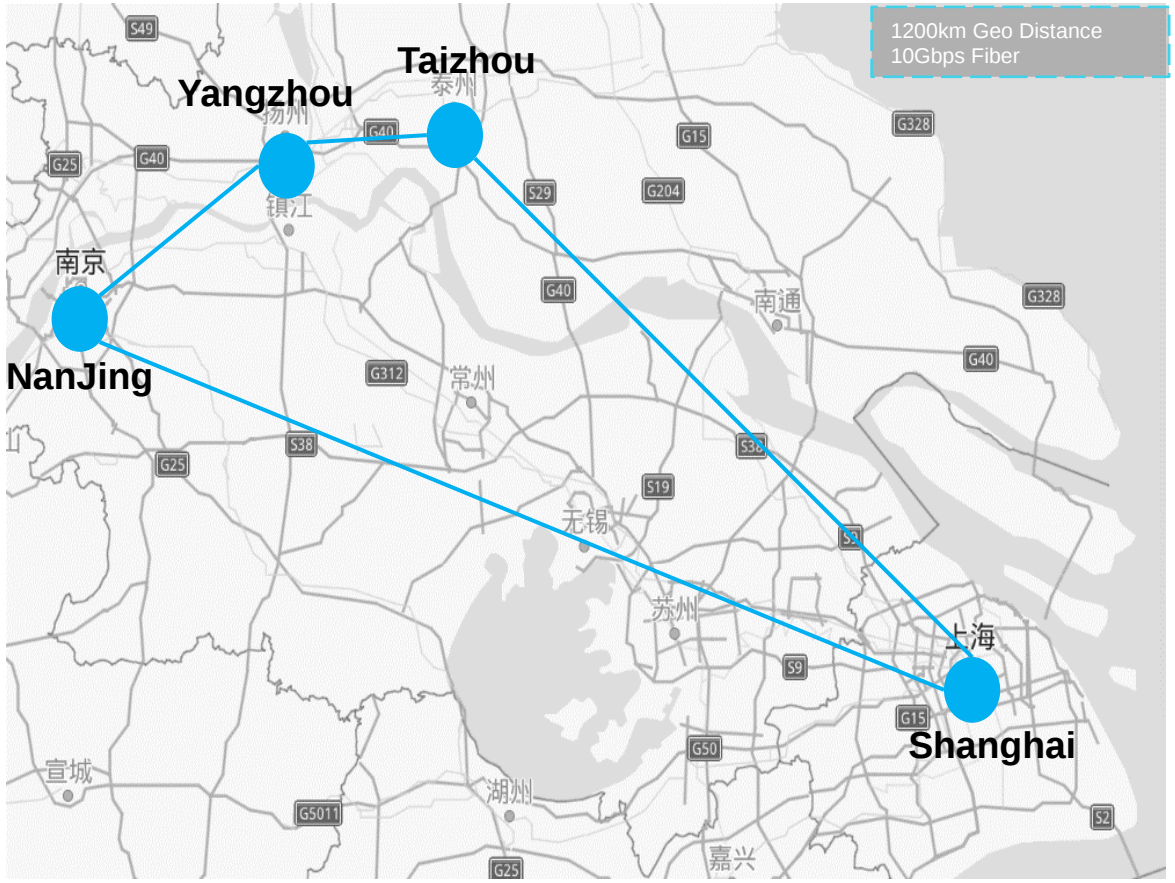
Fig. 5. Test topology for the PoC implementation.

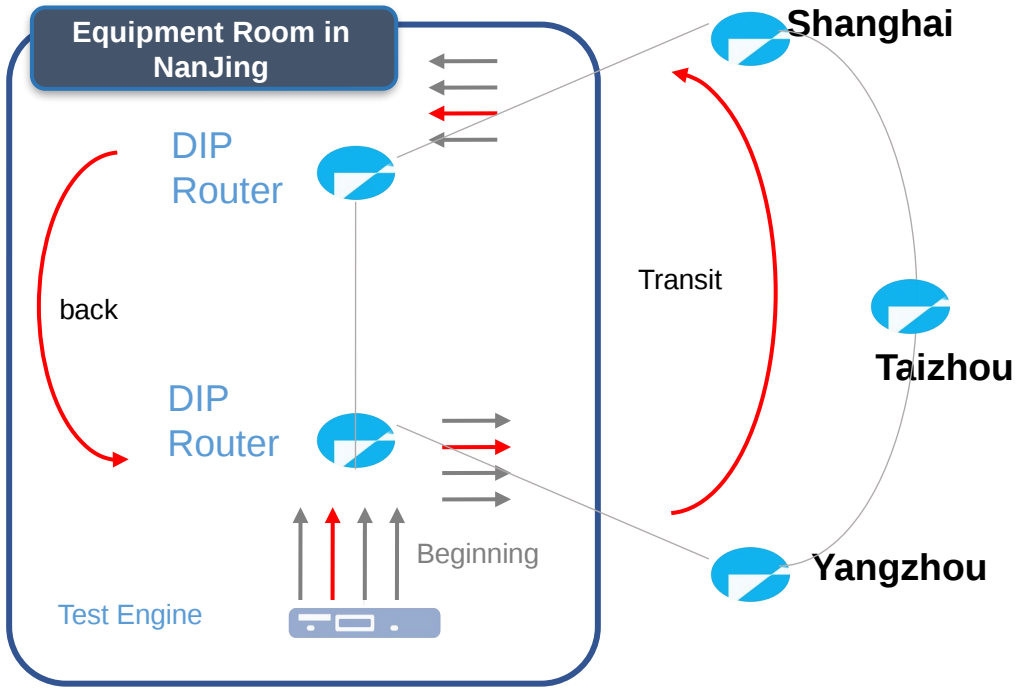
Large scale validation

- 2020:
  CENI: Research network across china
  - Connecting all mayor cities
- Connected PoC routers to 100 Gbps fibers of network
  - And performed validation testing
- Report:
  - https://ceni.org.cn/406.html

CENI DIP (tcqf) Testbed 2020

Example test case (NanJing region)





Target Flow：1Packet/10μs & pktsize is 1500B

Interference flow：128Packets/1ms & pktsize is 1500B

CENI TCQF (“DIP”) Testbed 2020

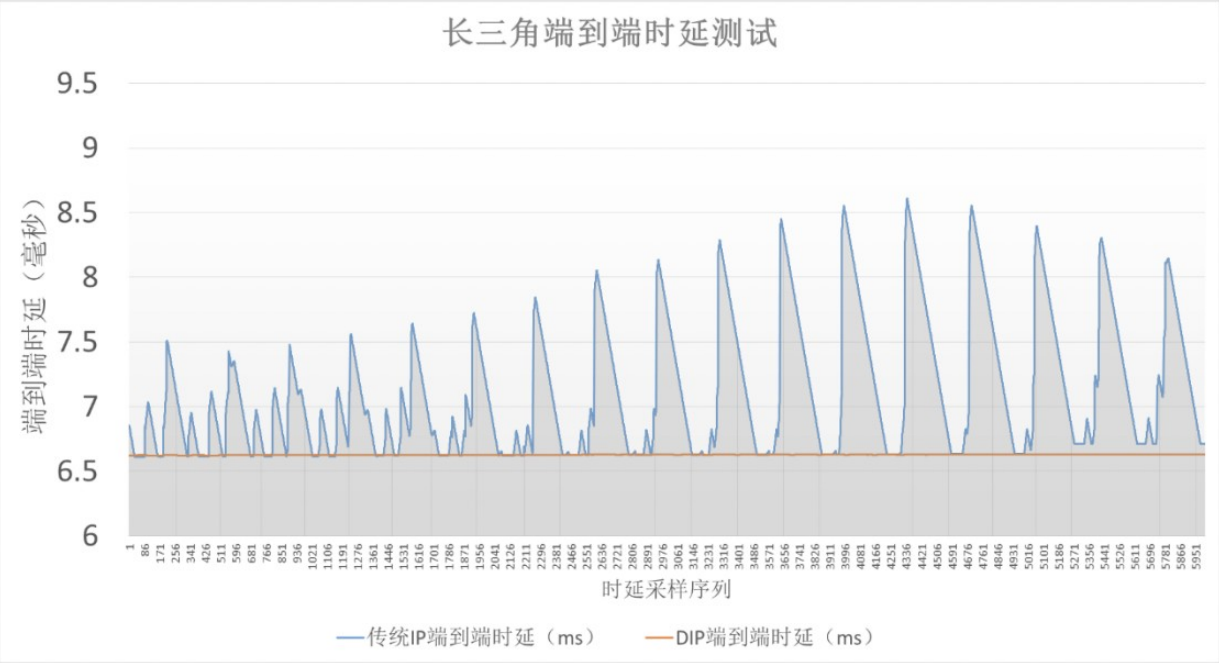


表 4 长三角综合试验网 DIP 测试统计结果

干扰流数量	传统 IP 转发端到端时延（微秒）				DIP 转发端到端时延（微秒）			
	最小	平均	最大	**最大抖动**	最小	平均	最大	**最大抖动**
1	6304	6306	6386	**83**	6360	6374	6389	**29**
2	6304	6311	6564	**261**	6360	6374	6389	**29**
3	6304	6321	6751	**447**	6360	6374	6389	**29**
4	6304	6370	7612	**1308**	6360	6374	6389	**29**
5	6304	6463	7977	**1673**	6360	6374	6389	**29**
6	6304	6577	8343	**2039**	6360	6374	6389	**29**
7	6304	6695	8608	**2304**	6360	6374	6389	**29**

Summary

- TCQF can be (easily) implemented at scale in existing type of wide-area network routers
- TCQF has been validated through simulations and PoC implementations in wide-area (2000 Km paths) network deployment.
- TCQF provides very low jitter (“on-time” forwarding)
  - We think this is core requirement for many core use-cases

- TCQF can support any IETF network layer data-plane
  - Validation with IPv6
  - Drafts currently IP/IPv6/MPLS
  - SR – SRv6, SR-MPLS equally supportable
  - TBD: BIER (would require RFC8296 header enhancements)

- IP/IPv6/MPLS could be supported short term without new packet headers
  - More on pro/cons of new headers later in slide deck

From CQF to TCQF

From draft-yizhou-detnet-ipv6-options-for-cqf-variant

CQF

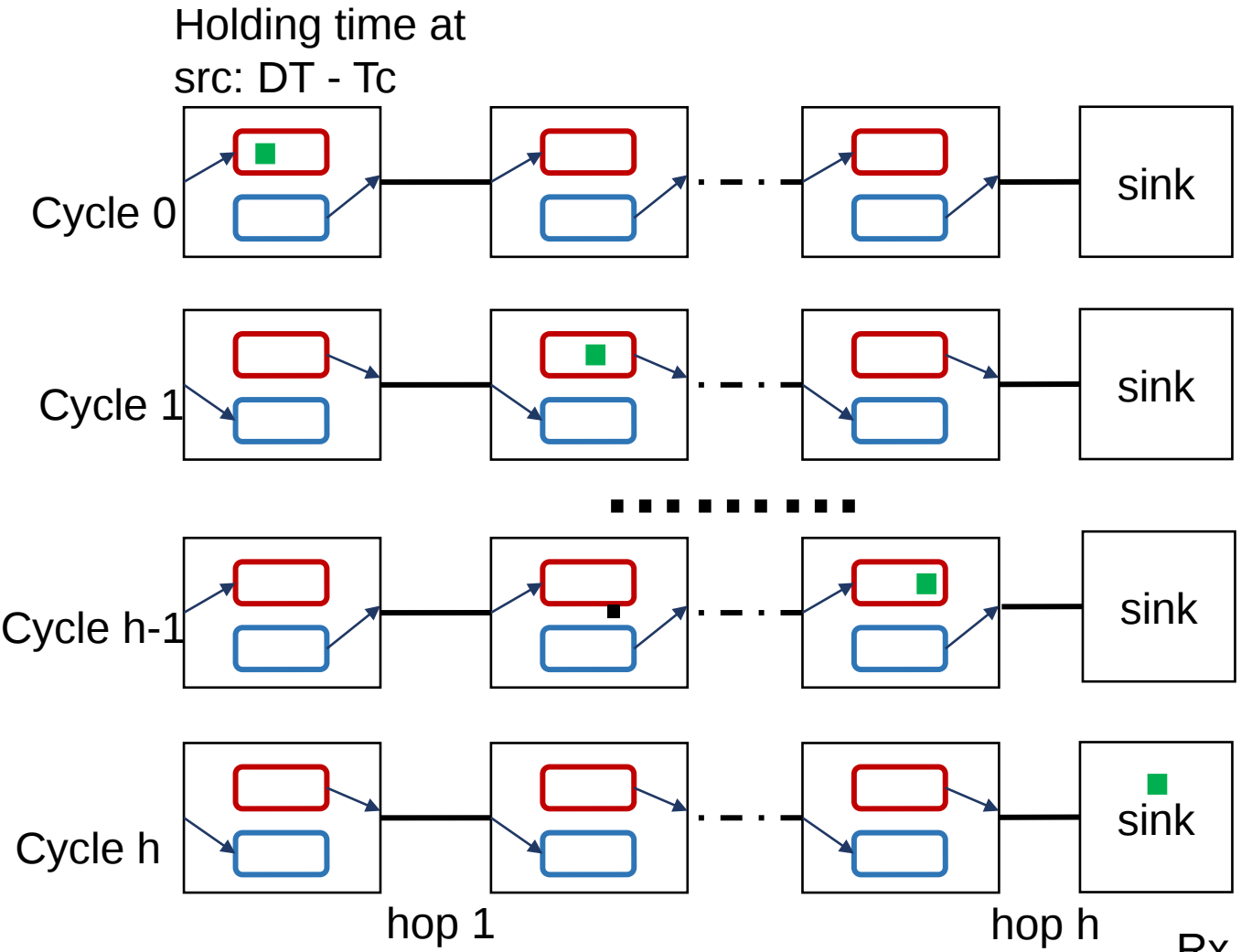
- Cyclic Queuing and Forwarding (CQF), IEEE802.1Qch
  - Part of IEEE Time Sensitive Networking (TSN) standards
  - Evolved from TSN Time Aware Shaper (TAS), IEEE802.1Qbv
    could be called a simple profile of TAS (which itself is quite complex)

- At core of TAS/CQF are per-hop forwarding of packets based on their arrival time on each switch
  - This allows to operate TAS/CQF without TAS/CQF specific headers

- TAS/CQF designed for <= LAN deployments
  - Intra-Car ... Manufacturing floor
  - Speed 100 Mbps, 1 Gbps (max 10 Gbps, but rarely used)

Fundamental CQF operations (2 buffer)

Most simple bounded latency calculus, Low jitter



- 2-buffer per port. Input and output swap once every **cycle interval Tc.**
- E2e latency across h hops:
  - Max: (h+1) Tc
  - Min: (h-1) Tc +DT*
  - (*): DT = dead time (revisit later). very small in fundamental CQF
- Attractive “simplicity” features:
  - Simple calculable latency bound: only relevant to Tc and h, ≈ h*Tc
  - Simple maintenance: no per-stream per-hop state maintenance

Rx time after cycle h-1:
0 - Tc

Higher speed networks make basic CQF mechanism more attractive!

- Wide-area network deployment requires supporting one or combination of the followings:
  1. Smaller e2e queuing introduced latency bound
  2. Potentially larger number of hops (seen e.g.: L3 mobile access rings - 15++ hops!)
     - Even manufacturing floor networks may have multiple hops for policy reasons...
  3. Links with longer propagation latency (distance -> speed of light)
  4. Larger in-node processing time variance (more complex/larger nodes)

1. CQF latency bound ≈ h * Tc

   Higher speed link provides the potential to reduce Tc, even with larger h

   - allow at least one 1500B/max size packet to be sent within Tc
   - With inc… mount of data can be transmitted within a smaller cycle time
   - Counteract larger h

Cycle Time (µs)	Buffer size per Cycle (Byte) Link bandwidth			
	100Mbps	1Gbps	10Gbps	100Gbps
1	12.5	125	1250	12500
1.2	15	150	1500	15000
2	25	250	2500	25000
4	50	500	5000	50000
10	125	1250	12500	125000
12	150	1500	15000	150000
120	1500	15000	150000	1500000

Cycle time decreasing:
100x µs -> 10x µs -> few µs

CQF Dead Time issue

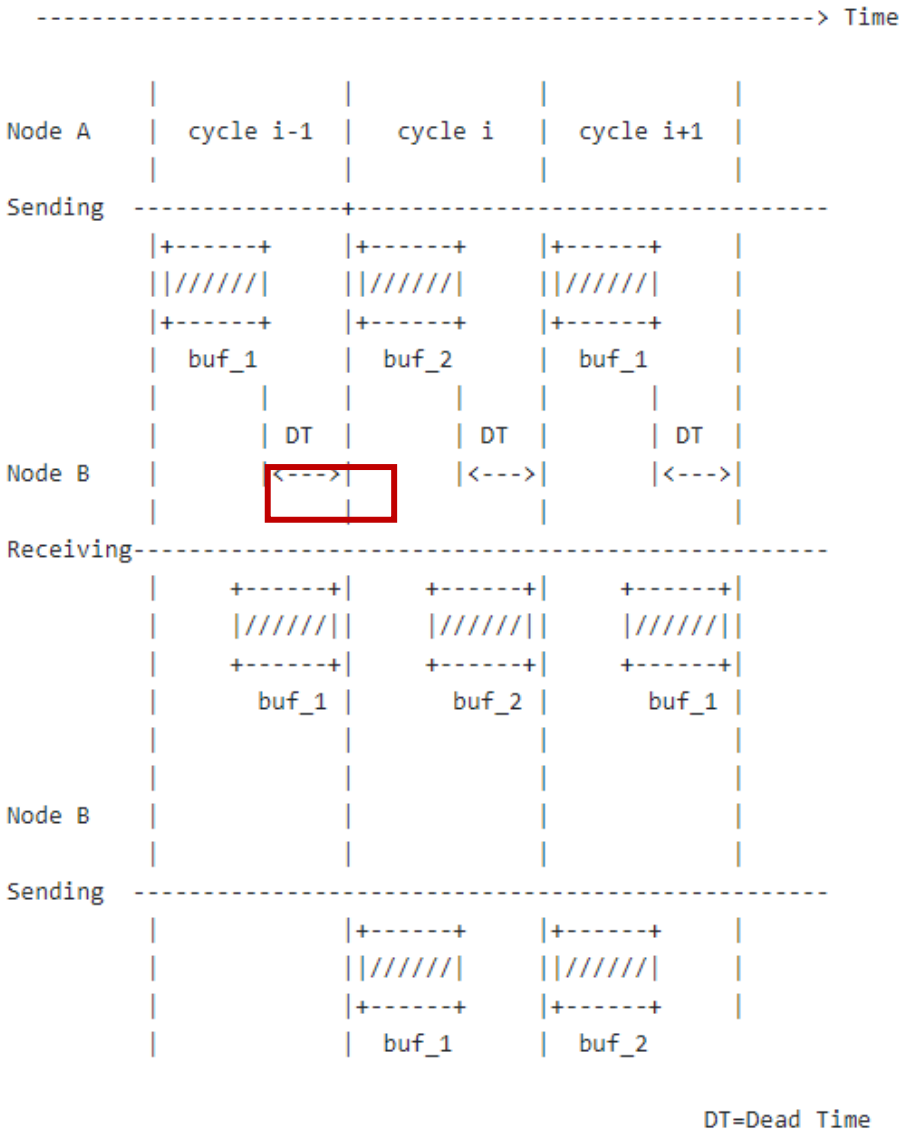


Figure 2: Fundamental Two Buffer CQF

Revisit DT (dead time): the last byte sent by node A in cycle (i-1) has to be ready for sending at node B before the start of cycle i.

DT is at least: max link propagation delay + max processing delay at the next node + max other time variations (e.g.: clock MTIE).

The longer the propagation or processing delay, the larger the DT.

The larger DT is in proportion to Tc, the lower the possible utilization

At DT >= Tc, throughput goes to zero!

Example for common 1Gbps CQF: LAN > 2 Km length => DT > Tc.

Dead Time issue makes CQF insufficient for large-scale:

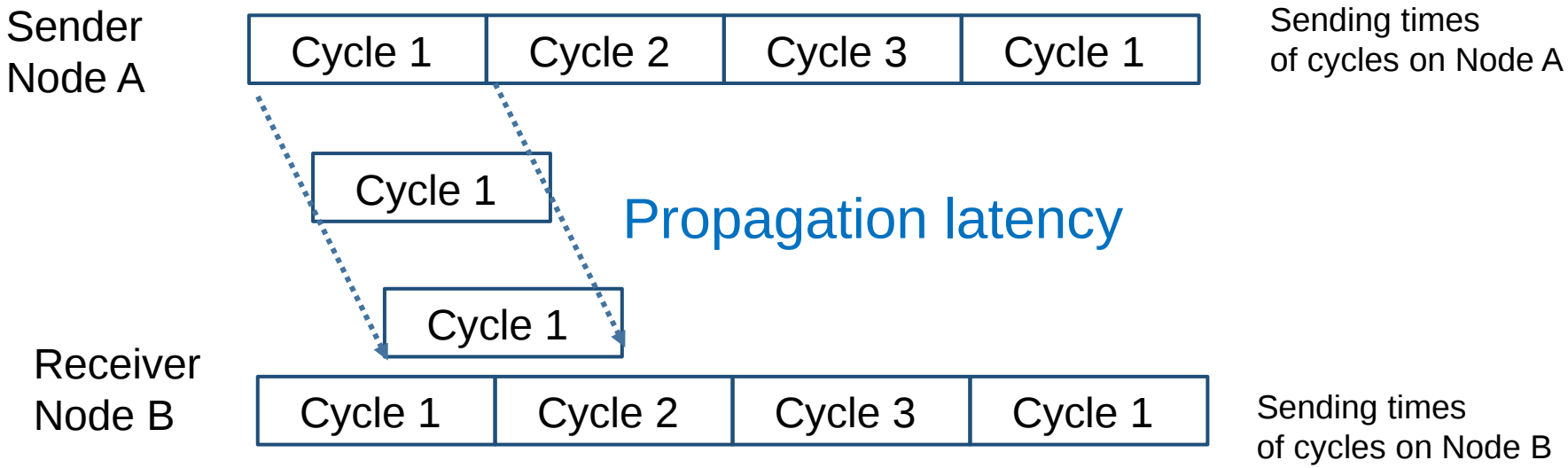
(1), (2) Shorter Tc for lower e2e latency bound

(3), (4) Larger DT because of longer link and/or processing time

Need smaller ratio of DT/Tc for better utilization

- Solution without Dead Time would be ideal! (100% utilization)

3 cycle operation

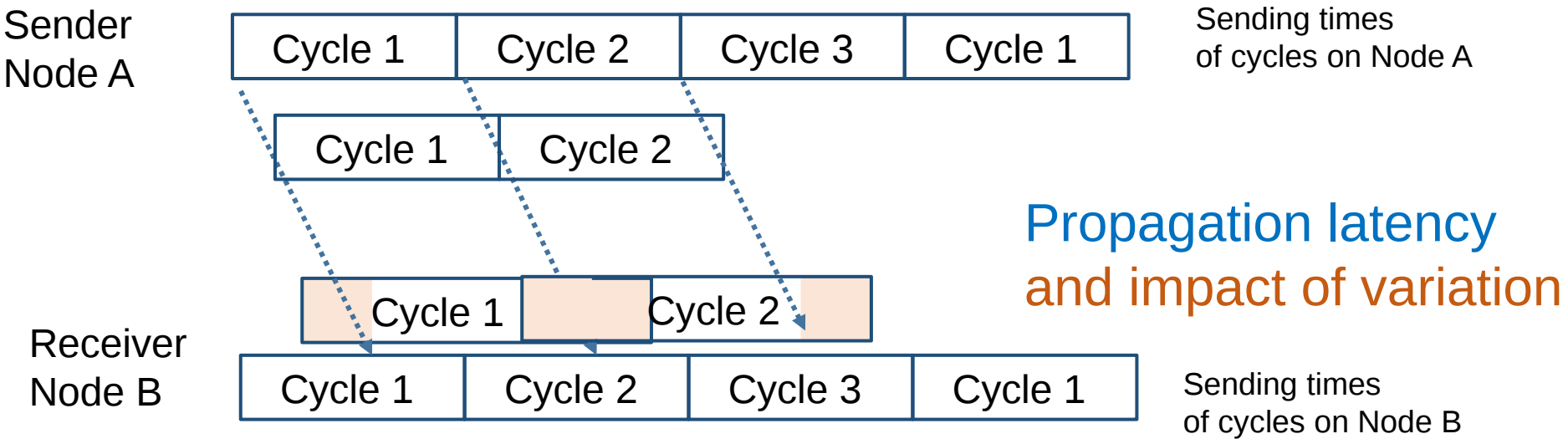


- In example, A/Cycle-1 needs to get mapped into B/Cycle-3
  - Because while packets from A/Cycle-1 arrive, B sends from Cycle-1 and Cycle-2
- This would work without packet header tag – as long as there is no jitter/variation whatsoever
  - Because we could exactly deduce from arrival time of packet when it was sent – and hence its cycle
  - This is one (non-adopted)proposal in IEEE-TSN (“multiple buffer CQF”)

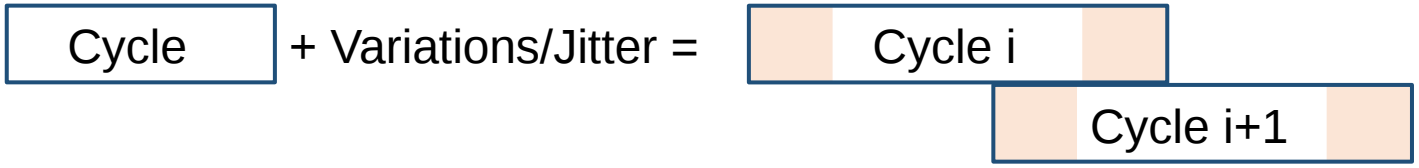
Variations

- Most important (our core reason!) ? Differences in synchronized clocks
  - Maximum Time Interval Error (MTIE)
  - If B thinks it is time T, then it is [T-MTIE...T+MTIE] at A
  - CQF at 1Gbps already requires sub usec MTIE so that variation due to MTIE does not impact performance
    - At 100Gbps, it would need to be 100 times better (impossible ?)
- Differences in link/media propagation latency
  - ~~Speed of light changes (great Sci-Fi movie!)~~
  - Length of link changes, e.g.: pole hanging copper (powerlines, cable) – up to 30..40% length between hottest/coldest times
- Differences in processing latencies
  - Irrelevant ?
  - FEC / ARQ introduced variation (8 msec in DSL FEC – bad example, low-speed)
  - WiFi / Radio links – FEC / ARQ / interference / reflections / ... ? (TBD)
  - High-speed , complex forwarder , software forwarders
  - Buffering out variations in all these cases at lower processing layer could be more complex than solving variations through TCQF

Operation with Variations / Jitter



- When packets arrive at B, B can not determine from arrival time which Cycle the packet was sent from.
  - Jitter adds to the possible “length” (in time) of the received cycle data



Impact of Variations for CQF (no tagging)

- We can not simply “guess” the sending cycle / put packet into one of the cycle-buffers
  - Messes up admission-control / latency calculation / creates possible congestion.
- We could use Dead Time concept
  - This quickly reduces possible throughput:
    With variations at 50% cycle-time, throughput is 0.
- With tagging, we never need to reduce throughput
- But: The higher the variations, the more cycles we need
  - Exact calculations currently not written into draft, but easy to add.
  - E.g.: variations < 50% cycle time -> 4 cycles suffice.

Summary

- CQF has attractive features and potentials for

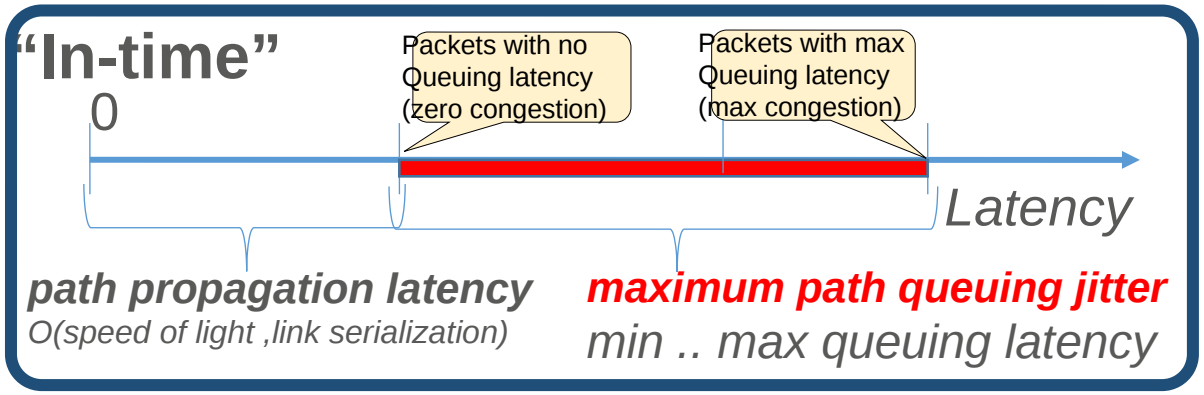
  Large-scale Deterministic Network deployments

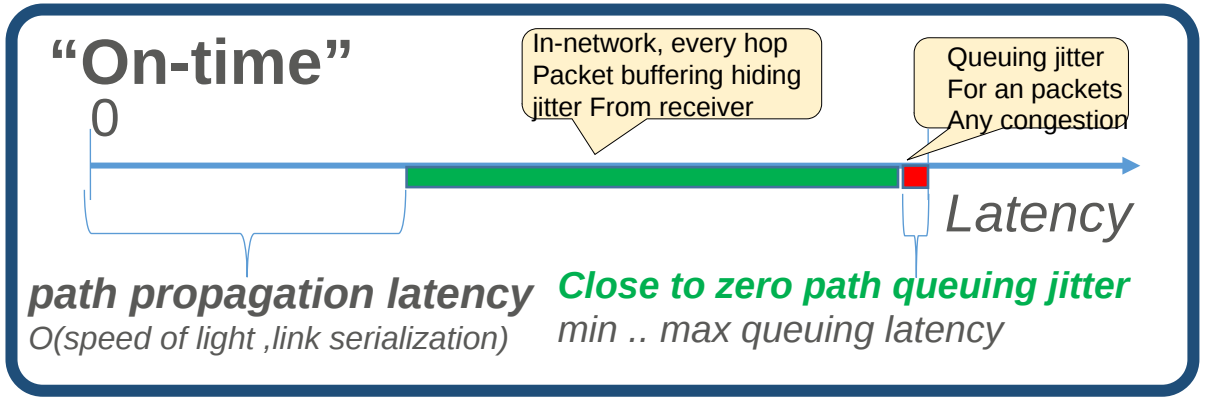
- Multi-buffer CQF is a straightforward extension from TSN CQF:
  - Easy to build (see PoC) with hardware that would also be required for CQF – but at higher speeds!
  - Cycle to Cycle mapping configuration based on link propagation latency
  - Number of cycles based on variations
- Eliminate “reception time based” assignment of packets to cycle buffer!
  - Rely on metadata “cycle-id” in packet instead (“tag”)
    - Tag-switching: predecessor of MPLS (hop-by-hop rewritten packet header field).
  - Many different options how to do this across various forwarding planes.

- Result:
  - Can support arbitrary links (propagation latency, latency-variation)
  - Can support lower clock-sync accuracy =~ higher-MTIE
  - Can support higher node propagation latency variations

Low Jitter

Jitter bounds – “In-time” vs. “On-time”





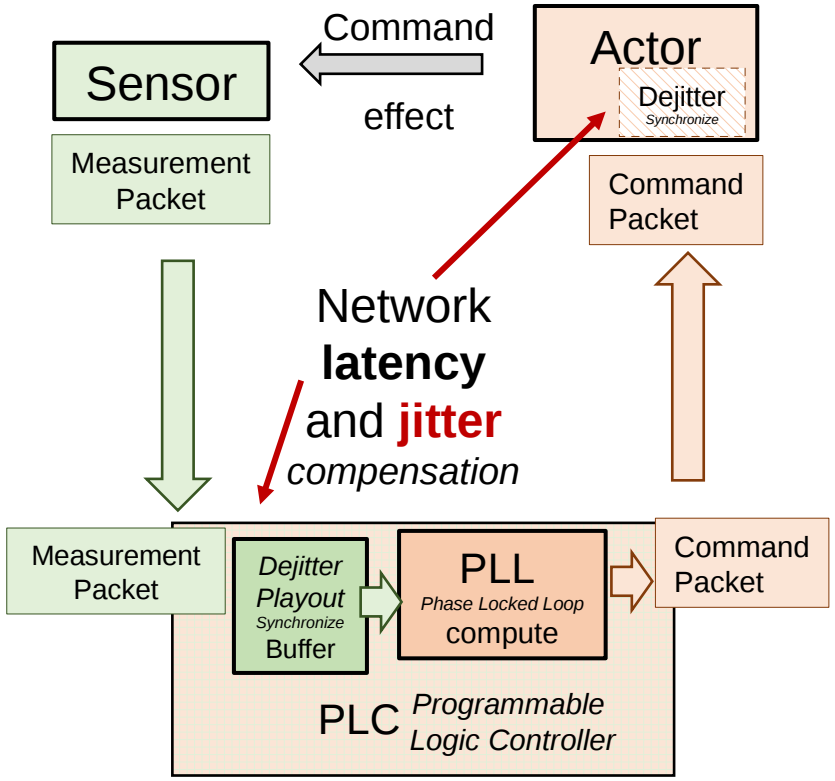
“In-time” delivers packets as soon as possible.

- Without traffic contention, no in-network per-hop queuing latency
- IETF IntServ/Guaranteed Service, TSN-ATS, C-SCORE use this model
- With maximum contending traffic, per-hop latency is maximum

- Result: “In-time” creates worst-case jitter between no..max traffic load
- On-time eliminates this jitter
  - Packet latency end-to-end always close to bounded latency
  - CQF, TCQF, “Damper” mechanisms provide this benefit
  - Othre mechanisms would require “playout buffer” on receiver / receiver edge.

Why we need low jitter

- Application traffic profiles, e.g.: Industrial Internet Consortium (IIC) _Time Sensitive Networks for Flexible Manufacturing Testbed https://www.iiconsortium.org/white-papers.htm
  - Isochronous, Cyclic, Audio-Video, (on-time), Alarm and Events (in-time), …
- Media playout and most control loops want on-time
  - Media: Synchronous playout
  - If network is in-time, packets must be buffered in application and consumed at maximum guaranteed latency time
- Playout buffer size requirement depend on network network size
  - Edge-router/receiver MUST be built against an assumed largest possible network ! Expensive war stories in industry how this can fail miserably
- In-time also raises clock synchronization needs on devices
  - On-time delivered packets carry implicit timing information !
  - Dumb devices (actors) may not be able to support dejittering and/or accurate clock
  - Classical example: accurate PLC with periodic “polling” of dumb actors/sensor without accurate clock: on-time allows for isochronous operation.
    - In-time would require much more complex sensor/actor/control-loops



Summary

- We need low-jitter “on-time” network services to support possible large number of “no-clock-sync” devices on network (sensors, small actors) for “Distributed IoT” use-cases.
- Per-hop “on-time” mechanism (TCQF, Dampers) are solution to avoid problem of playout buffer scaling on edge-router hardware.

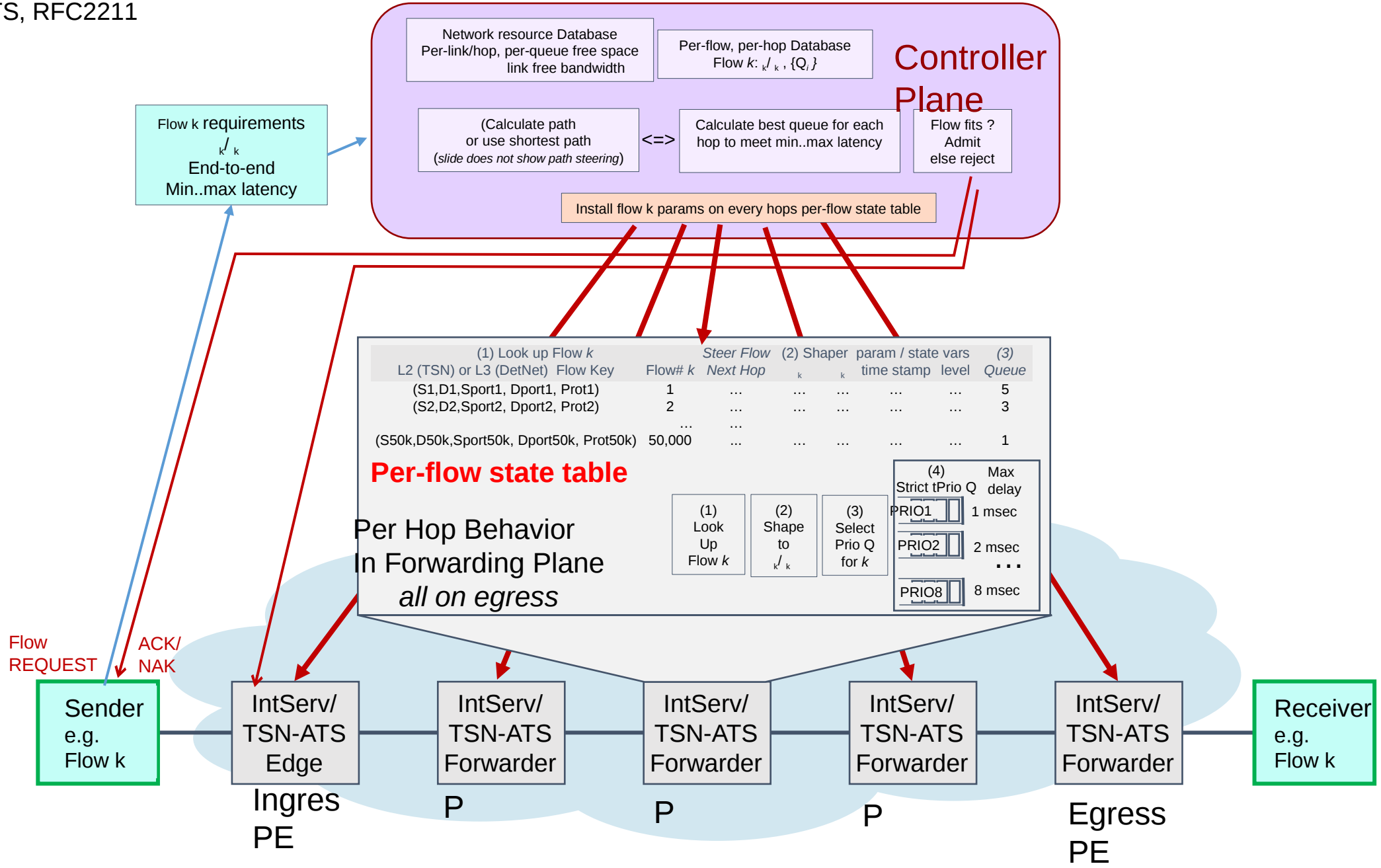
- In Metro-area-Ring networks, every node is can be an edge-node – so all nodes require clock-sync anyhow (with e.g.: TSN-ATS)!

- Aka: TCQF in our opinion is the most easy to build and scale mechanism near-term to support bounded latency and on-time jitter.

Scalability

Current DetNet model: per-flow,per-hop stateful

also TSN-ATS, RFC2211



O(N^2) issue

Realistic reference worst case scenario in large-scale DetNet

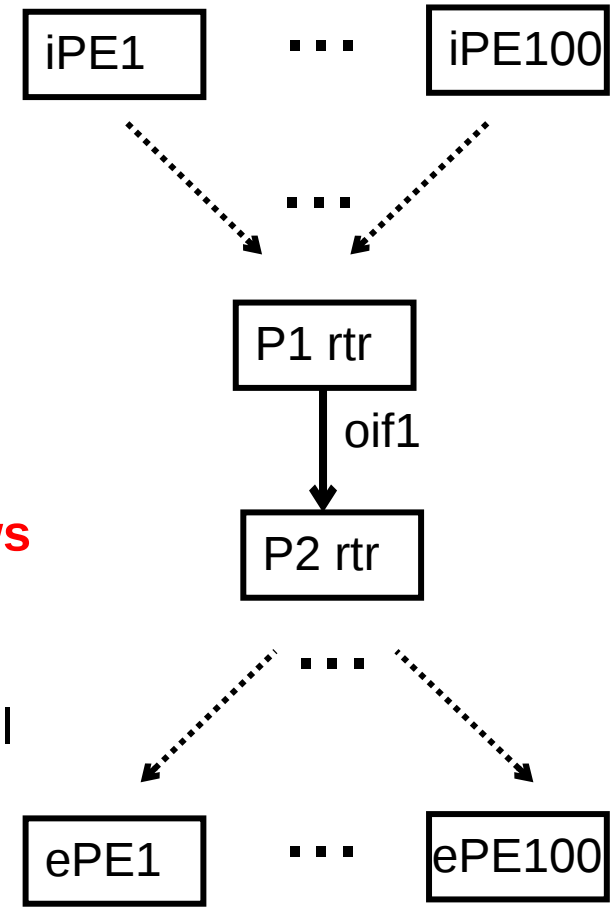
Assume we aggregate all DetNet traffic from one ingres iPEj to one egres ePEk into one aggregate DetNet flow.

- Most edge-aggregation we can do
- j=1...100, k=1...100

Total # flows: j * k = 100 * 100 = 10,000

These flows may all go through one (core PE) interface

oif1 (output interface 1) on P1 in example network



RFC2211 (IntServ): **per-flow shaping for 10,000 flows**

IEEE ATS (Async Traffic Shape): **interleaved regulators for 10,000 flows**

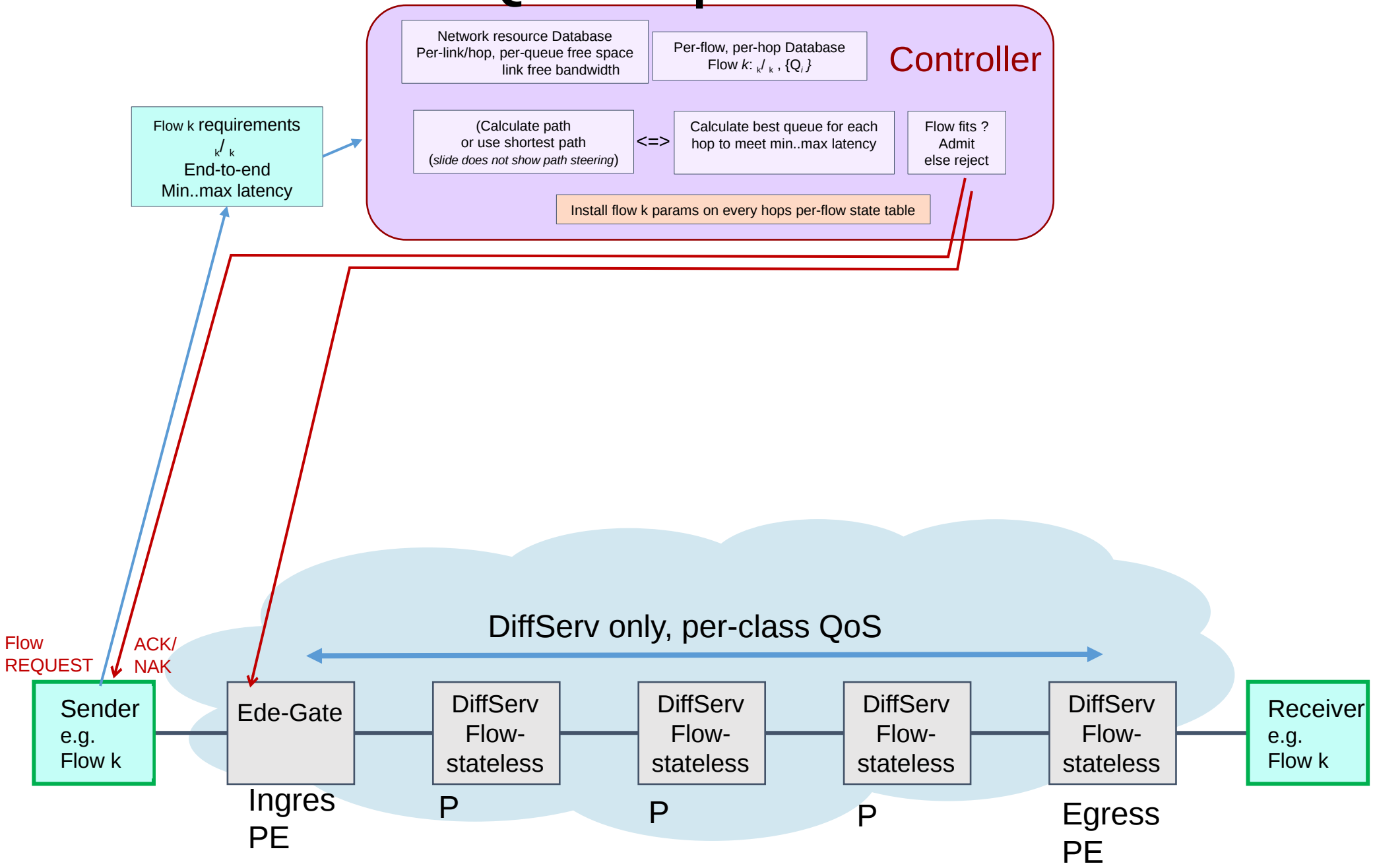
Every time rate and/or burst-size of any of these 10,000 flows changes (because one of its member flow changes):

Both IntServ and ATS require **signaling of new flow parameters** to all routers affected (P1, P2, ...) – *(limited optimizations possible).*

TCQF: **3 ... 5 cyclic queues** on P1 oif1.

No changes in any P router configuration when member flows change!

Desirable DetNet QoS option



Summary

- Metro and larger network operators do not want per-flow, per-hop state on P routers
  - Operational nightmare
    It is why Segment Routing (MPLS IPv6) and BIER (multicast) where created
  - Control plane performance, reliability, scale challenge (updates to each P router)
  - DetNet solutions need to support all those stateless forwarding “traffic-steering” options
- This is not new:
  - RFC2212 was never adopted in SP-networks, but quickly superseeded by DiffServ hop-by-hop
  - But the faster the network, the more hardware challenges per-hop, per-flow has:
  - High-speed routers may not be abble to cheaply implement large number of packet-flow-lookup,data-read/write cycles
- Need to decouple P-router state from applications
  - Can not have P-router state changes (such as aggregated flow-state) when new applications start sending DetNet traffic.
  - Only operator provisioned services currently have this (e.g.: with RSVP-TE), IP Multicast allowed applications to automatically do this – big security/reliability issue
    - But we should want to support automatically self-deployment applications (without ccomplex provisioning steps across core!).

Packet encapsulations

Encapsulation considerations

- Short-term attractive (to simplify initial adoption/deployments):
  - No packet header changes necessary:
  - Cycle-ID can go into DSCP (IP/IPv6) or MPLS TC (Traffic Class) fields
  - Single-domain use: RFC2474, section 6: 16 private DSCP
    - Would need at minimum 3 cycle-ID could have up to 16.
  - Only advanced bounded latency forwarding option for DetNet with this benefit ?!

- Long-term better:
  - Define packet header that explicitly considers Cycle-ID
    - Customers do not like to “overload DSCP” - (“legacy mechanism”, difficult to manage), and TC is quite limited.
  - We should be able to support for packets at least the following:
    - Queuing (TCQF): cycle-ID (and possible extensions)
      - Timestamp (sender) may be desired too
    - PREOF (flow-id, sequence-number)
    - Steering: If not using source-routing (e.g.: SRH), then some type of “aggregate-ID” packet header filed on which steering policy can be configured
  - Can we / should we have multiple extension headers ?
    - Or combine all required DetNet functions into single header ?

Encapsulation options from
from draft-yizhou-detnet-ipv6-options-for-cqf-variant

- Discussion with 6MAN WG:
  - What type of IPv6 extension header to use – HbH or DoH
  - Traditionally DoH would be incorrect
    - Because every router would need to inspect/modify header (Cycle-ID)
    - But DoH would likely allow to pass through un-supporting routers easier...

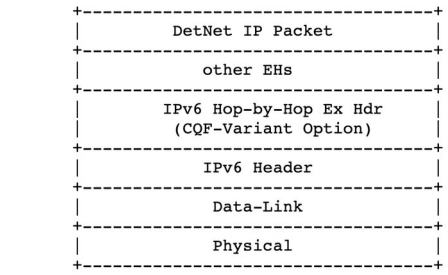


Figure 6: Example 1: CQF-Variant Option Encapsulated in HbH

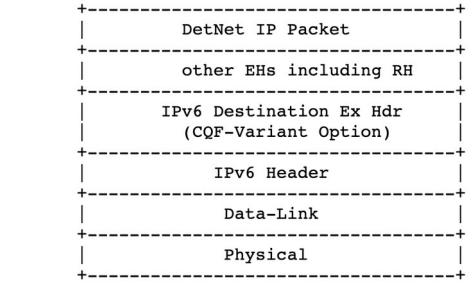


Figure 7: Example 2: CQF-Variant Option Encapsulated in DOH

- DetNet ?! Discussion
  - What fields to have in packet header
  - Example / simple proposal from draft: Cycle-ID + extensions
  - Other options discussed in draft.
  - Should also add fields for PREOF...

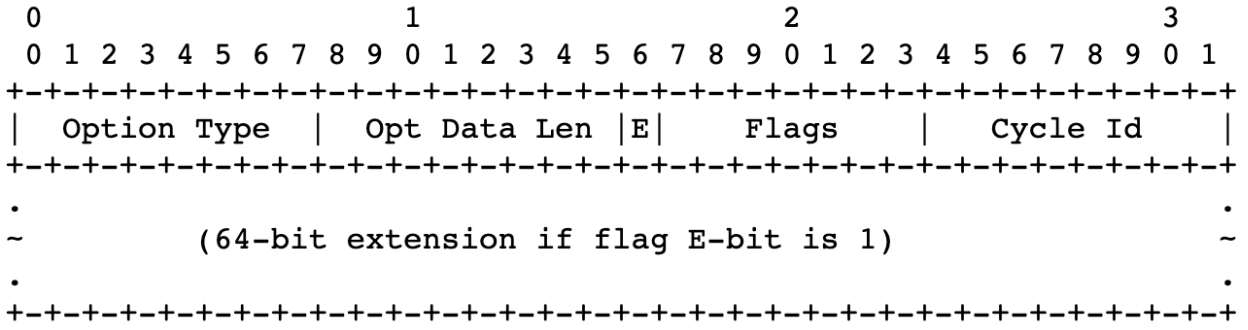


Figure 5: CQF-Variant Option Format Example

Specification outline

What do we (want / should must ?) specify

Specification outline (current target)

Section 1: Overview, motivation, background

Section 2: How TCQF fits into DetNet architecture

But maybe we need architecture extension (P/PE distinguishing) independent of the specification of the mechanism

Section 3: P Mechanism specification – forwarding plane independent

Per-hop configuration model (cycle mapping establishment)

Per-hop (P node) packet processing spec (output queue and cycle ID determination)

Textual and pseudocode

Section 4: Ingress (PE node) per-flow packet processing spec – forwarding plane independent

Configuration model, forwarding specification, initial queue and cycle ID determination:
textual and pseudocode

Section 5:

Per-encapsulation specifications (header rewrite and other considertions – pseudocode, text):

IP/IPv6 DSCP

MPLS TC

Proposed IPv6 extension header - “Deterministic IP” (DIP) extension header ?!

Specification outline (current target)

Section 6: Further considerations

- High-speed implementation optimizations on ingress
- Computation of cycle-mapping at controller
  with link latency, different cycle-clock-offsets
- Support of TCQF across links with different bandwidth
  controller plane issue!
- Controller plane consideration
  applicability to centralized vs. Decentralized controller plane

References

- Validation references,...

DO WE NEED THIS ALL ? IS THIS ALL WE NEED ????

Maybe we should discuss/know this for all proposals ???

Backup / Old slides

CQF with more than 2 buffers (“CQF Variant”)
has the potential to support (3) & (4)

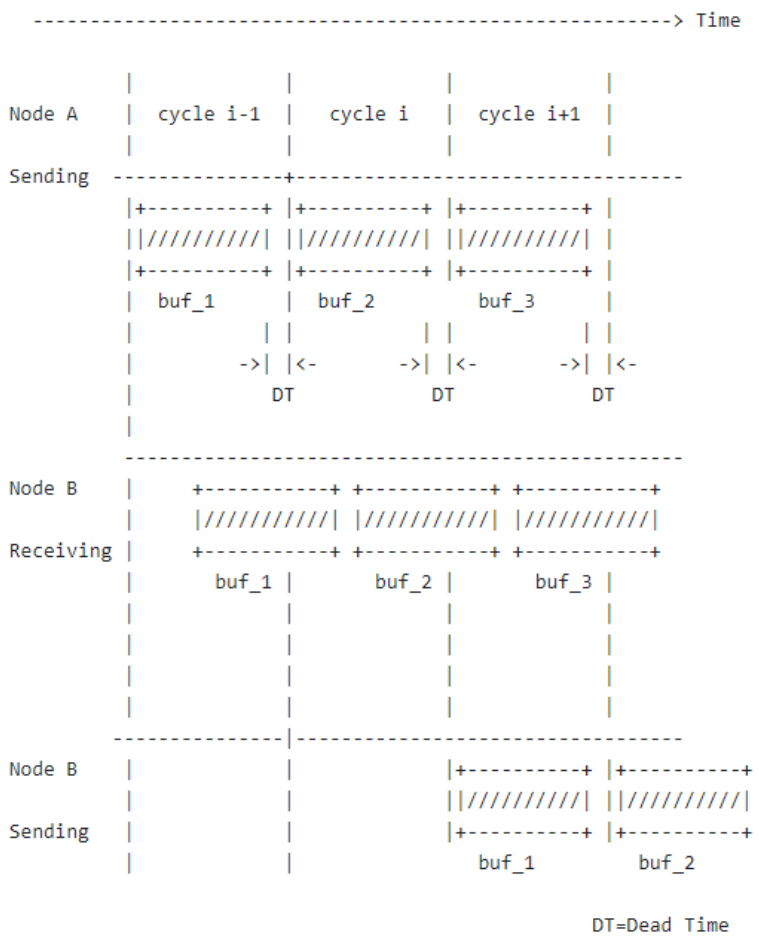


Figure 3: Three Buffer CQF

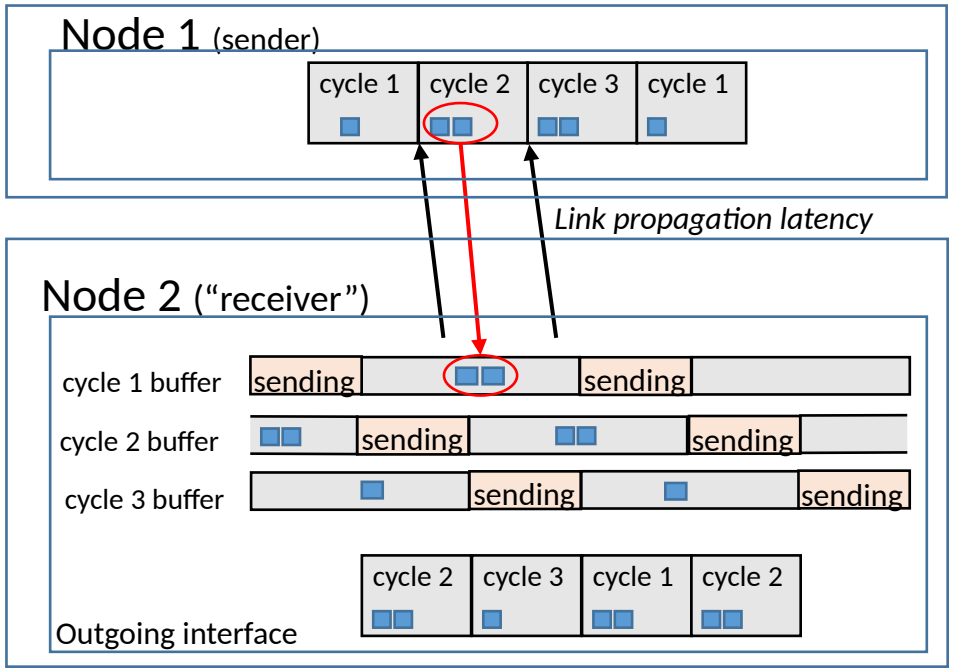
- 3 buffer works in rotation manner
- A straightforward variant to fundamental 2-buffer CQF:
  - Configuration is similar
  - Can easily deduce from fundamental CQF without the rigid requirement to produce new standard
- More than 3 buffer is required when the receiving time spans over two cycle interval boundaries.
- In general, it is feasible.

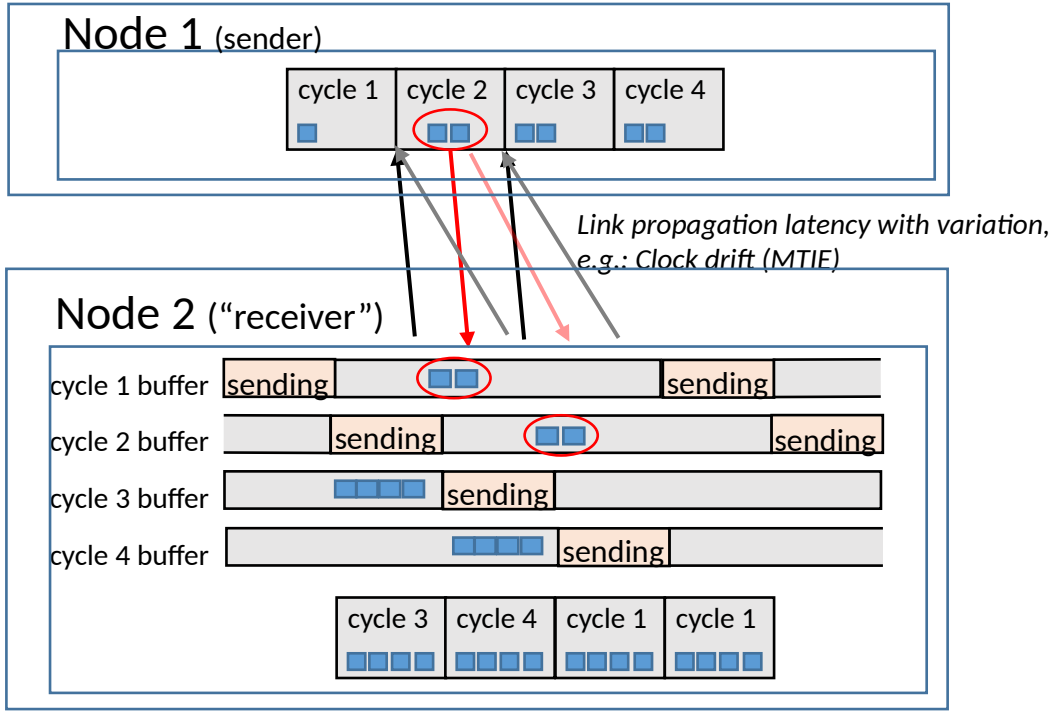
Old slide 1

(from prior presentation)

3 Cycles always suffice as long as variations are negligible

4 Cycles sufficient with limited variations





Old slide 2
(from prior presentation)

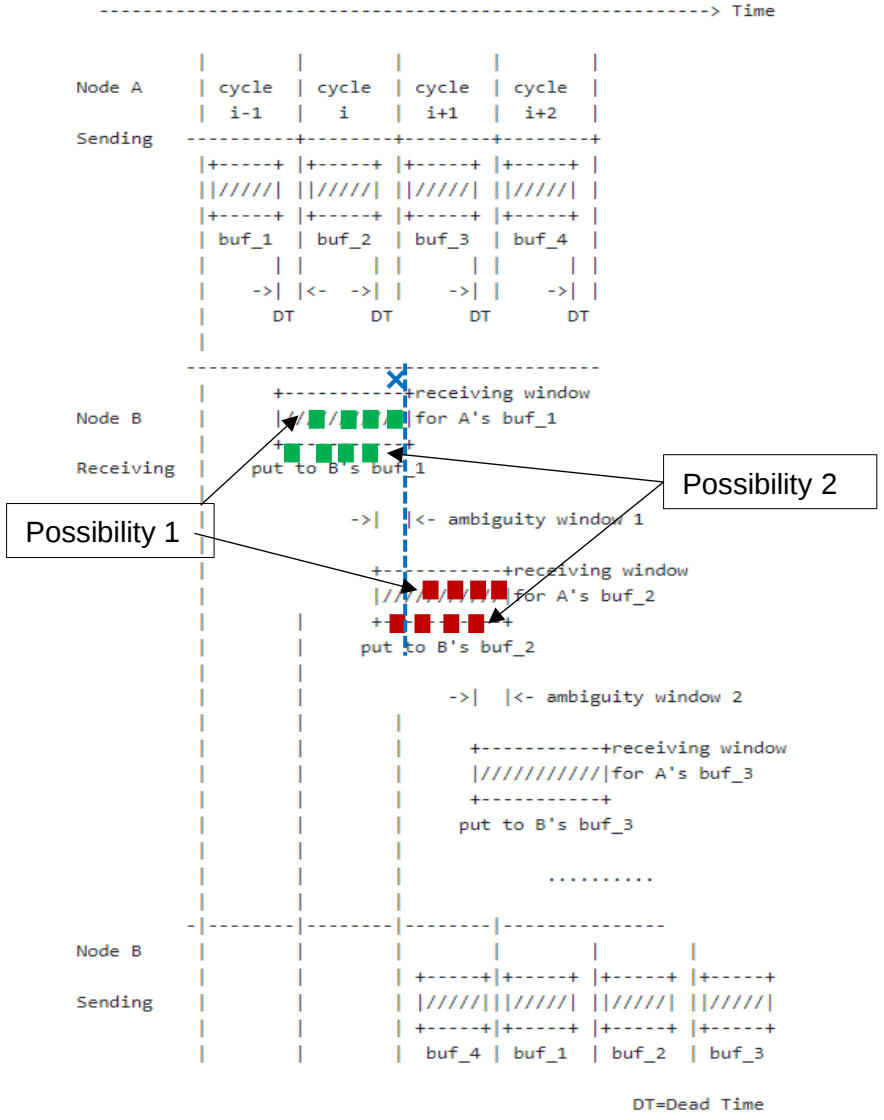


Figure 4: Ambiguity of time based cycle demarcation in CQF

- Time ambiguity window exists for two consecutive cycles – from different sources
  - Node processing delay variation
  - Link propagation latency variation (FEC, link retransmissions, temperature elongation,...)
  - Clock sync MTIE (Maximum Time Interval Error)
- MTIE ~= clock sync accuracy
  Clock sync accuracy requirements increase the smaller TC and DT become:
  - 1... 100 Gbps: 100x more accurate PTP clock
  - 100x Cost factor ?! (not quite but important to watch)
- Conclusion: Relying on reception time to decide on reception buffer is impractical for large-scale deployments
- Way out: pkt carry cycle id metadata at output to help the downstream node determine the correct cycle buffer for the paket