DetNet

Bounded Latency-02

draft-finn-detnet-bounded-latency-02

Norman Finn, Jean-Yves Le Boudec, Ehsan Mohammadpour,
Huawei EPFL EPFL

Jiayi Zhang, János Farkas, Balázs Varga
Huawei Ericsson Ericsson

IETF 103 DetNet WG

Bangkok, 8 November, 2018

A reminder to new attendees …

- DetNet is about an **upper bound** on end-to-end latency – **not** low average latency.
- Bounded latency leads to the ability to compute exactly how many buffers are required to achieve zero congestion loss.
- **Feedback** that slows down flows to avoid congestion is **not an option** for the application space of interest to DetNet.
- Mathematically sound assurances can be given on latency and congestion loss.

Major changes from -01 to -02

- The intent of the document, abstract and section 1, has been clarified.
- Clause 5 was reorganized.
- The queuing model in 7.1, Figure 3, has been expanded to show the regulators, the output queues, and the non-DetNet queues.
- The detailed descriptions of an 802.1Q bridge's queuing mechanisms, in Figure 4 and Figure 5 of -01, have been replaced by simple textual descriptions of frame preemption and time scheduled queuing.
- A mathematical description of IntServ queuing has been added (7.5 in -02).
- A new section 8 has been added to describe time-based queuing techniques including Cyclic Queuing and Forwarding (8.1) and Time Scheduled Queuing (8.2).

Two different problems to be solved

Any given application may be interested in one, the other, or both of:

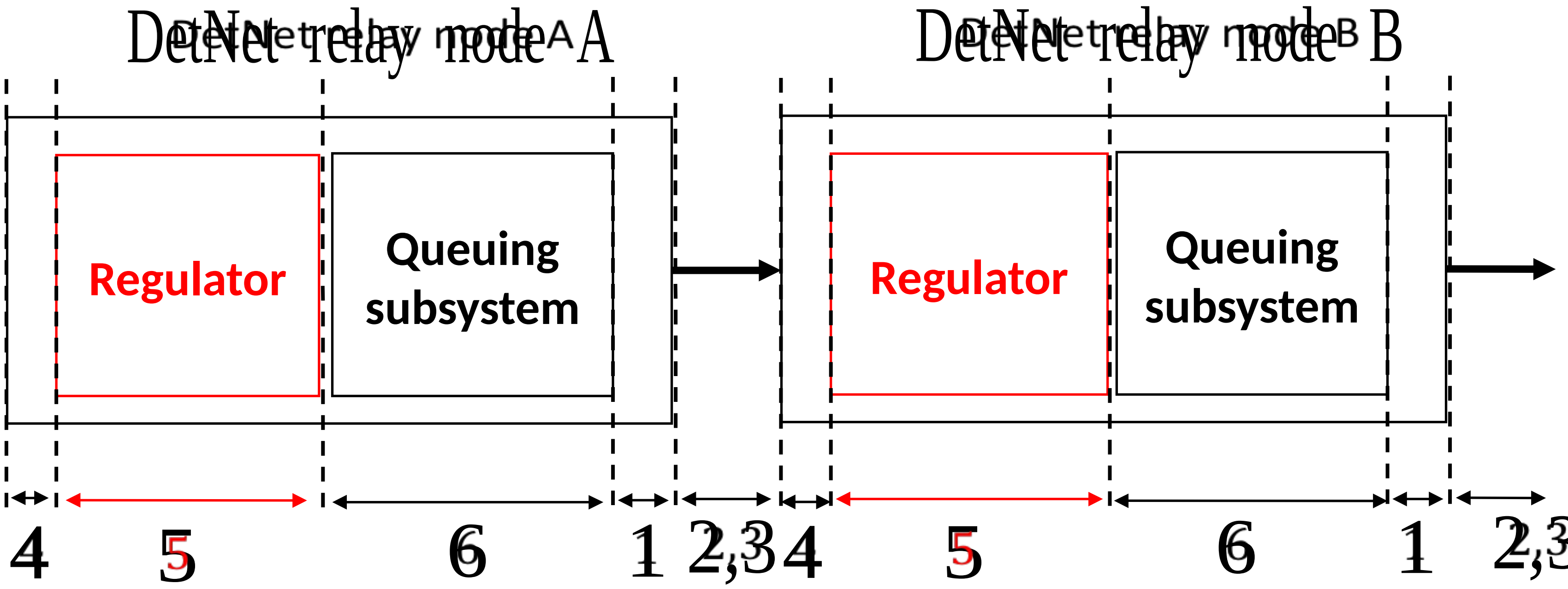
- The **Static** problem: Given the complete set of DetNet flows to be accommodated by a network, their paths and bandwidth characterizations, compute the worst-case latency that can be experienced by each flow, and the buffer requirements in each relay node to guarantee zero congestion loss.
- The **Dynamic** problem: Given a network whose total capacity is limited by some set of configured parameters, and given only one DetNet flow, its path and bandwidth characterization, compute its worst-case latency and the per-relay node buffer requirements that can be guaranteed no matter what other DetNet flows may be subsequently created (subject always to the network capacity).

At present (bounded-latency-02)

- The **Static** problem is described in the mathematical sections of the text (e.g. sections 5.2, 7.4, 7.5).
- The **Dynamic** problem is described in the non-mathematical sections of the text (e.g. sections 1, 4.1, 9.2).
- This will be clarified in version -03.

4.2. Relay system model [updates]

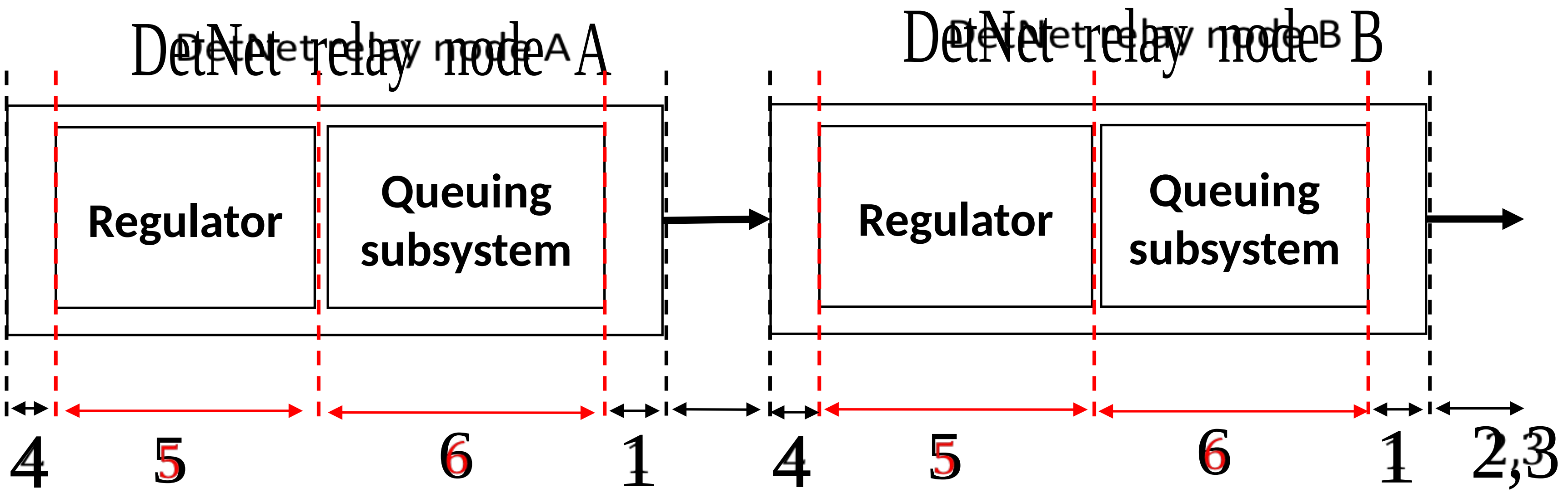
1) Output delay
2) Link delay
3) Preemption delay
4) Processing delay
5) Regulation delay
6) Queuing subsystem delay



End-to-end latency bound calculation

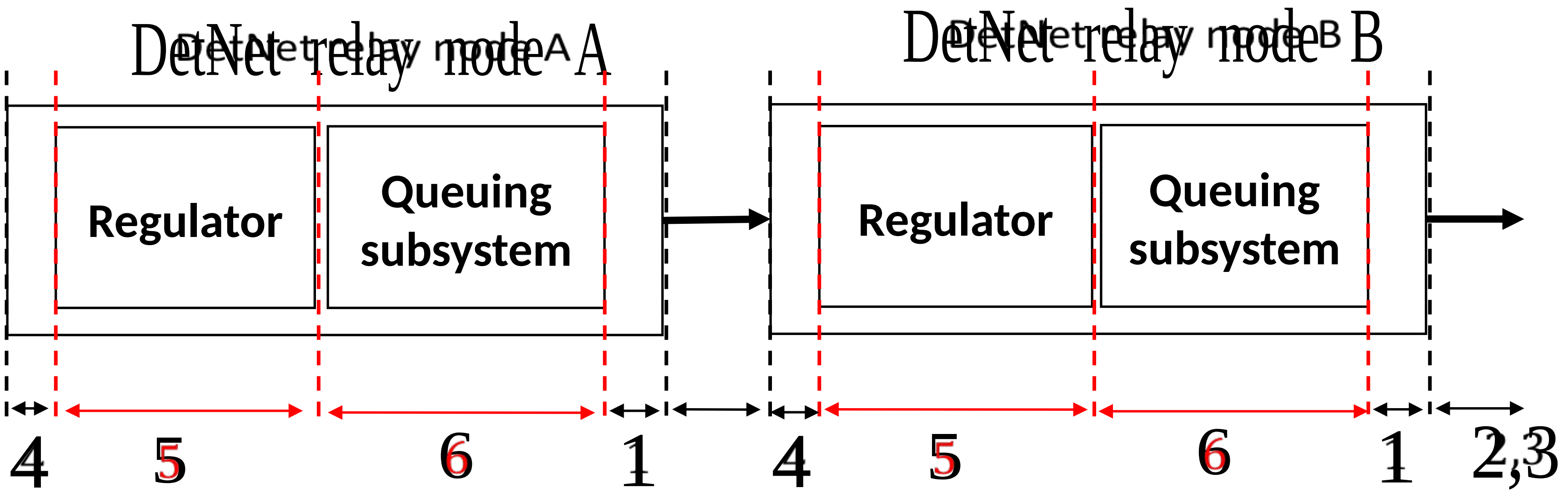
E2E Delay = sum(non-queuing delay) + sum(queuing delay)

= sum (1,2,3,4) at each node + sum (5,6) at each node



Non-queuing delay

- The sum of delays 1,2,3, and 4 at every node.
- An upper bound on it is technology specific.
- An upper bound on it is independent of flow specification.

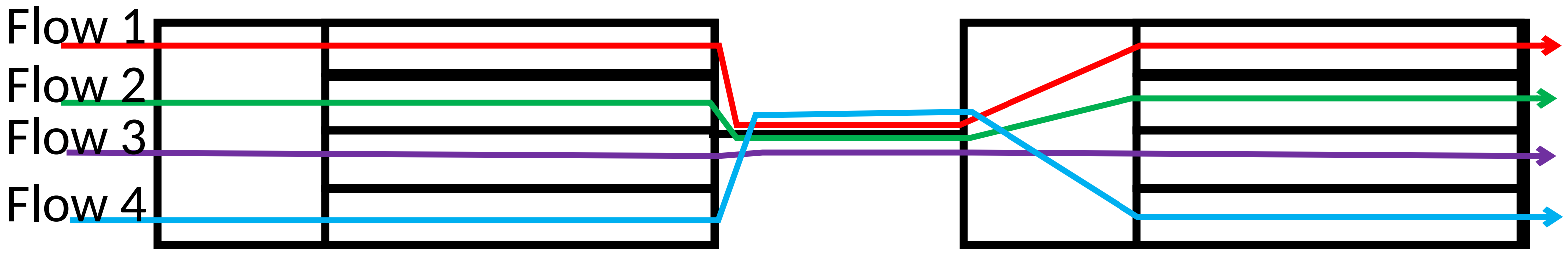


Queuing delay

- Two queuing strategies:
  - **Per-flow Queuing**:
    - Each flow is using its own separate queue.
  - **Per-class Queuing**:
    - Each class of service has its own separate queue.
    - Multiple flows sharing same queue

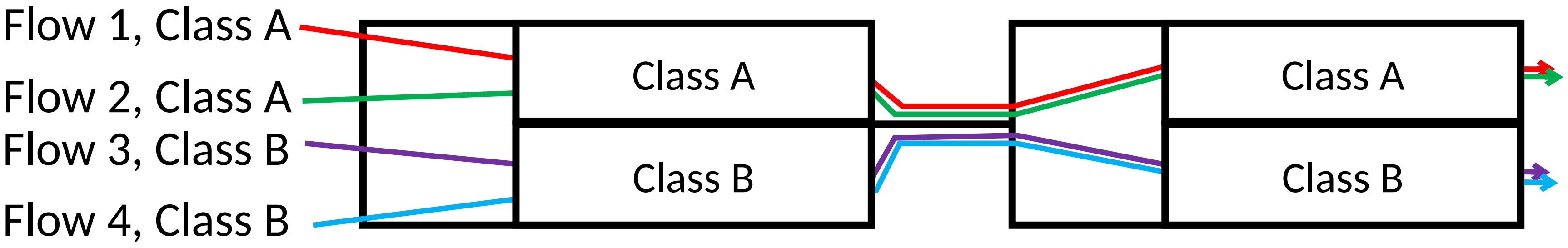
Per-flow queuing

- Separate queue for each flow
- Example: IntServ
- Obtain per-flow per-node and end-to-end delay bound using:
  - Abstraction of a node with guaranteed delay and rate
  - Information on traffic specification for the flow



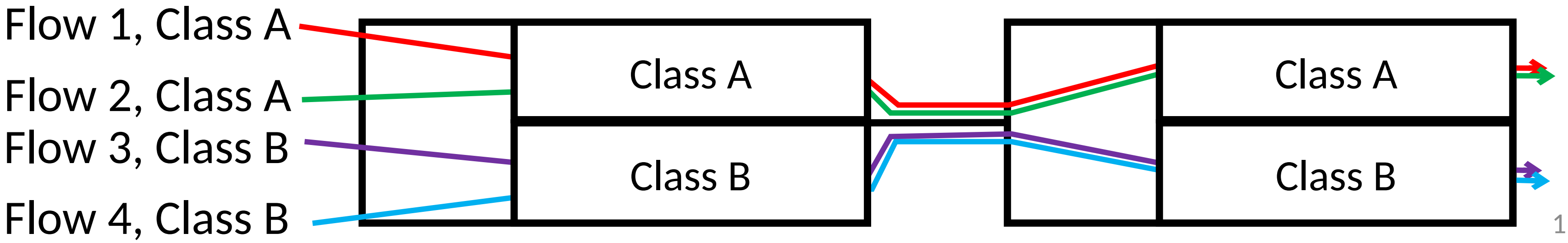
Per-class queuing

- Separate queue for each class
- Example: Time-Sensitive Networking (TSN)



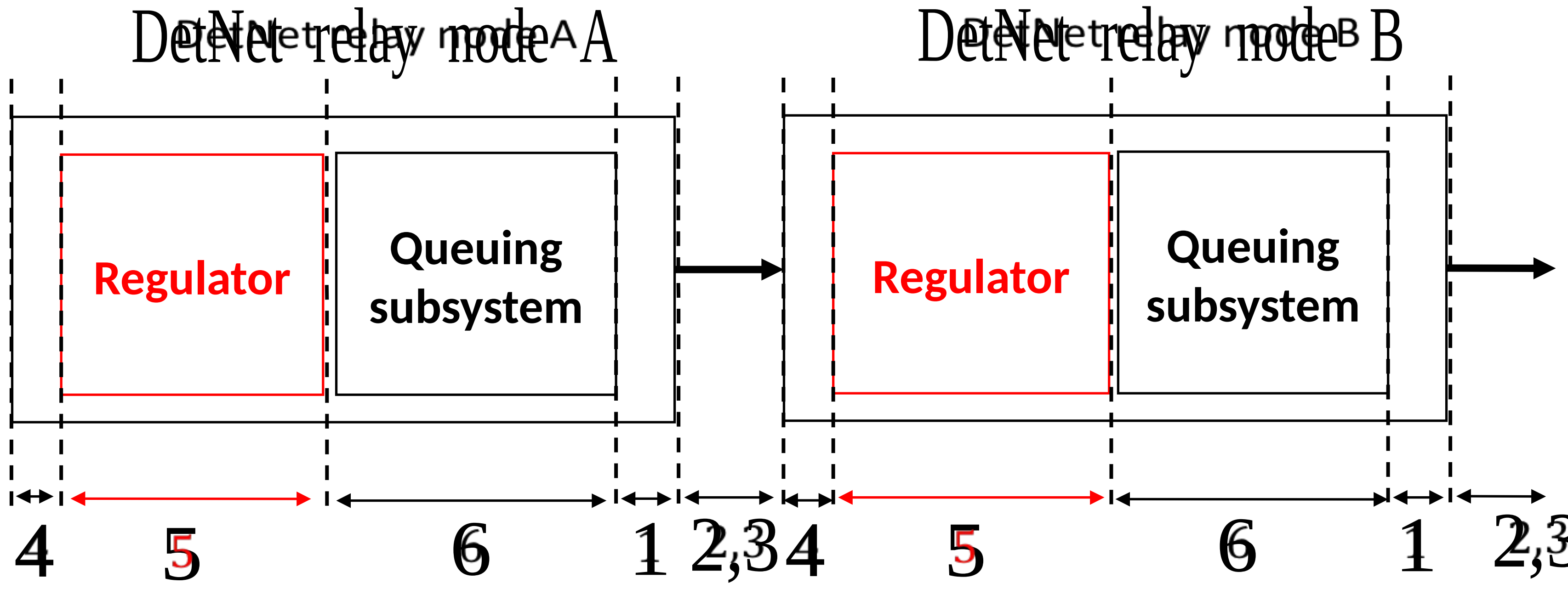
Per-class queuing

- Key issue: **burstiness cascade**;
  - Individual flows that share a resource dedicated to a class may see their burstiness increase.
  - Cause increased burstiness to other flows downstream of this resource.
  - Hardness of calculation of end-to-end delay bound.
  - Even if a bound is calculated, it is dependent to all the flows.
  - Addition of a flow requires recalculation of the the delay bounds.
- Solution: **Reshaping** at every node, like interleaved regulator



4.2. Relay system model [updates]

1) Output delay
2) Link delay
3) Preemption delay
4) Processing delay
5) Regulation delay
6) Queuing subsystem delay

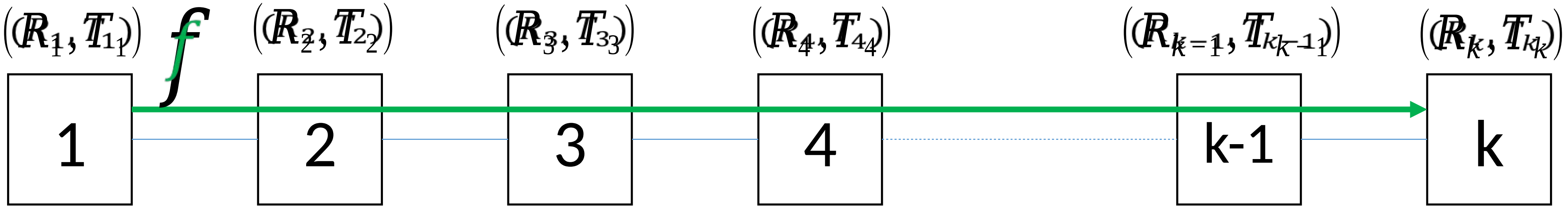


Per-flow end-to-end queuing delay calculation (IntServ)

- Each node i guarantees rate R_i and delay T_i to flow f.
- Traffic of flow f during time t is bounded by $a(t) = r\, t + b$.
- End-to-end delay bound for flow f:

$$D = T + \frac{b}{R} \quad ; \quad T = \sum_{i=1}^{K} T_i \, , \qquad R = \min_{1 \le i \le k} R_i$$

- Each node guarantees rate and delay to flow
- Traffic of flow during time is bounded by .
- End-to-end delay bound for flow :



7.5. IntServ

In Integrated service (IntServ), reservation is made along a path for flows, only if routers are able to guarantee the required bandwidth and buffer. IntServ is an example of per-flow regulation.

  - Input flow conforms to token bucket regulator (r, b)
  - IntServ node provides rate-latency service (R, T)

- Delay/buffer bound is the maximum horizontal/vertical distance between arrival curve and service curve, as shown in the figure.
  - Delay bound =
  - Buffer bound =

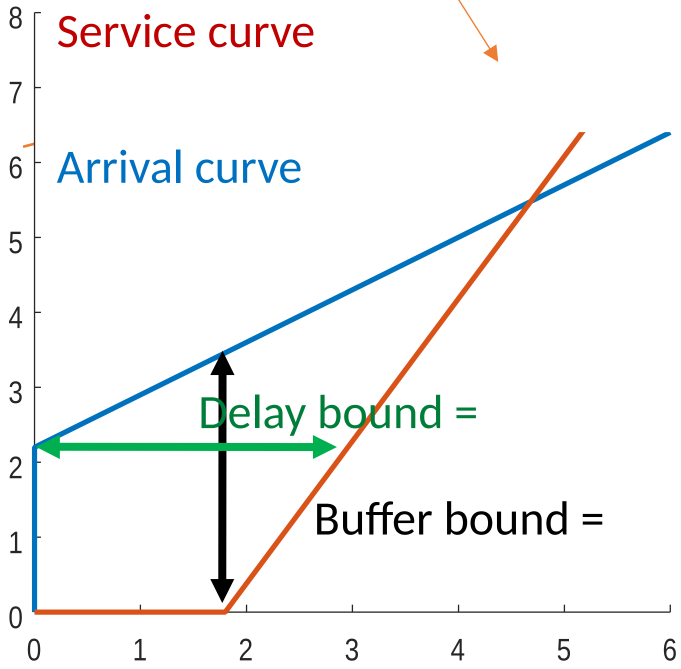
Per-hop bound

- For end-to-end delay bound, we use concatenated service curve
  - Delay bound

End-to-end bound

$$where\ R_{e2e}=min\left(R_1,\ldots,R_N\right), T_{e2e}=T_1+\ldots+T_N$$

R, T: Influenced by other prioritized flow on this node

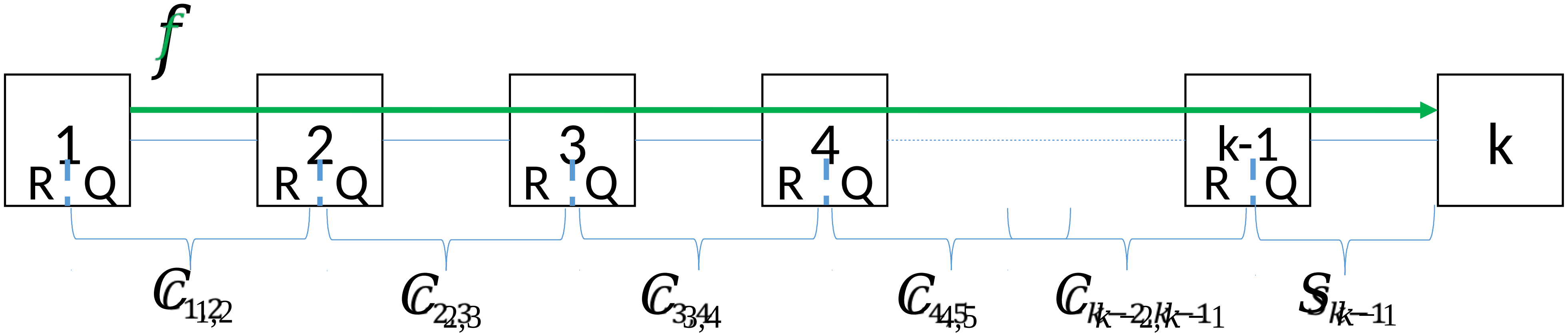


Per-class end-to-end delay calculation (TSN)

End-to-end delay bound for flow :

End-to-end delay bound for flow f:

$$D = C_{1,2} + C_{2,3} + \cdots + C_{k-2,k-1} + S_{k-1}$$



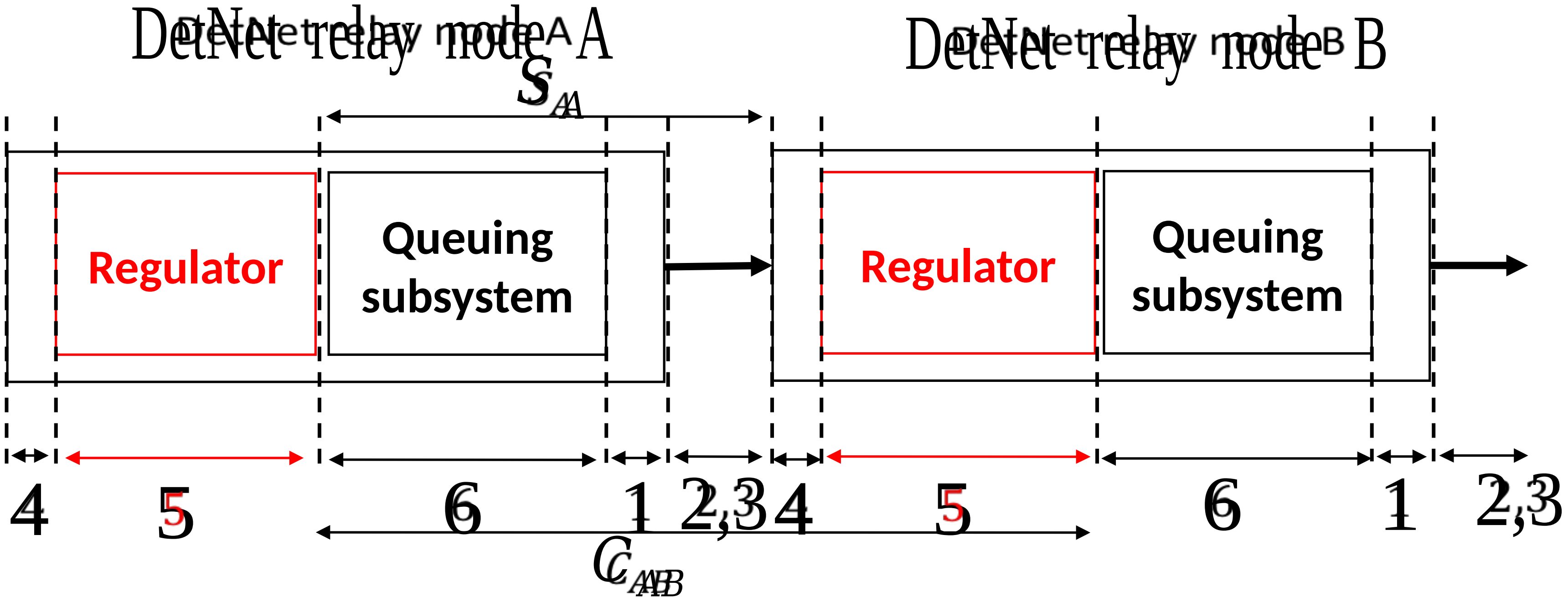
R: Regulator
Q: Queuing Subsystem

Ref: [Mohammadpour, Stai, Mohiuddin, Le boudec, 2018]

Interleaved regulator is for free i.e. does not increase worst-case end-to-end latency!

- Define:
Define:
  - $C_{AB} = \sup\{(6_A + 1_A + 2_A + 3_A) + (4_B + 5_B)\}$
  - $S_A = \sup\{6_A + 1_A + 2_A + 3_A\}$
- Directly From [Le boudec, 2018]:
Directly From [Le boudec, 2018]:

$$\boxed{C_{AB} = S_A}$$

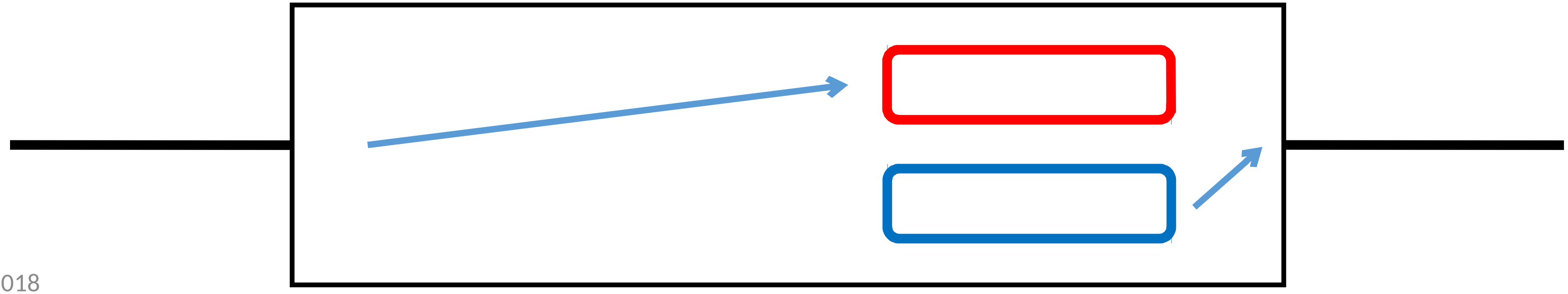


8. Time-based DetNet QoS

- The calculus used in section 7 does not apply, except perhaps at the edges.
- Packets are output according to some kind of repeating schedule.
- Two methods have been standardized in IEEE 802.1:
  - **Cyclic Queuing and Forwarding** (CQF, IEEE Std 802.1Qch-2017).
  - **Scheduled Traffic** (IEEE Std 802.1Qbv-2015, called “time-scheduled queuing” in draft-finn-detnet-bounded-latency-02).

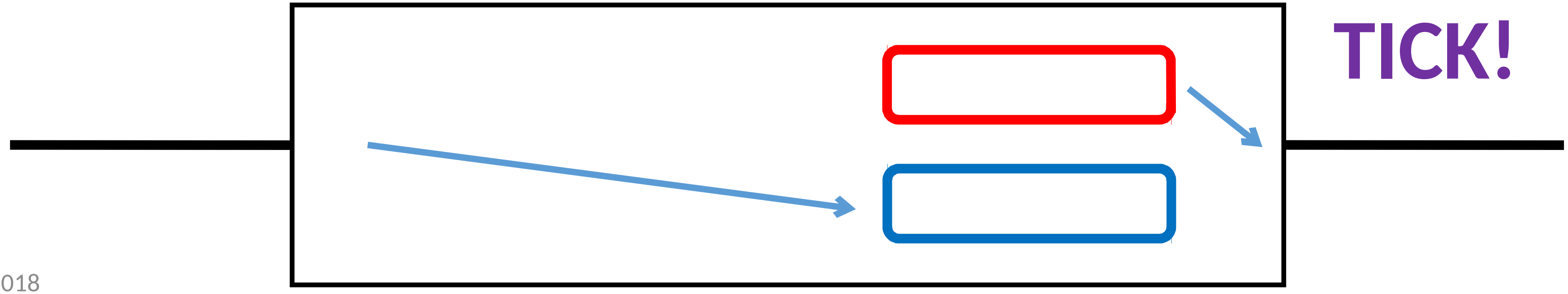
8.1 Cyclic Queuing and Forwarding

- Two-buffer version: Two buffers per port. Input and output buffers swap at the same moment, once every cycle, period T_C. Small guard band to allow for transit and forwarding time. All relay nodes are synchronized and swap buffers at the same moment. Cycle time T_C > transit time + forwarding time + clock inaccuracy + max data transmit time.



8.1 Cyclic Queuing and Forwarding

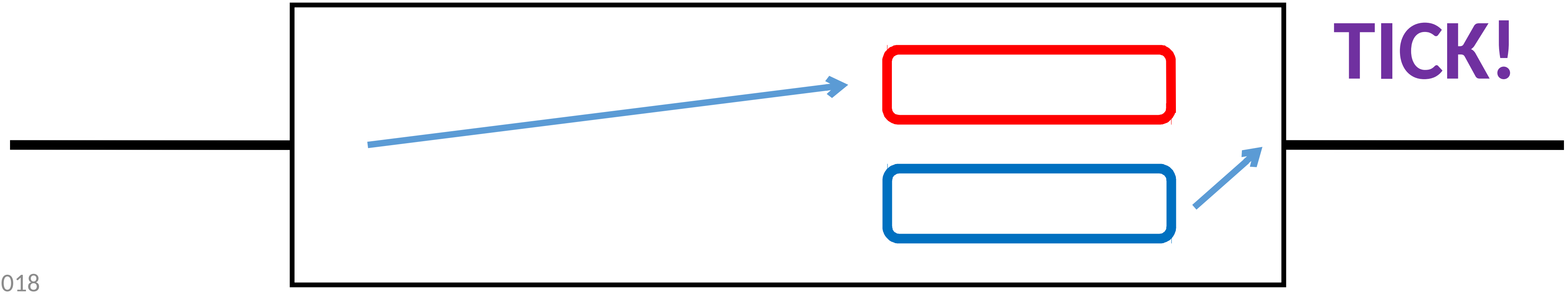
- Two-buffer version: Two buffers per port. Input and output buffers swap at the same moment, once every cycle, period T_C. Small guard band to allow for transit and forwarding time. All relay nodes are synchronized and swap buffers at the same moment. Cycle time T_C > transit time + forwarding time + clock inaccuracy + max data transmit time.



TICK!

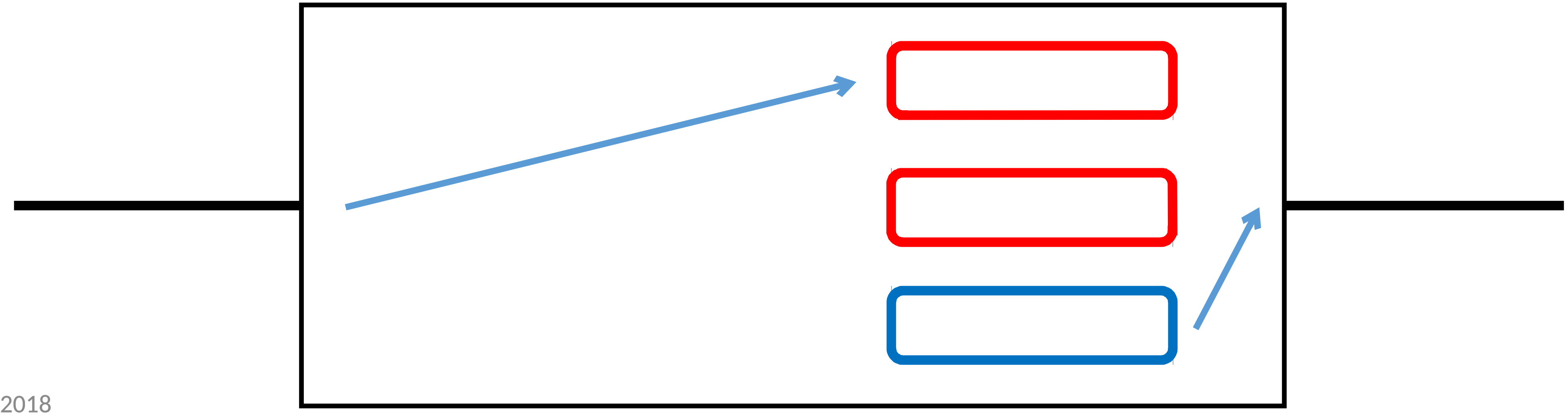
8.1 Cyclic Queuing and Forwarding

- Two-buffer version: Two buffers per port. Input and output buffers swap at the same moment, once every cycle, period T_C. Small guard band to allow for transit and forwarding time. All relay nodes are synchronized and swap buffers at the same moment. Cycle time T_C > transit time + forwarding time + clock inaccuracy + max data transmit time.



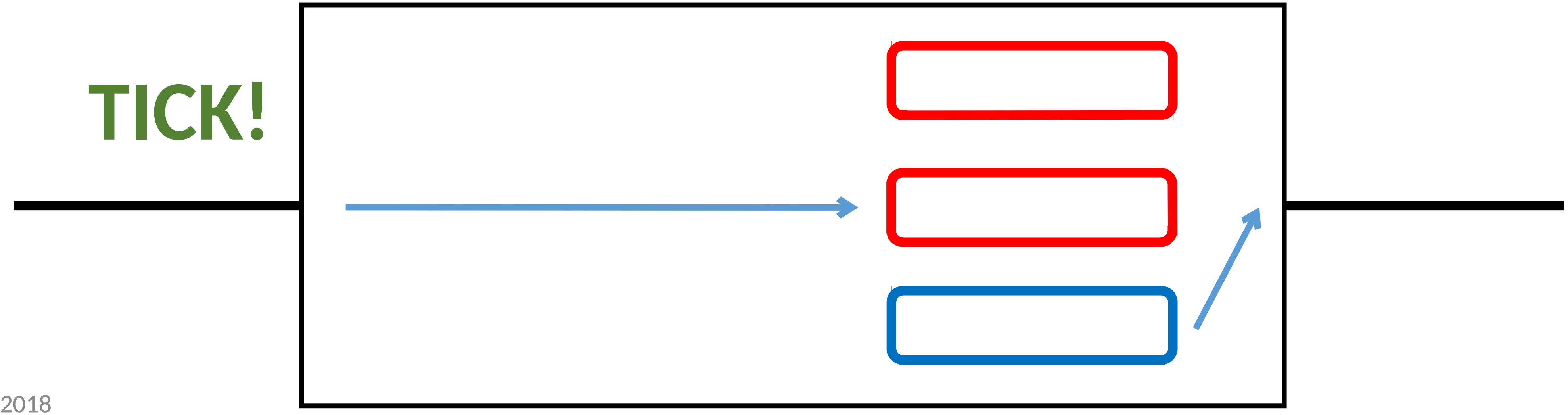
8.1 Cyclic Queuing and Forwarding

- Three-buffer version: Three buffers per port. Same as two-buffer version, but input buffer swap is out-of-phase with output buffer swap to allow for arbitrary link delay.



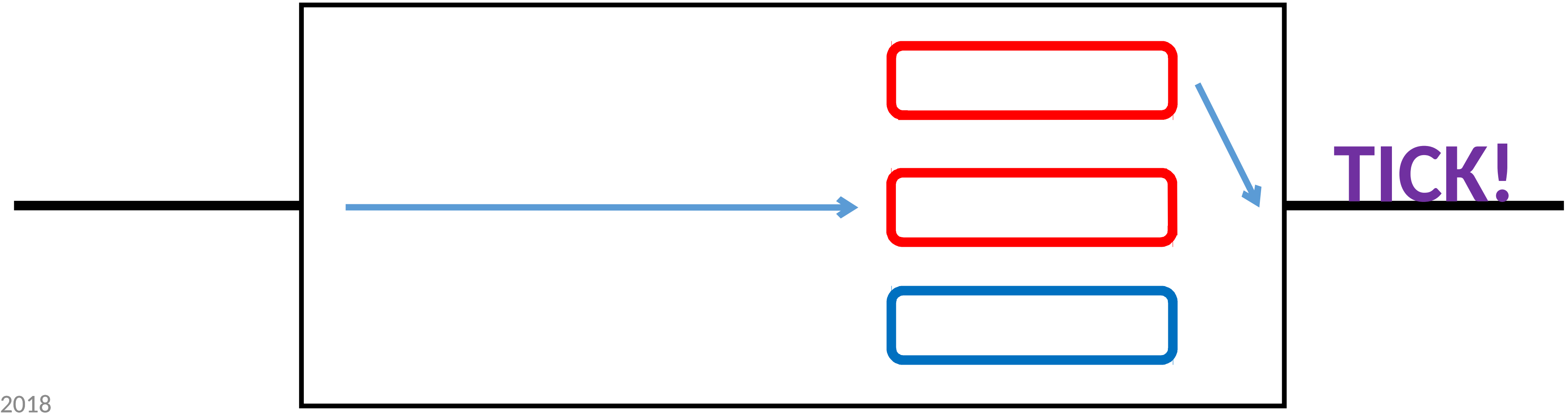
8.1 Cyclic Queuing and Forwarding

- Three-buffer version: Three buffers per port. Same as two-buffer version, but input buffer swap is out-of-phase with output buffer swap to allow for arbitrary link delay.



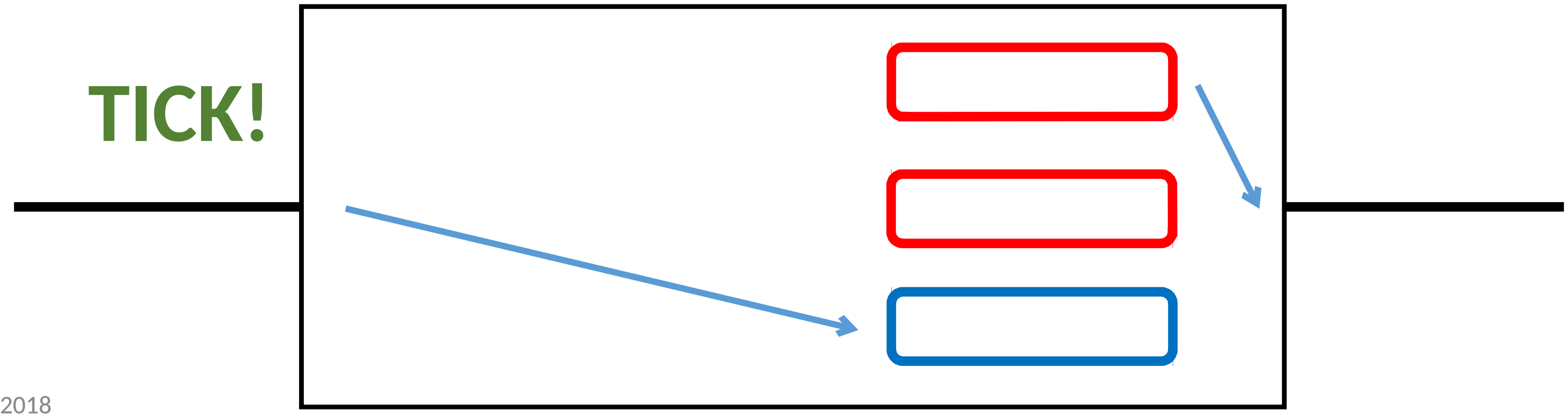
8.1 Cyclic Queuing and Forwarding

- Three-buffer version: Three buffers per port. Same as two-buffer version, but input buffer swap is out-of-phase with output buffer swap to allow for arbitrary link delay.



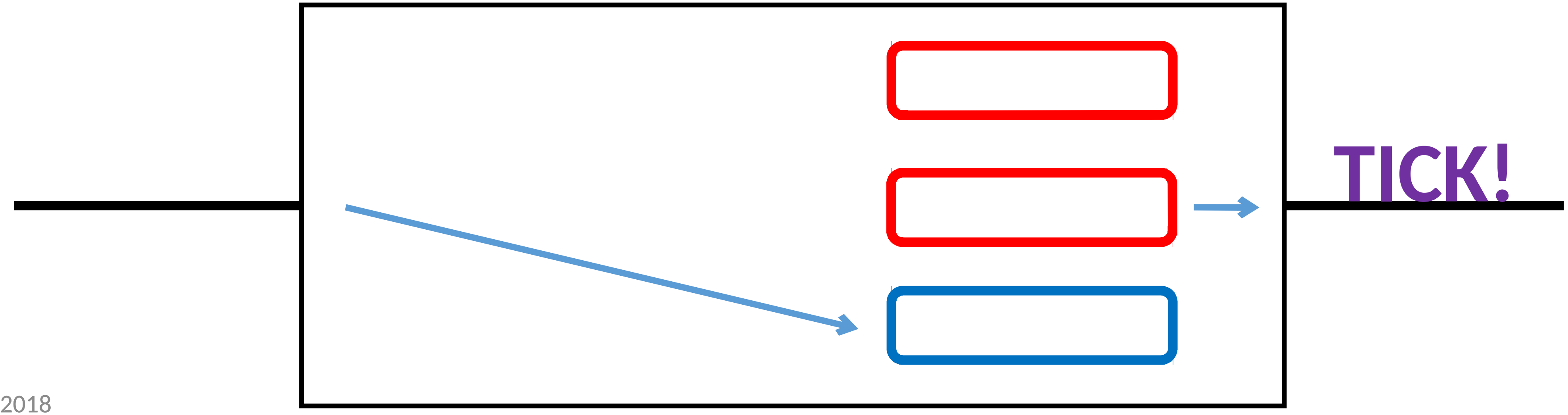
8.1 Cyclic Queuing and Forwarding

- Three-buffer version: Three buffers per port. Same as two-buffer version, but input buffer swap is out-of-phase with output buffer swap to allow for arbitrary link delay.



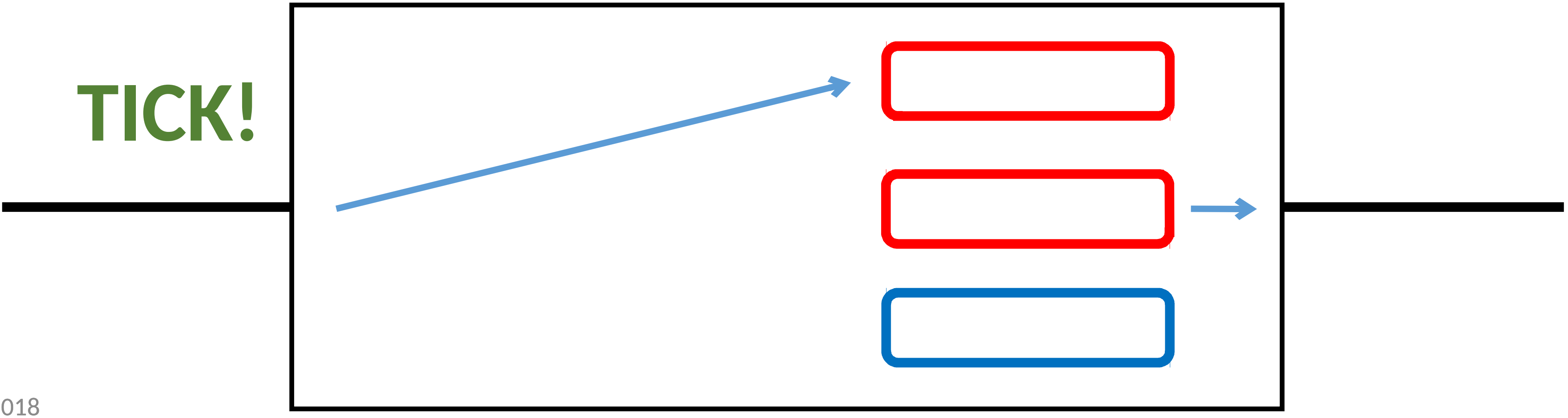
8.1 Cyclic Queuing and Forwarding

- Three-buffer version: Three buffers per port. Same as two-buffer version, but input buffer swap is out-of-phase with output buffer swap to allow for arbitrary link delay.



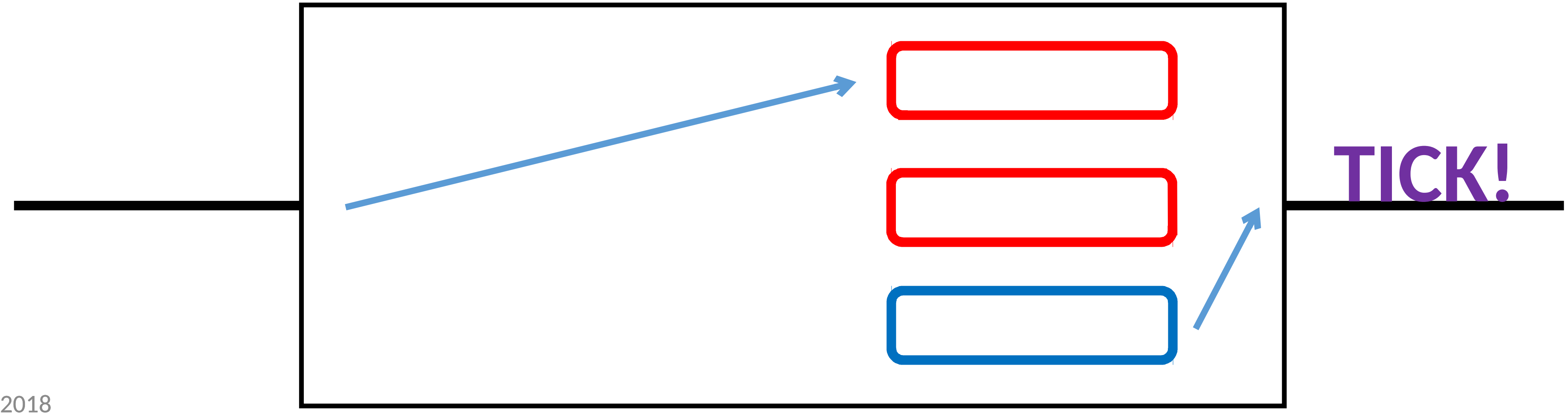
8.1 Cyclic Queuing and Forwarding

- Three-buffer version: Three buffers per port. Same as two-buffer version, but input buffer swap is out-of-phase with output buffer swap to allow for arbitrary link delay.



8.1 Cyclic Queuing and Forwarding

- Three-buffer version: Three buffers per port. Same as two-buffer version, but input buffer swap is out-of-phase with output buffer swap to allow for arbitrary link delay.



8.1 Cyclic Queuing and Forwarding

- Computing the delay is much simpler than the calculus used in bucket/credit schemes: every packet spends two or three cycles T_C at each hop, plus an integral number of cycles T_C (maybe 0) in transit and forwarding delay per hop.
- Resource allocation is trivial: total bandwidth cannot exceed that which can be transmitted in one cycle.
- Multiple buffer sets with different cycle times can run on a single port to supply different classes of service.

8.2 Time-schedule queuing

- Every output queue has a gate, controlled by a rotating schedule with (maximum) 1 nanosecond precision, from a synchronized clock.
- **This solution is different** from all others in the draft, in that bandwidth can be multiplexed in time. DetNet flows are not assumed to run continuously.

8.2 Time-schedule queuing

- Good news: Scheduling every transmission with simultaneous optimization for latency, buffer space, jitter, interference with best-effort traffic, and any other QoS parameter you can name.
- Bad news:
  - 802.1Qbv, at present, supports only 8 scheduled queues.
  - Computing a schedule for the network is an NP-complete problem, although practical algorithms are in use, today.

QUESTION

- It is the intention of the authors that draft-finn-detnet-bounded-latency be adopted by the Working Group, to become normative text for how one can provide the bounded latency and zero congestion loss using already-published standards.
- **Is this draft headed in the right direction?**

References

[1] J.-Y. Le Boudec, “A Theory of Traffic Regulators for Deterministic Networks with Application to Interleaved Regu- lators,” *arXiv:1801.08477 [cs]*, Jan. 2018. [Online]. Available: http://arxiv.org/abs/1801.08477/, (Accessed:09/02/2018).

[2] E. Mohammadpour, E. Stai, M. Mohiuddin, and J.-Y. Le Boudec, “ End-to-end Latency and Backlog Bounds in Time-Sensitive Networking with Credit Based Shapers and Asynchronous Traffic Shaping,” arXiv:1804.10608 [cs.NI], 2018. [Online]. Available: https: //arxiv.org/abs/1804.10608/

Thank you