Considerations on Ingress Policing for 802.1Qbv

Johannes Specht, Univ. of Duisburg-Essen

Soheil Samii (soheil.samii@gm.com), General Motors

Motivation

- Ingress policing requirements based on the traffic class
  - *Stream-based* token bucket may be appropriate for traffic classes with credit-based shaping and “best effort” traffic (Markus Jochim, IEEE 802.1 TSN Plenary, Dallas, TX, November 2013 – examples and evaluation already presented)
    http://www.ieee802.org/1/files/public/docs2013/tsn-jochim-ingress-policing-1113-v2.pdf
  - Urgency-based scheduler (UBS): Ingress policing is a built-in property of the shaper (automatic threshold enforcing at egress)
    http://www.ieee802.org/1/files/public/docs2013/new-tsn-specht-ubs-perfchar-1113-v1.pdf

What about the other traffic classes?

- Time-aware shaper: bandwidth only; no policing in time domain is currently defined – examples to follow in this presentation

About this slides

Content

- The next slides show multiple error cases and possible countermeasures, i.e. mechanisms of ingress policing for 802.1Qbv.
- The mechanisms are far from being complete – more could be done on layer 2 (protection of 802.1CB, …).
- The mechanisms are not mapped on yet known/standardized mechanisms but focus on what appears reasonable on layer 2 w.r.t. 802.1Qbv. Mapping can be discussed at the end of this slide set.

Note on Cut-through and Store & Forward

- The figures in this slide set show cut-through behavior for simplicity. The explained mechanisms are applicable for both, store & forward and cut-through bridges.

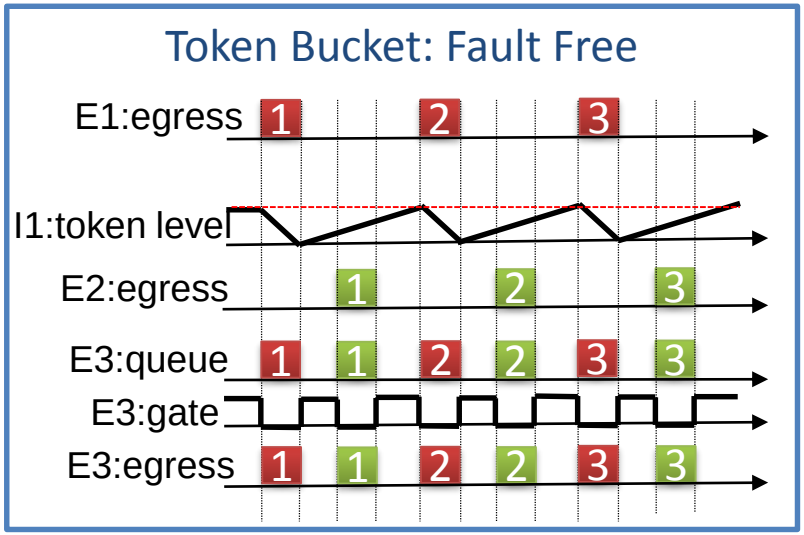
More about this slides

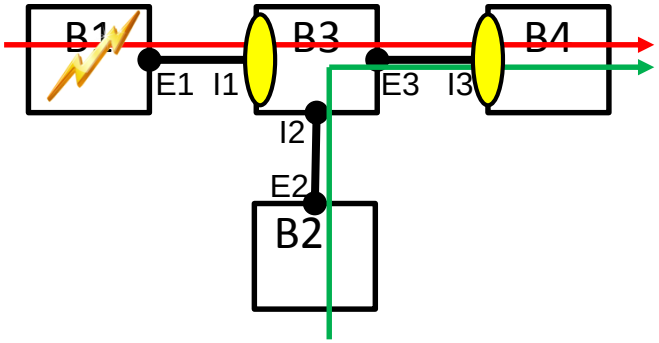
Goals, Anti-Goals and Assumptions

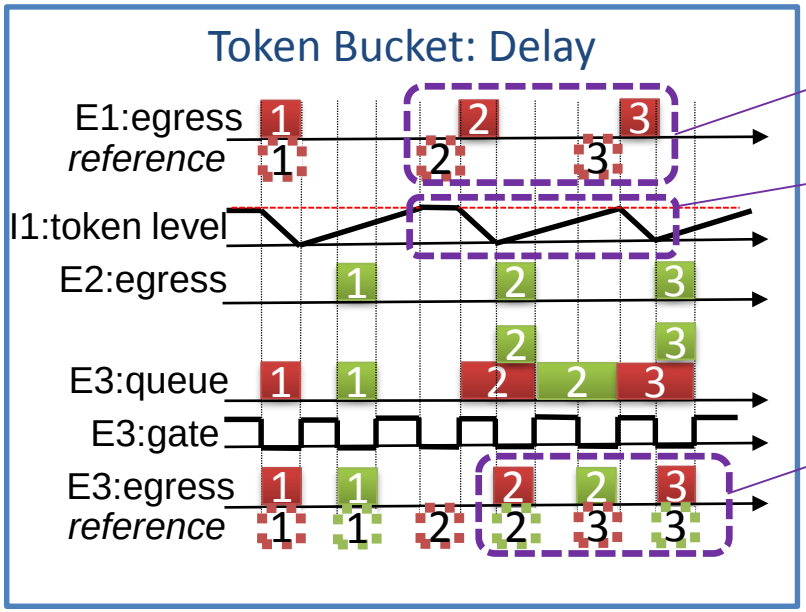
- The goals of the mechanisms is to:
  - Entirely prevent congestion/disruption of fault free streams by faulty streams
  - Enable unambiguous detection of faulty devices/prevent false positive detection
- It is assumed that a faulty box (end-station or bridge) send‘s arbitrary data at arbitrary times (babbling-idiot).
- It is not assumed that some faulty transmissions are more “unlikely” than others, nor that some boxes fail “more unlikely” than others, etc.
- It is assumed that at most one box can fail at a time (single fault assumption).
- It is not a goal to “magically repair” faulty streams. These are considered as broken, faulty, non-trustworthy, non-repairable, lost [PERIOD]

WHAT DOES NOT WORK FOR 802.1QBV

Token bucket alone does not work for TAS







Delayed Packets

Token limit reached, but this does not affect delayed packet acceptance

Delayed packet 2 of B1 (faulty) congests the queue: Packets 2, 2 and 3 sent in wrong windows

WHAT MAY WORK FOR 802.1QBV

Part 1 - Timing

1. Ingress Windows

Extend the 802.1Qbv gate-states by an ingress open/close flag, i.e. ingress gate:

- Open: Accept consecutive started packets until next ingress close
- Close: Discard consecutive started packets entirely

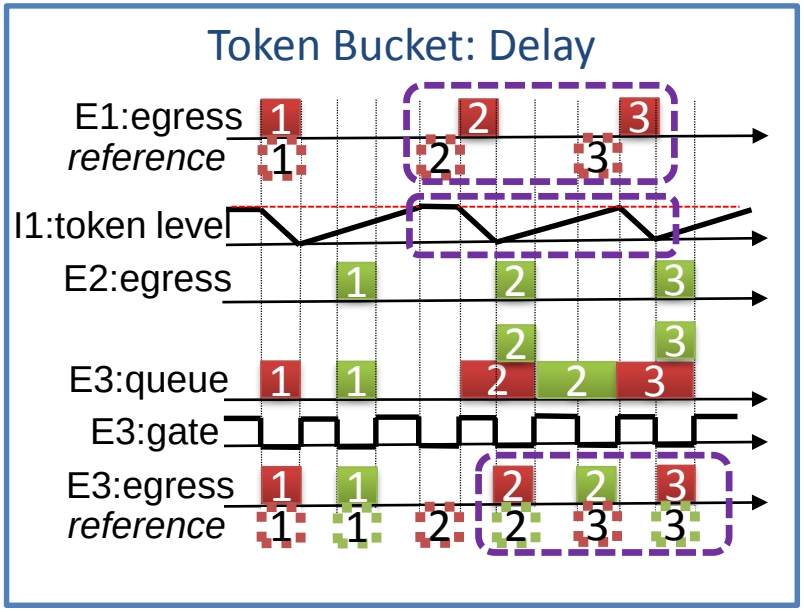
Implication:
Common time for egress and ingress operation at the same port

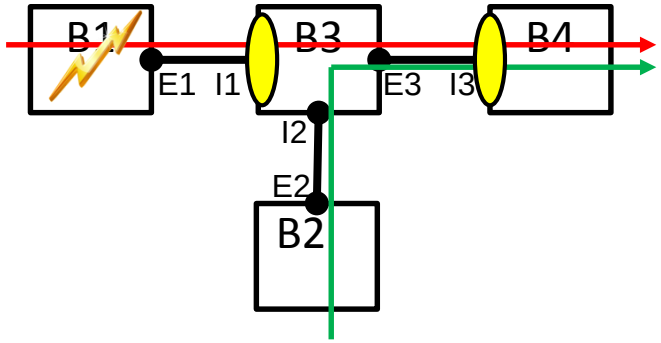
2. Octet Limits

Add octet limits associated with ingress windows and common octet counter:

- Increase octet counter by octets of packets started after transition to open until associated octet limit is reached
- Cut through: Discard octets octet limit is exceeded
- Store and Forward: Discard packet if octet limit is exceeded
- Clear octet counter and current octet Limit at transition to close

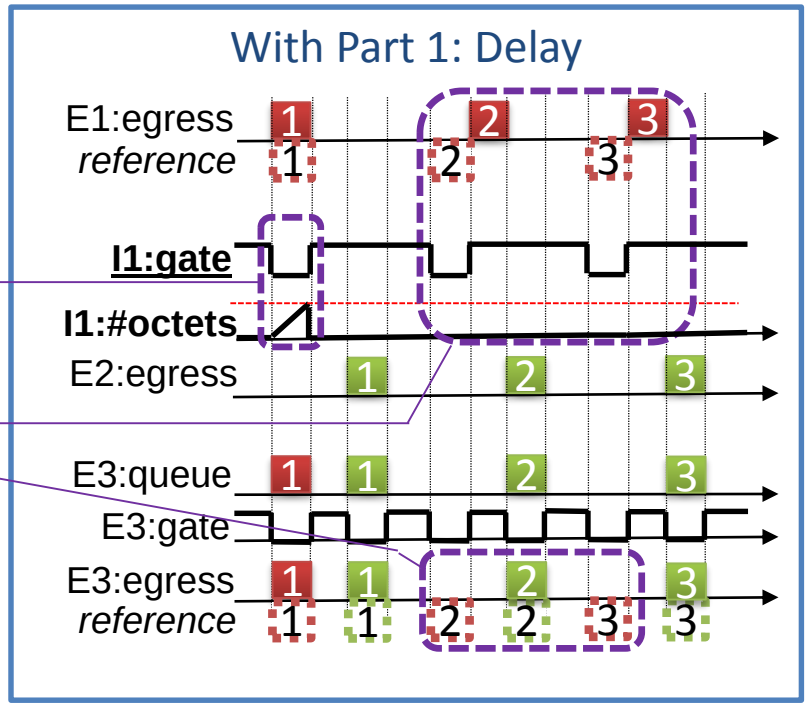
Ingress windows





Ingress open → Accept packet 1
Octet count increased by packet 1
Ingress gate closes → Sets octet count to 0

Delayed packets 2 and 3 arrive during closed ingress window → Entirely discarded



Ingress Windows vs. Octet Limits

Both needed

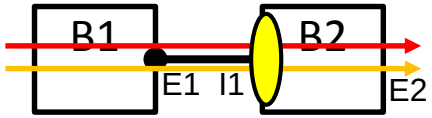
Ingress windows (receiver) must be wider than egress packets (sender) to avoid false positive reactions:

- PTP clocks are not 100% equal, even in the fault free case
- 802.1Qbv implementations may „narrow“ the configured event times
- Allowed variances of packet/octet duration (+-100ppm or more), preamble length, etc. before being rejected otherwise
- …

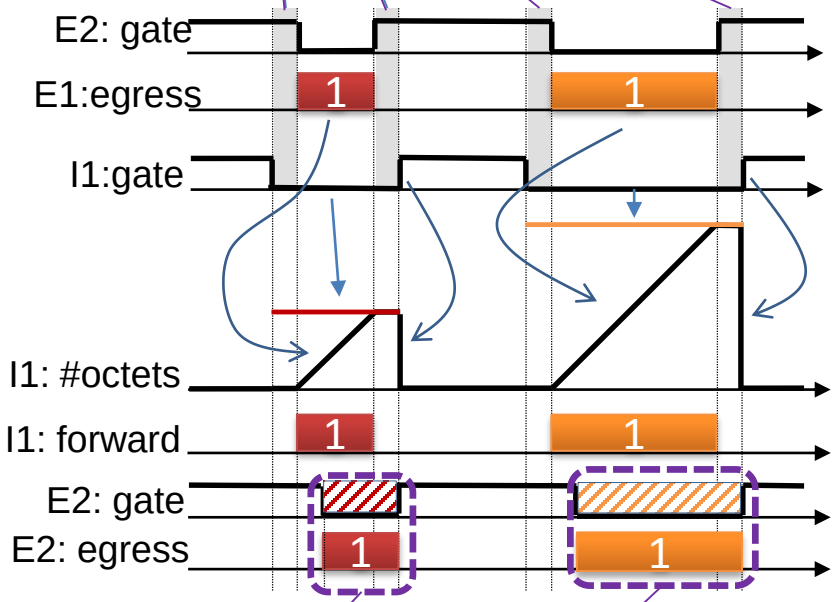
In case of faults, a sender can transmit more octets in one ingress window than expected before the end of the window is reached

- Octet count synchronized to packet reception can limit the exact number of octets in a window
- Windows sizes/expected number of octets can differ per window at one egress port → Each ingress window requires an associated octet limit

Fault Free Case



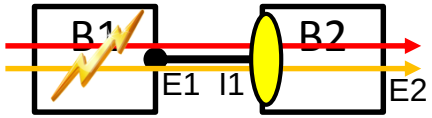
Variances (PTP, 802.1Qbv, …)



Scheduling:
Egress windows aligned to the end of corresponding ingress windows (or later) prevents increasing window size (tolerance) along path

Assumption
- ingress and egress clocks in one bridge are equal

Faults covered by Ingress Window

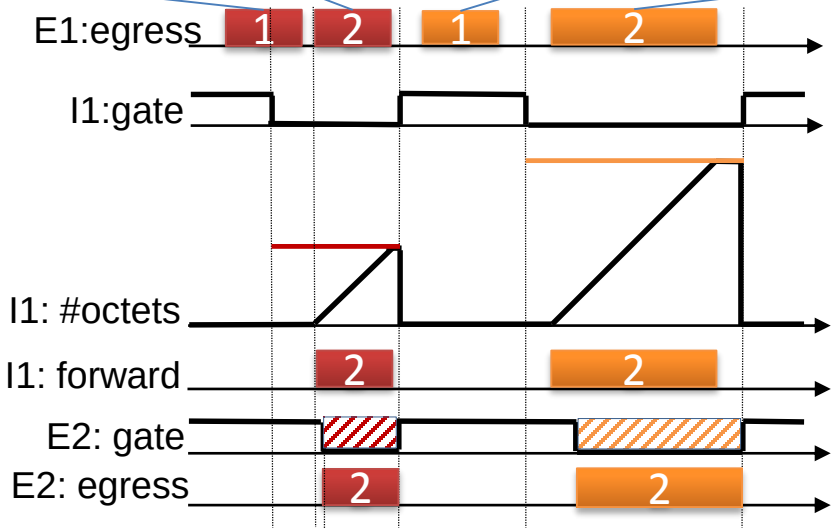


Starts before ingress window → Entirely discarded

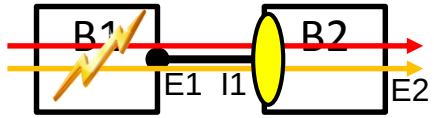
Starts in ingress window → Ok

Starts out of ingress window → Entirely discarded

Expected → ok



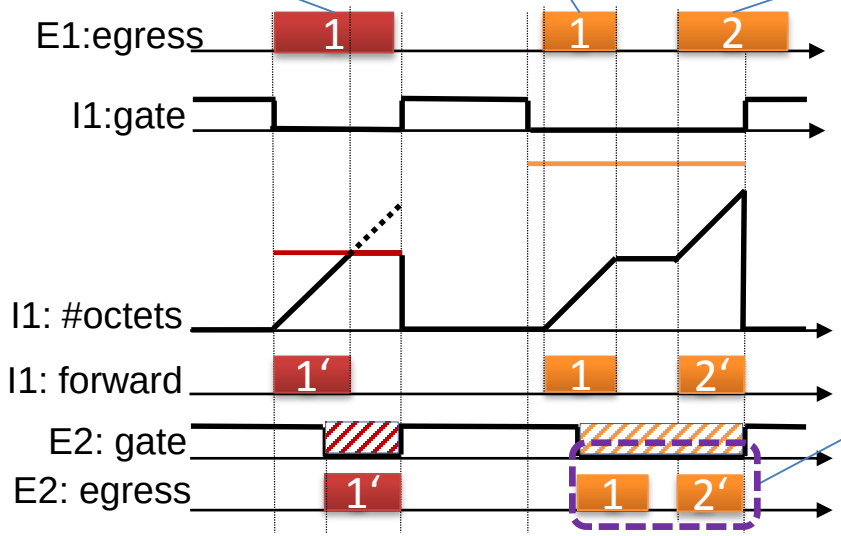
Faults covered by Octet Discarding



Exceeds octet limit → Octets discarded

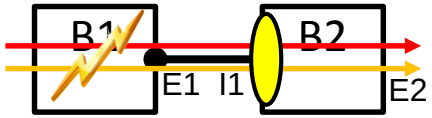
Starts in ingress window and below octet limit → Ok

Starts in ingress window but exceeds the end of the window → Octets discarded



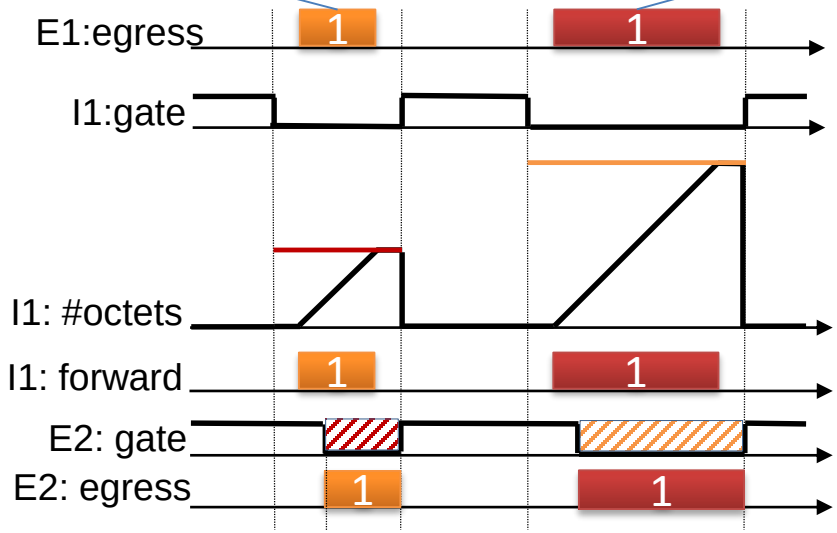
Assumed to be ok:
- Orange stream is faulty anyway.
- Stays within planned limits , i.e. cannot congest other streams.

(Yet) Uncovered Faults …

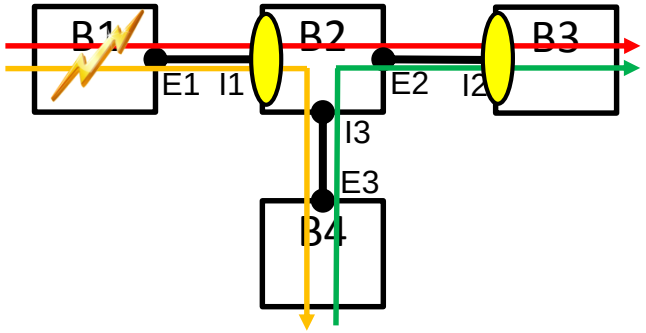


1 sent by B1 in an egress window of a red packet (does not exceed octet limit).

2 sent by B1 in an egress window of a orange packet (does not exceed octet limit).



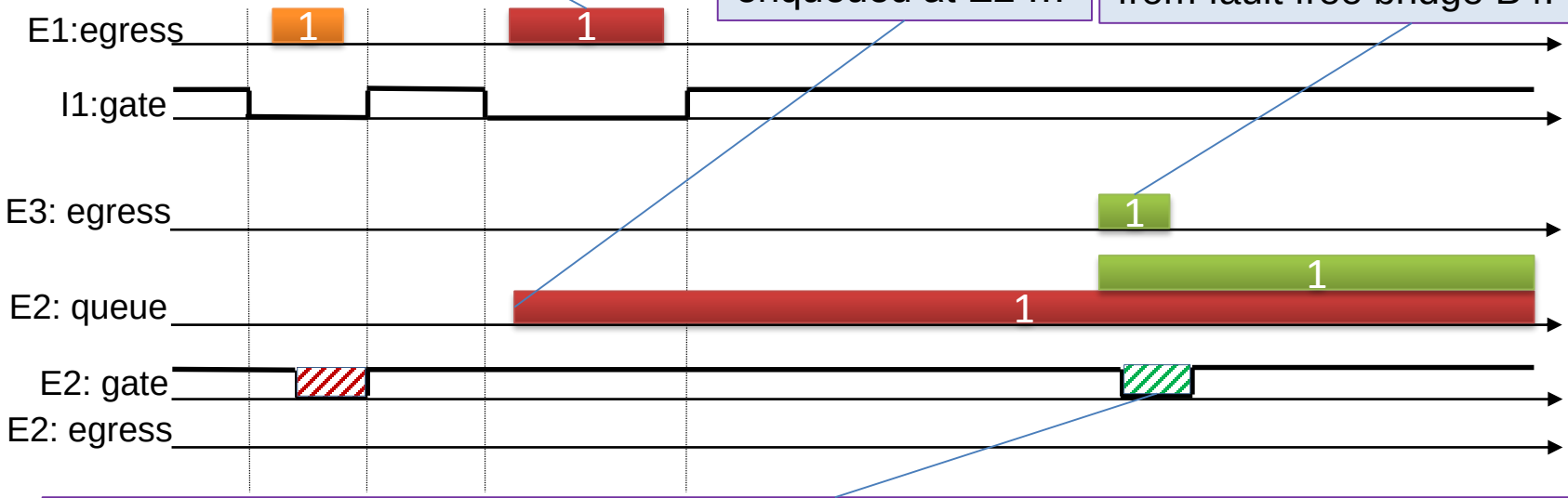
… why this is a problem



Oversized packet 1 arrives in (sufficient large) ingress window of 1

Packet 1 enqueued at E2 …

… while another packet 1 arrives from fault free bridge B4.



1 exceeds the window size planned for packet 1
→ packet 1 is not transmitted, queue at E2 blocked forever

Octet limits/counter not shown to simplify the illustration.

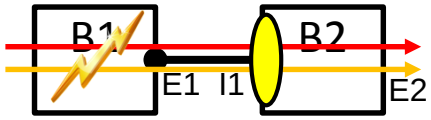
Part 2 – Masquerading

1. Ingress Windows
2. Octet Limits
3. **Masquerading Filters**

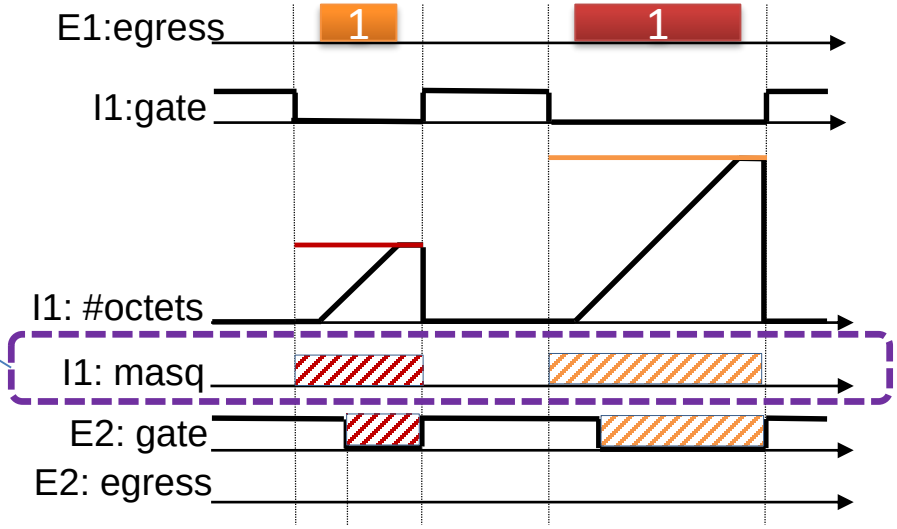
Associate forwarding information with each ingress window to:

- Unambiguously identify:
  a. The entire scheduled path to the listener(s)
  b. All scheduled egress queues on the path to the listener(s)
- Discard packets starting in ingress window in case of mismatch

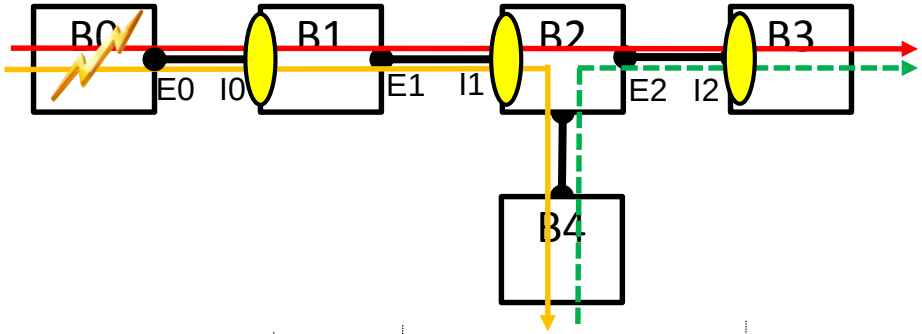
Faults covered by Masquerading Filters



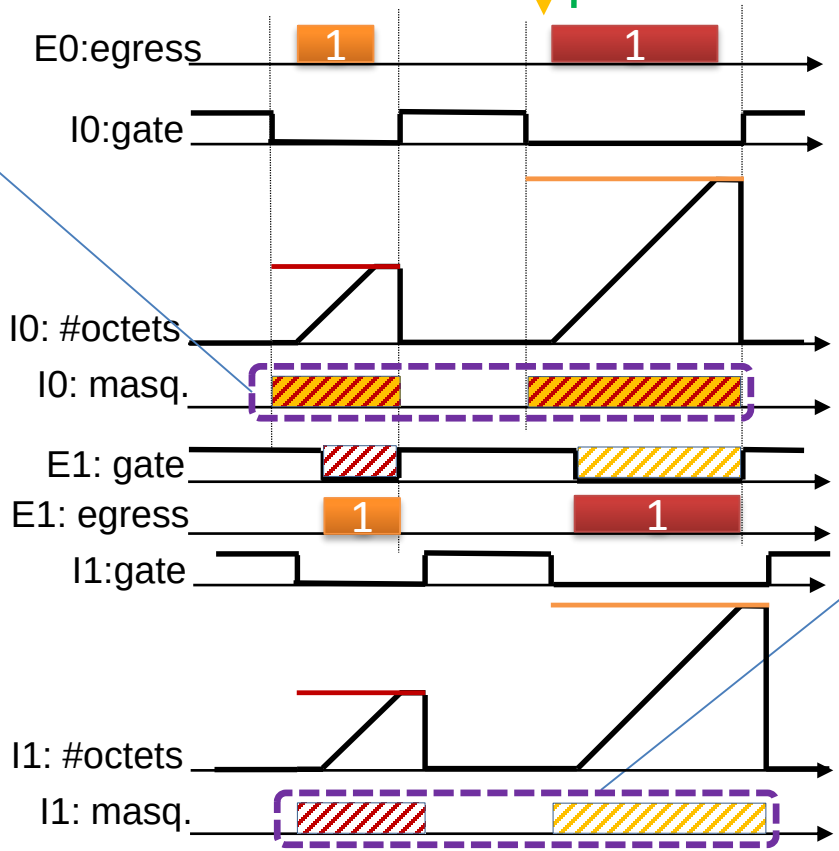
Detects that packet an orange packet arrives in the window of a red packet and vice versa.



Why local forwarding information (port map, etc.) would be insufficient



Masquerading filter in B1 based on e.g. port map – does not detect the wrong packets



Masquerading filterin B2 will detect the wrong packets BUT cannot identify that B0 was faulty, i.e. B2 may classify B1 as faulty (**false positive**)

Mapping the mechanisms to standard(s)

Octet Limits: Is MEF 10.3 the right tool?

- Specified to operate octet-accurate?
- Writable token/octet levels at ingress open/close events?
- Tokens added at rate=0 (i.e. not automatically added over time)?
- Red&green-only operation?
- Continuous operation for cut-through (or is the combination TAS+cut-through+policing useless at all – at least Automotive use seems unlikely)?

Input Windows/Gate Events: 802.1Qbv?

Masquerading Filters – Circuits & Stream Gates?

Thank you for your Attention!

Questions, Opinions, Ideas?

Johannes Specht

Dipl.-Inform. (FH)

Dependability of Computing Systems
Institute for Computer Science and
Business Information Systems (ICB)
Faculty of Economics and
Business Administration
University of Duisburg-Essen

Schuetzenbahn 70
Room SH 502
45127 Essen
GERMANY
T +49 (0)201 183-3914
F +49 (0)201 183-4573

Johannes.Specht@uni-due.de
http://dc.uni-due.de