DetNet

Bounded Latency-04

draft-finn-detnet-bounded-latency-04

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

Jiayi Zhang, János Farkas, Balázs Varga
Huawei Ericsson Ericsson

IETF 105 DetNet WG

Montréal, 22 July, 2019

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 (and vice versa).
- **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 -03 to -04

- Section 3 reorganized—the “reserve before use” paradigm applies to both the static and the dynamic latency computation problems.
- All of the various supported queuing techniques have been made subsections of section 6, “Queuing techniques”.
- The different queuing techniques have been given more equal attention, some enhanced, some shortened.
- Section 8, “Parameters for the bounded latency model”, has been deleted.

Clause 3

Flows are created by:

1. Configure the network.
2. Characterize the flow.
3. Establish the path the flow is to take.
4. Compute the ability of the network to handle the flow and the suitability to the flow's requirements of the QoS offered, e.g. compute latency.
   - The **Static** latency computation: Recompute every flow's latency whenever any flow is added or removed.
   - The **Dynamic** latency computation: Compute absolute worst-case latency once, when flow is created.
5. If satisfactory results, reserve the resources and give the sender permission to start.

Clause 6: Queuing techniques

6.2 Preemption: The transmission of exactly one Ethernet frame can be suspended many times, with critical frames transmitted in each gap.

6.3 Time-scheduled queuing: Each output queue is gated by a synchronized, rotating schedule set by management.

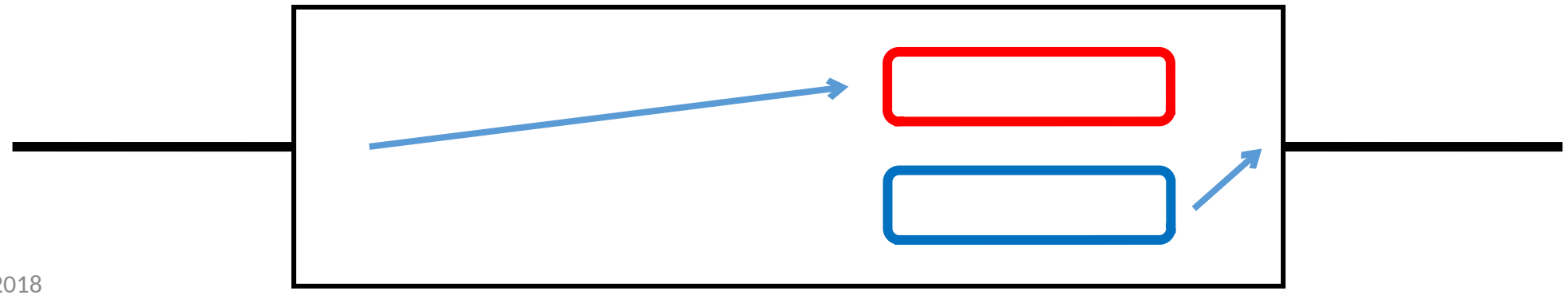
6.4 Asynchronous Traffic Shaping: Hierarchical per-flow and per-class shaping, with fewer than one queue per flow.

6.5 IntServ: Hierarchical per-flow and per-class shaping, without one queue per flow.

6.6 Cyclic Queuing and Forwarding: Double- or triple- buffering for each class on each port, with buffers cycled in synchrony across network.

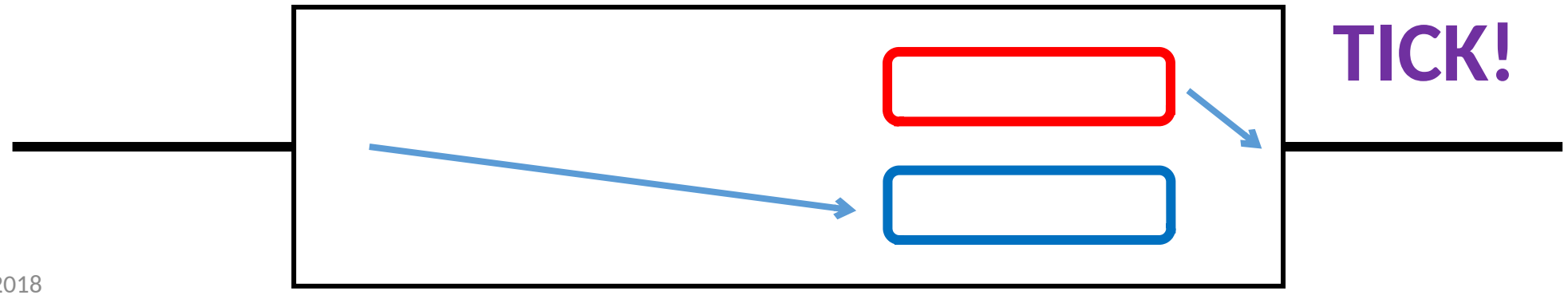
6.6 Cyclic Queuing and Forwarding

- 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.



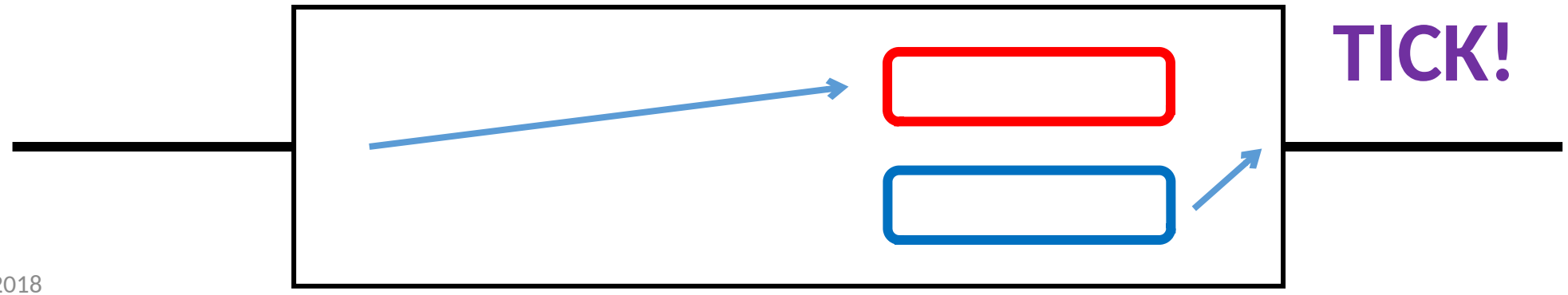
6.6 Cyclic Queuing and Forwarding

- 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.



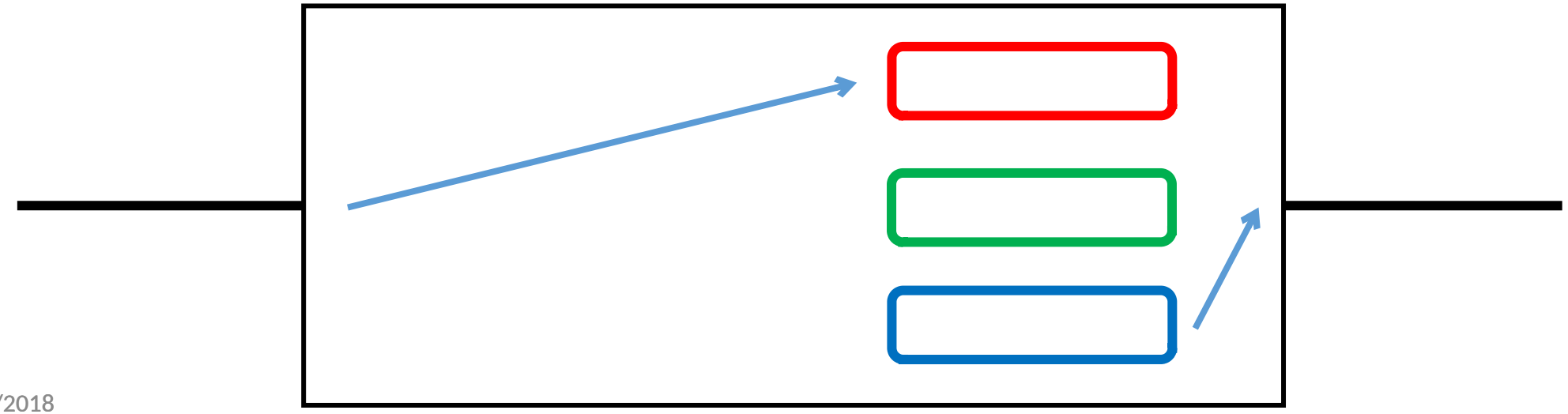
6.6 Cyclic Queuing and Forwarding

- 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.



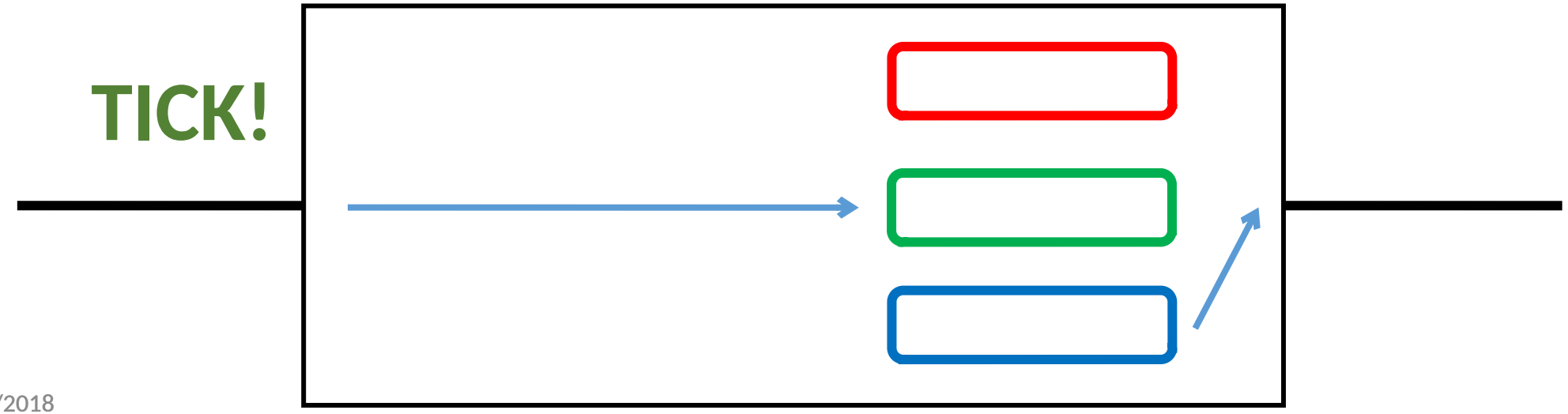
6.6 Cyclic Queuing and Forwarding

- 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.



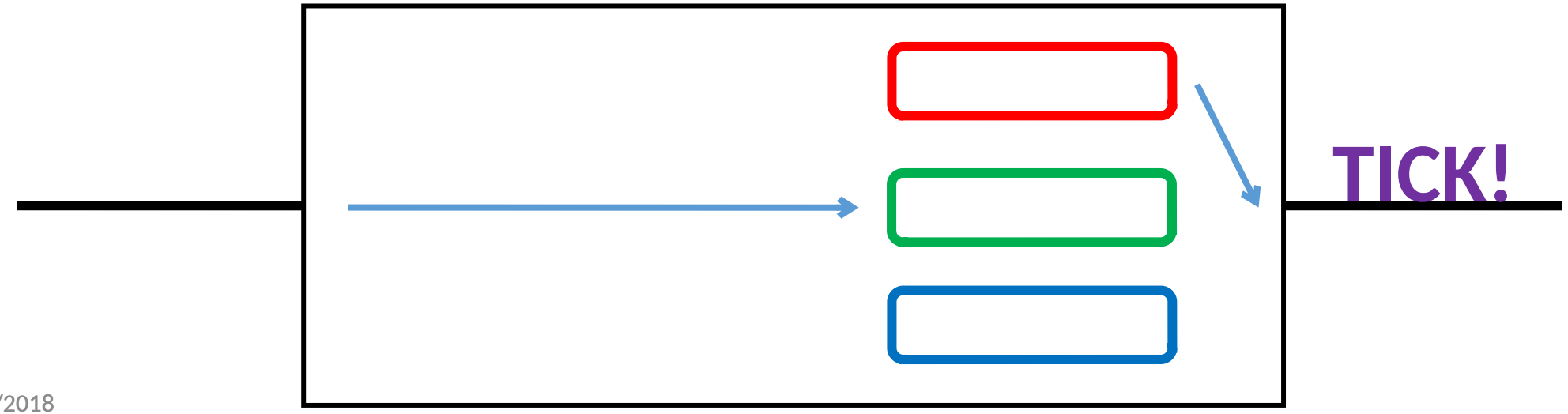
6.6 Cyclic Queuing and Forwarding

- 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.



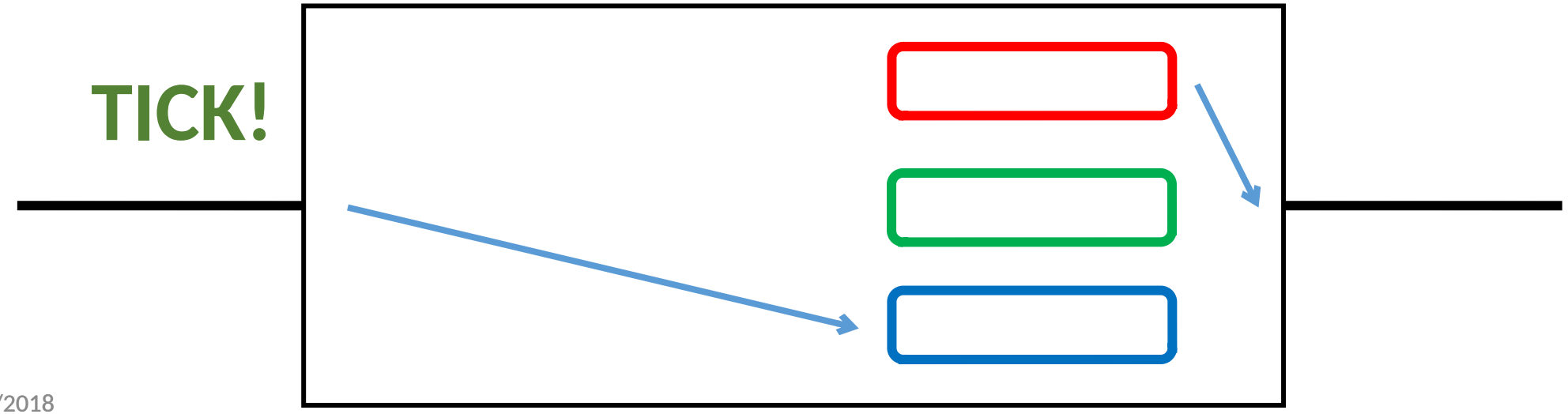
6.6 Cyclic Queuing and Forwarding

- 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.



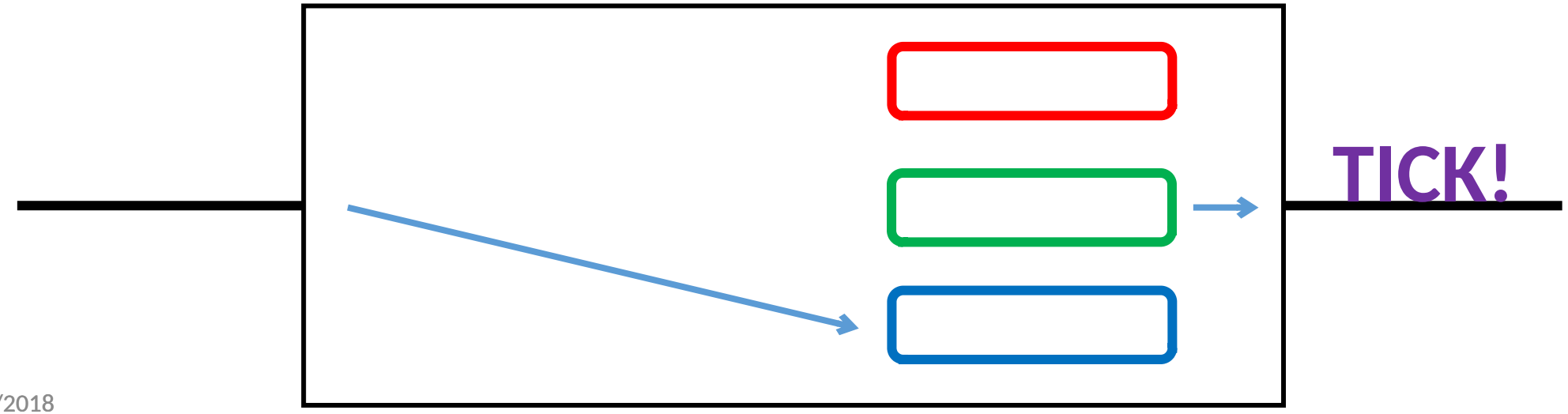
6.6 Cyclic Queuing and Forwarding

- 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.



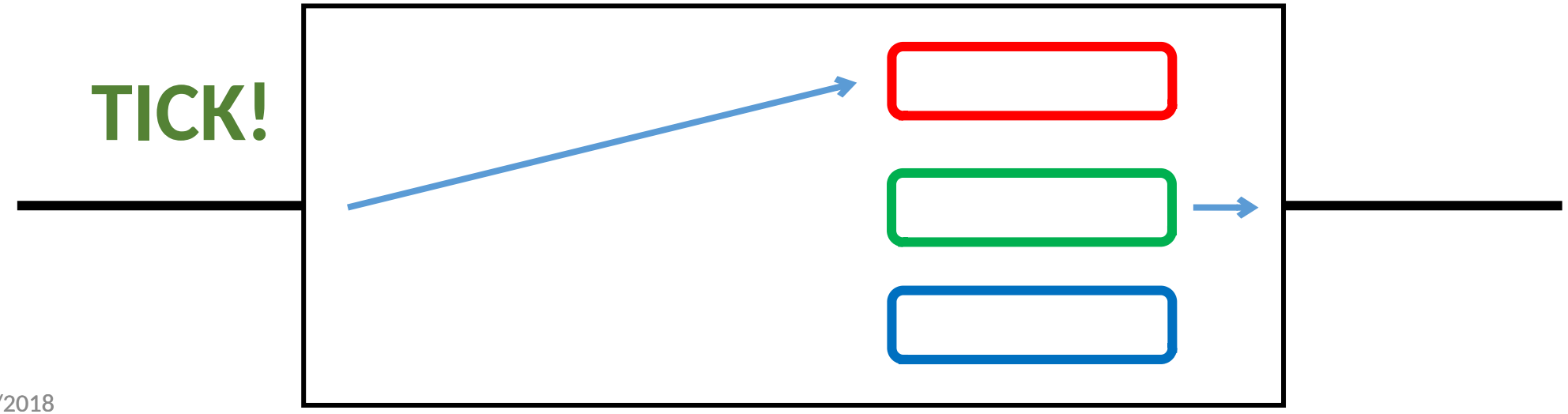
6.6 Cyclic Queuing and Forwarding

- 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.



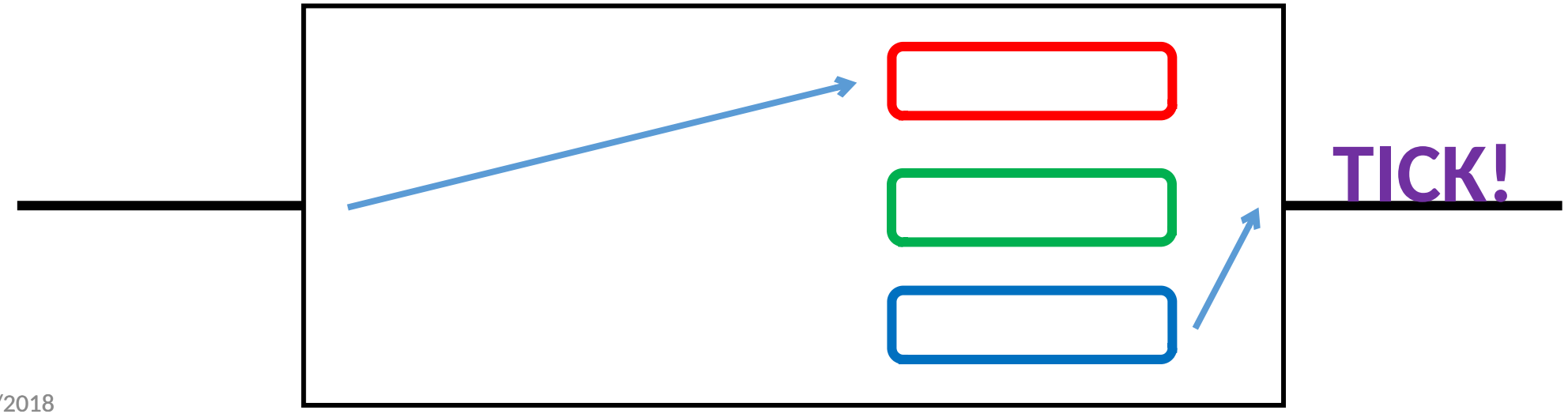
6.6 Cyclic Queuing and Forwarding

- 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.



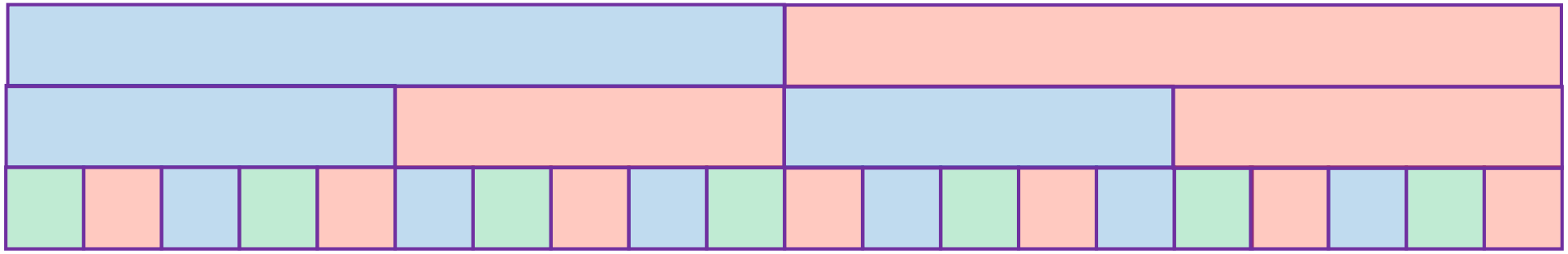
6.6 Cyclic Queuing and Forwarding

- 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.



6.6 Cyclic Queuing and Forwarding

- Time-based CQF is defined in IEEE 802.1 standards.
- Packet-marker based CQF is suggested in private DetNet drafts.
- CQF can be operated at multiple frequencies on one port to serve more than one Class of Service (bandwidth/latency range):



Summary*

Technique	Latency computation	Overprovisioning necessary	Handles predictably bursty flows	State required per-hop	Time sync required
6.3 Time-scheduled	Static NP hard	Small	Yes	Per class schedule	Yes
6.4 IntServ	Static (recompute all flows on any change)	Small	No	Per-flow state, per-flow queue	No
6.5 Time-Aware Shaping	Static (recompute all flows on any change)	Small	No	Per-flow state, per-port-pair queue	No
6.6 Cyclic Queuing & Forwarding	Dynamic (trivial addition)	More	No	None	Yes

* This table is a generalization. There are many factors that can mitigate the differences. Other queuing schemes have been proposed that make other trade-offs.

Final steps…

- Refining the terminology to conform DetNet.
  - Using DetNet terminology and terms.
- Formal delay analysis of CQF.
- Per-node buffer size calculation.
- Consistency check with the other WG drafts.

QUESTION

- Are we ready for adoption?

Thank you