Scheduled queues, UBS, CQF, and Input Gates

Norman Finn
Cisco Systems

V04 January 14, 2015

Issues addressed in this contribution

1. Building a robust network has always required "Access Lists," meaning (to this author) the ability to perform special actions on frames that match specified criteria such as VLANs, addresses, EtherType, IP protocol, etc. Given the added importance of robustness in Time-Sensitive Networks, and the need to protect good data flows from malfunctioning end stations, perhaps it is time we defined some very basic input controls.
2. Given that only a few time-gated queues are available, it becomes difficult to control a large number of circuits, especially if some require tightly-controlled jitter.
3. Interestingly, the solution(s) suggested, here, to the above issues also happen to offer one way to describe and/or implement Cyclical Queuing and Forwarding (CQF).

Acknowledgements

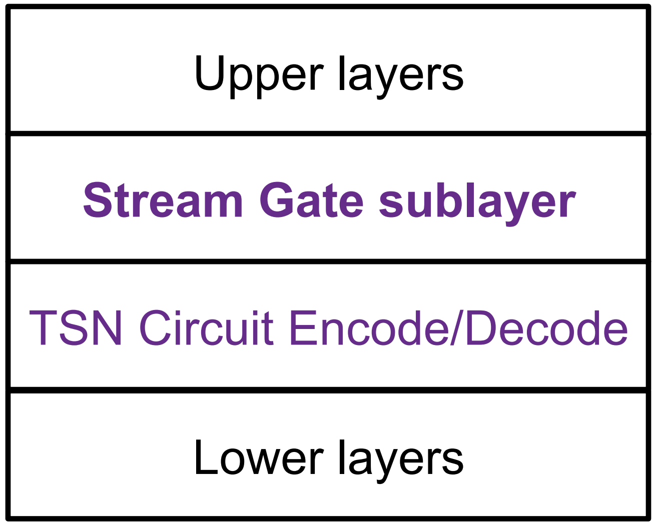
- This presentation borrows ideas from presentations by, and verbal communications with,

  Marcel Kiessling (kiessling.marcel@siemens.com),
  Feng Chen (chen.feng@siemens.com),
  Yong Kim (ybkim@broadcom.com),
  Soheil Samii (soheil.samii@gm.com), and
  Helge Zinner (Helge.Zinner@de.bosch.com).

- This author’s acknowledgement and expression of thanks to those gentlemen does not imply that they do, or do not, agree with anything in this presentation.

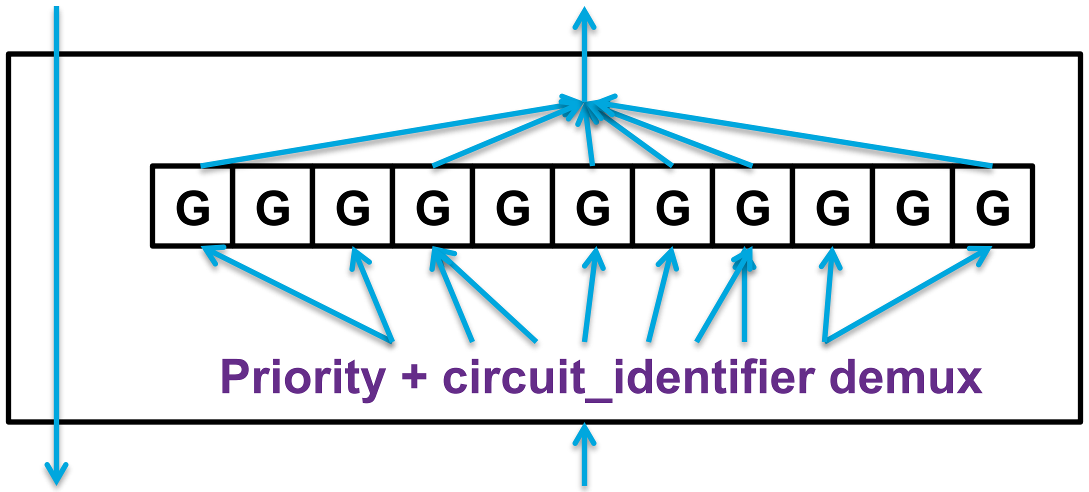
Input gates

Stream Gate sublayer in stack



- Stream Gate sublayer requires a circuit_identifier, which it gets from TSN Circuit Encode/Decode function (see P802.1CB D0.5 Annex C).

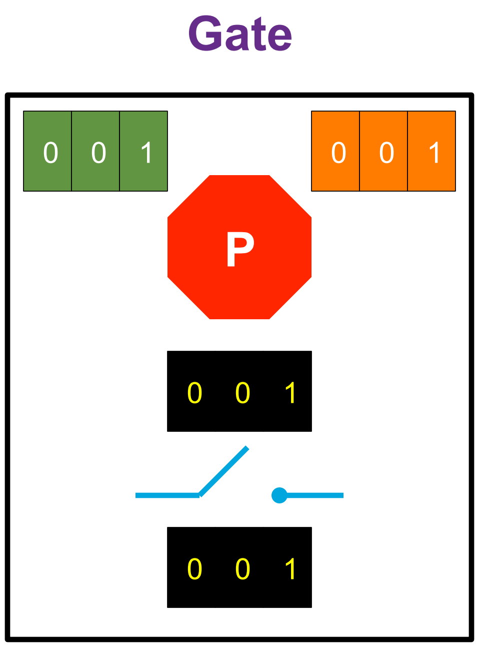
Stream Gate sublayer



Applies to frames coming **up** the stack, **not down**.

- The Stream Gate sublayer has some number of Gates.
- The priority and circuit_identifier select to which Gate a frame is directed.
  - Either the priority, the circuit_identifier, or both, can be wildcarded (any value matches that is not explicitly in the list).
  - Priority + circuit_identifier + wildcard = you can select specific TSN circuits, non-TSN flows, TSN flows not specifically selected, etc.

Stream Gates



- Each Gate has:
  1. A pass / don't pass **switch**. (Optionally, always closed)
  2. An (optional) standard 802.1Q **policing** function (or any other policing function [please suggest others], but the 802.1Q function is easily available)
  3. **Counters** of frames: e.g. passed, marked down, and discarded.
  4. A Service Class output specifier (not a priority specifier).
  5. Filters, e.g. max frame size.
- The pass / don't pass switch can optionally be controlled by a circular schedule, similar to the P802.1Qbv schedule.

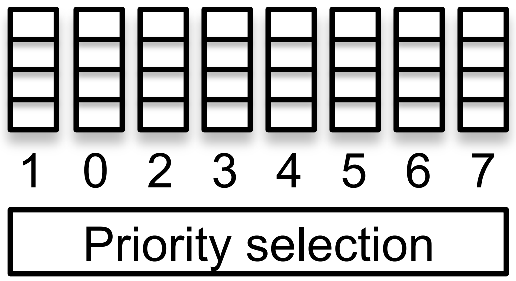
Counters and alarms

- Note that the counters, and programmable alarms associated with certain counter conditions, are important for detecting latent errors ("fool's paradise" issues), where one leg of a redundant path fails, but the data keeps arriving on the remaining path.
- They are also important for tracking down the exact cause of a failure detected downstream from the cause.

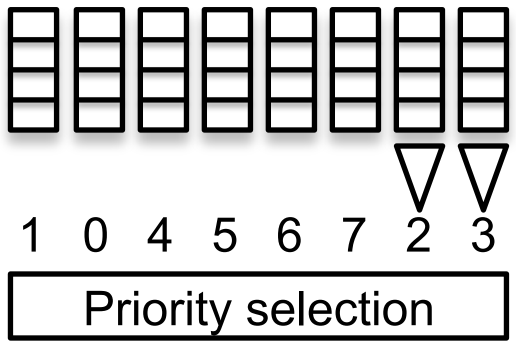
Queue selection by circuit_identifier

Priority queuing and AVB queuing

- 802.1Q: Priority (including weighted round robin)

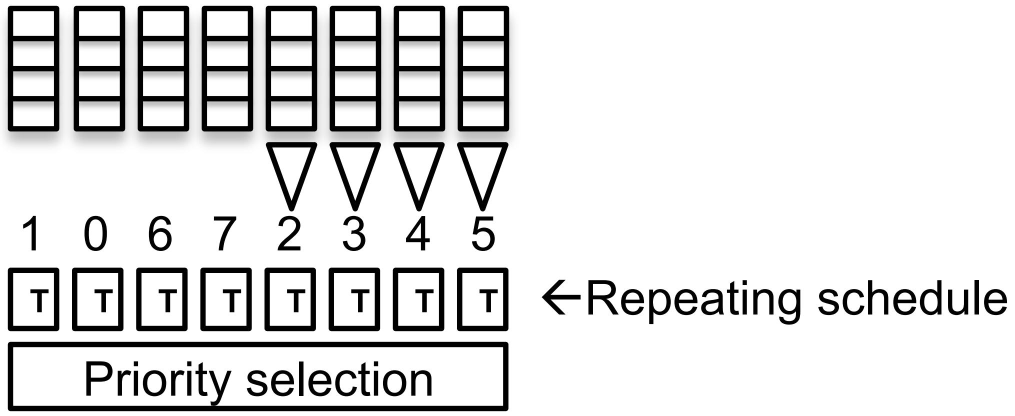


- 802.1Q-2012 (802.1Qat) adds shapers. ∇ Shaped queues have higher priority than unshaped queues, because they can still guarantee bandwidth to the highest unshaped priority.



Time-gated queues

- 802.1Qbv: A circular schedule controls a gate between each queue and the priority selection function.



- (Don't worry about how many queues have shapers. Maybe none. Maybe most. Read on.)

Queue selection by priority

- The queue is selected by a table mapping priority to service class, where “service class” is synonymous with “queue selector”.
- An issue in industrial networks is the number of queues. Offering tightly-controlled jitter requires extra queues, so that you know what particular circuit’s frame you are gating out of the queue.
- But, there may be several circuits that need tight jitter control. These circuits may take various paths through the network that overlap In one way on one port, and another way on another port. (Flows 2+3 on one port, 1+3 on another, 1+2 on a third.) **This is very hard to do using just priorities.**

Example of why it’s hard

- Two stations each transmit a frame on their respective streams A and B during the same time window, both to a single bridge.
- Both go to the same output port on the bridge.
- Because the two streams have different requirements for latency, etc., the bridge wants to schedule them to be output at specific times, first stream A, then later, stream B.
- One way to accomplish this is to place the two frames in two different queues.
- This can be accomplished by giving the two streams different priorities.
- But, this may need to happen at several places in the network. At each bridge, there will be different requirements for splitting flows into queues, so the number of priority values is overrun.

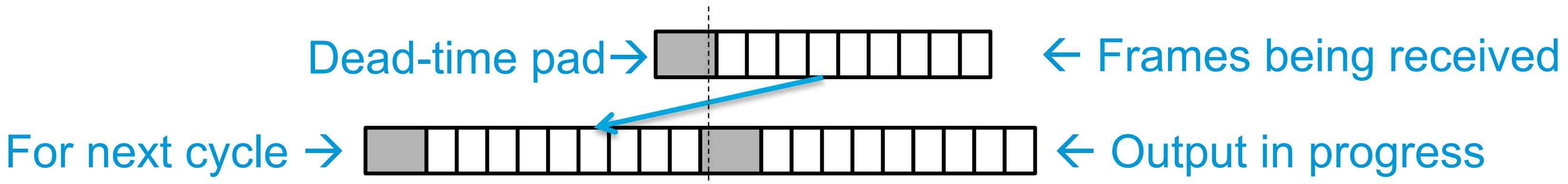
Queue selection by circuit_identifier

- If the circuit_identifier can be used in addition to the priority to determine service class, and thus select a queue, and especially if the arrival time of a frame for a certain circuit can be tightly controlled by input gates, then it becomes much easier to control jitter for a certain number of circuits; you can use the same queue for multiple circuits, spacing them out in time over the cycle, and know that you will not get out of synch.
- Obviously, using the circuit_identifier to select a queue is essential to Johannes Specht's UBS.
- Not quite so obvious is that using {priority, circuit_identifier} to select a shaped queue is **all** you need for UBS.

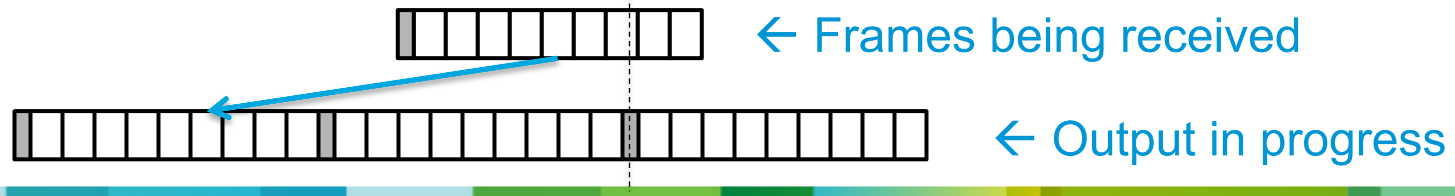
Cyclical Queuing and Forwarding

One model for CQF

- There is an issue with assigning incoming frames to the right output cycle; all frames in the same incoming cycle have to go out in the same outgoing cycle.
- Depending on the details of the length of the cycle vs. the lengths of the wires, this requires either two or three cycle-sized buffers for each output port.
  - ➢ If the wire length and bridge transit time are negligible compared to the cycle time, two cycle-buffers are sufficient.



  - ➢ Otherwise, three cycle-buffers are required to avoid output starvation.



Output queues and CQF

- Let us note that **we can use credit-shaped, time-gated queues for the CQF buffers**.
- You use the time gates to alternate between the 2 (or 3) buffers.
- You use the shaper to space out the transmissions so that they do not have too large an impact on the high-priority but non-TSN traffic (such as BPDUs!), being careful to ensure that the cycle's data will always get out during the window.
- All that is needed to do CQF in addition to what we have is to use time gates to do the queue selection.
- Note that the timing for queue selection is attached to the input port, not the output port; queue selection must be synchronized with the previous hop's output.

Output queues and CQF

- And, we just talked about **input gate timing**!
- Instead of a simple on-off switch, the output of each Input Gate's schedule is a service class (queue selector). Either the class varies with time, or you feed one stream to multiple gates, only one of which is passing frames at any given moment. This allows CQF to be implemented without additional buffers. (Note that we also need a "no class = drop" value.)
- The only change to 802.1Q is to allow the service class to be selected by the input gate, instead of the priority → service class table.
- Note that varying what queue is selected by the frame does not change the frame's priority field, at least in IEEE 802.1Q.

Output queues and CQF

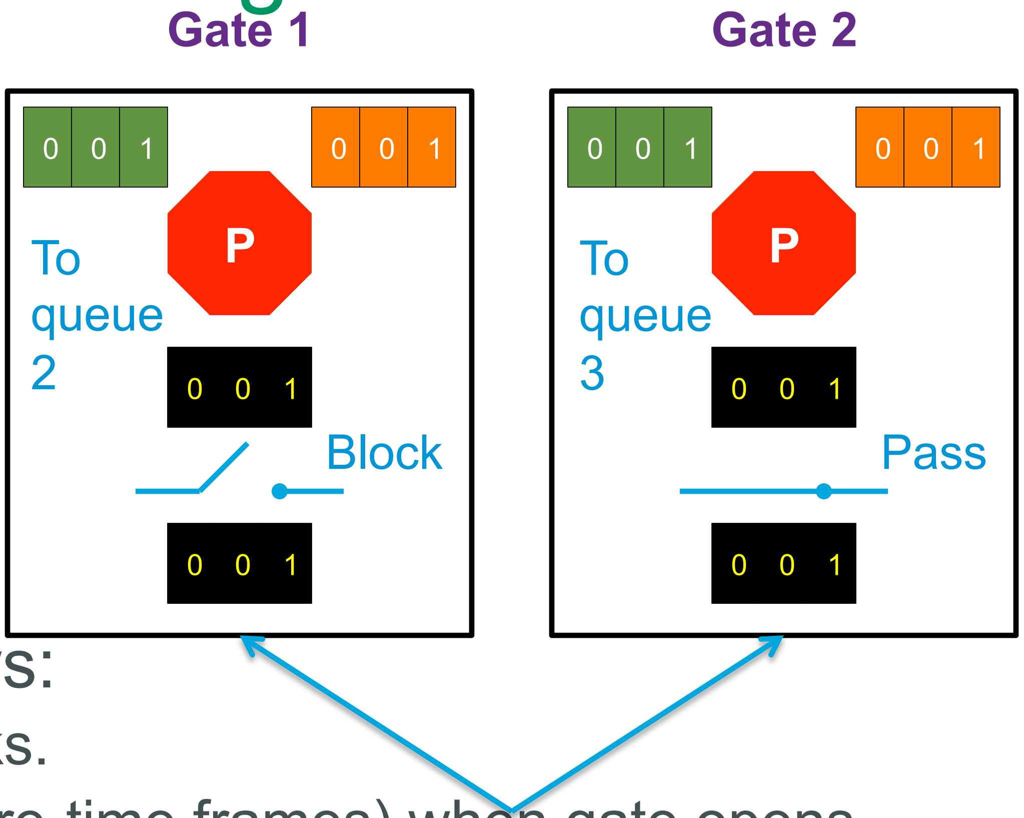
- It is also possible to formulate CQF by assigning multiple priority values to the CQF function, and using priority to mark the cycle to which a frame belongs, but this has severe issues if you have to use three queues, anywhere.

Guaranteeing latency and 0 congestion loss when some flows fail

Robust behavior: CQF

- Suppose some device fails, spewing valid TSN frames at a rate higher than allowed by its reservation (contract).
- If those frames are passed to an output port and mixed with other devices' properly-working streams, all using all of their bandwidth, then some frames will be discarded, because the total bandwidth of the output queue is being exceeded.
- Those frames can be discarded from any stream, not just the offending stream.
- No credit-based policer is accurate enough to prevent this.
- Time-scheduled flow gates, with a trivial policer algorithm, could police CQF queues perfectly accurately.

Two Input Gates feeding two CQF queues



- One flow is sent to both gates.
- Gates alternate, one is off when the other is on.
- Policer works as follows:
  1. Reset when gate blocks.
  2. Allow *n* bits of data (wire-time frames) when gate opens.
  3. More than *n* bits are “red” and cause an alarm.

  NOTE: The “wire-time data bits” count is dependent on the output port(s). That’s the beauty of why 802.1Q doesn’t say exactly where, in your implementation, you place the policer.

Robust behavior: AVB

- I made the claim, earlier, that no credit-based policer is accurate enough to prevent this.
- Consider the case of many flows, all running at their maximum rate.
- One flow injects one extra packet per minute.
- After less than an hour, the queues start to overflow.
- We have not seen a presentation for a policer that will prevent this, or will detect which flow is causing the problem.
- One solution: Use UBS, and when a queue overflows, take action.
- That is: **the policer and the shaper/queue are the same thing**, so AVB policing is **perfectly accurate** with respect to the UBS granularity.
- Then, all we need to do is enable "queue overflow" to trigger mitigation actions.

Alternative formulations

- This presentation suggests:
  - Sending the same frame to multiple Stream Gates; and
  - Having a time-scheduled on/off switch on each one; and
  - Having an output Service Class parameter.
- One could define things differently, for example:
  - Send a frame to only one Stream Gate; and
  - Have the Service Class output (or CQF policing) vary with time.
- This author does not have a strong preference, but thinks that the first method is easier to specify.

Failure mitigation

Failure mitigation

- There are a number of failures that Stream Gates can detect:
  - ➢ Packets rejected by a Stream Gate.
  - ➢ Packets marked red (or yellow) by a policer.
  - ➢ Overflow of a CQF or AVB queue.
- There are a number of mitigation actions that can be taken:
  - ➢ Discard an offending frame.
  - ➢ Close one or more (or all) Stream Gates to block specific streams (including the non-TSN “stream”).
  - ➢ Shut down the port.
  - ➢ Send a notification to the system administrator.
  - ➢ Shut down the system.
- Tying failures to, and automatically executing, mitigation actions can improve the robustness of the network.

Summary

Summary

- The combination of:
  1. Scheduled per-circuit variation of service class selection;
  2. Per-circuit policing and frame counting;
  3. A new CQF policer;
  4. Queue overflow detection; and
  5. Selectable error mitigation actions
- Allows us to:
  1. Implement CQF;
  2. Implement UBS;
  3. Provide a larger number of circuits with low jitter; and
  4. Configure a much more robust network.
- All or part of these ideas can be incorporated either into the CQF PAR or into a new PAR for Per-Circuit Quality of Service.

Thank you.

cisco