Peristaltic Shaper:
updates, multiple speeds

Michael Johas Teener
Broadcom, mikejt@broadcom.com

Agenda

- Objectives review
- History
- Baseline design and assumptions
- **Higher speed links**
  - What do we want?
  - What can we get?
  - How do we mix speeds?
  - Suggestions

Objectives review

- **Deterministic distributed delays for all streams**
  - really, this time I mean it!
  - queues distributed between bridges evenly
- **Scalable delays with link speed**
  - 10x shorter delays ~~for Class A traffic~~ over links with a 10x speed increase
- **Multiple traffic classes**
  - Equivalent to AVB
  - This time we will make sure the “observation interval” is programmable!
- **Use ~~Gen 1 SRP or future SRP/IS-IS~~ 802.1Qcc**

History

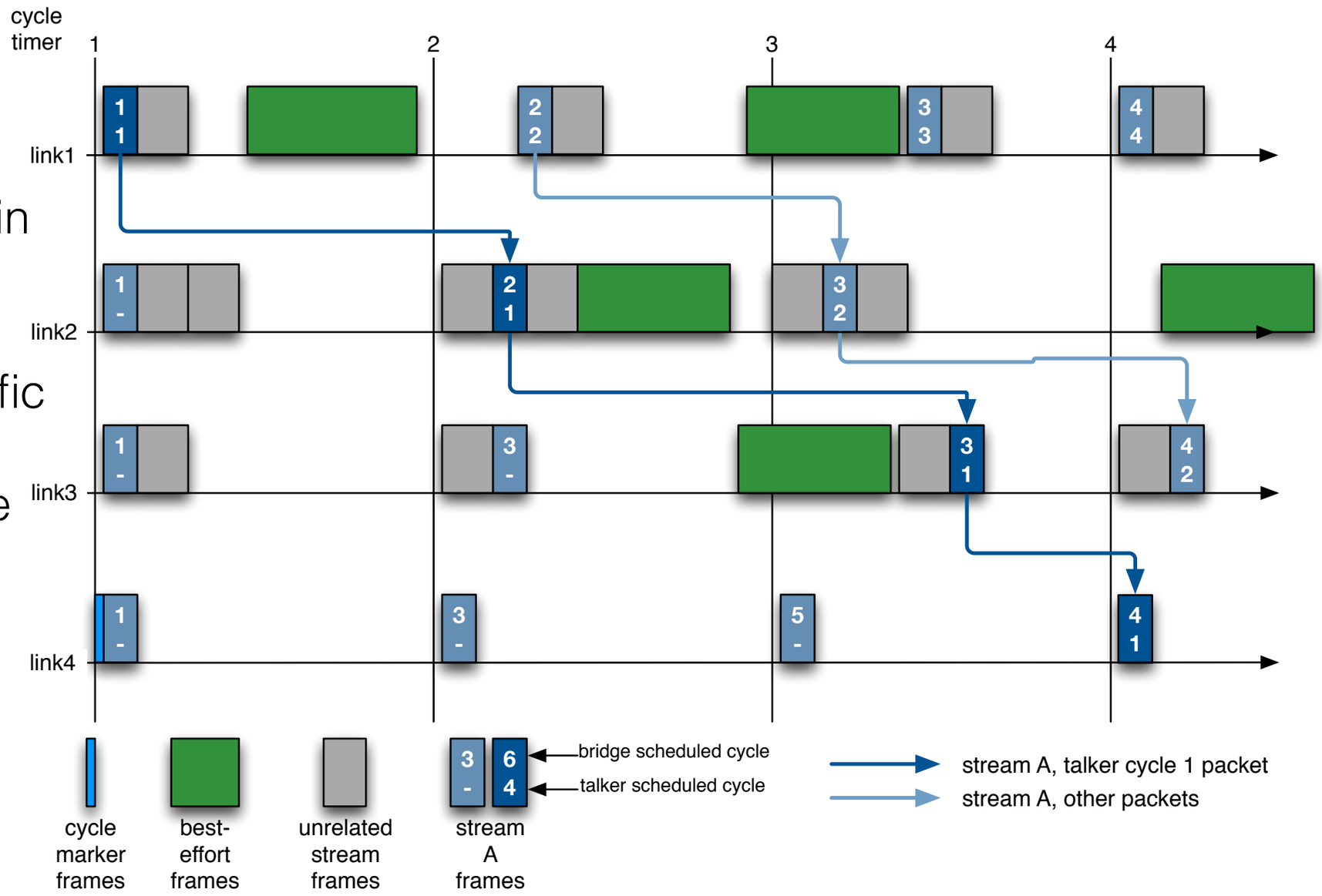
- **See my old presentation for details, but ...**
  - This is an old idea, made feasible by scheduled queues and preemption and ingress policing and class-based QoS and SRP
  - http://www.ieee802.org/1/files/public/docs2012/new-avb-mjt-back-to-the-future-1112-v01.pdf
- **Unknowns:**
  - Interaction of multiple speed links on the path
  - Interoperation with current credit-based shaper

Basic operation

Ensure that the cycle time is greater than the sum of the longest interfering frame plus all the isochronous traffic

now all isochronous traffic will arrive within the same cycle

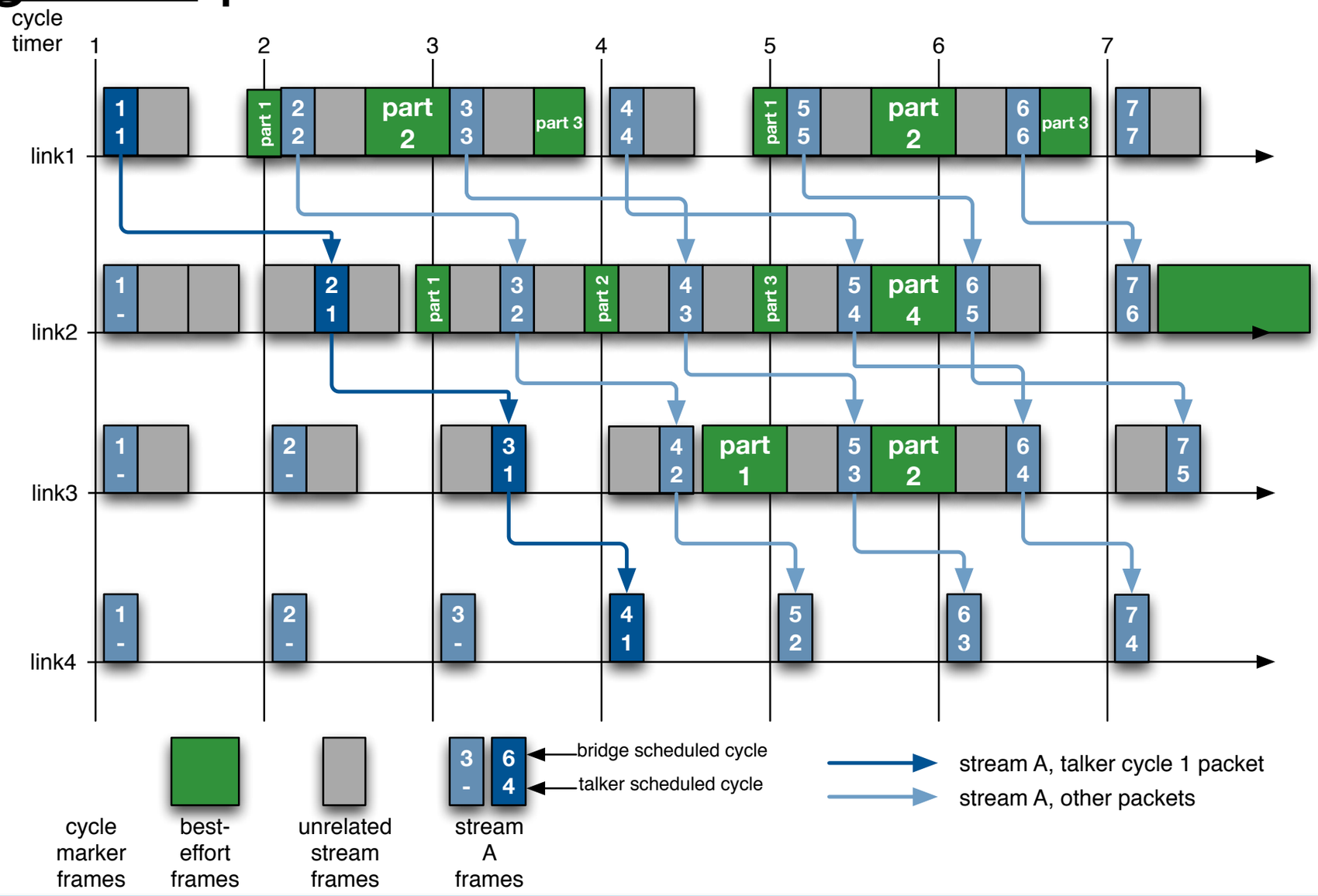
if, of course, this traffic is complete queued at the start of a cycle and is the highest priority



Adding preemption

Now the cycle time must be greater than the longest interfering *fragment* plus all the isochronous traffic

if the max isoch traffic is 75% of the available BW, then the fragment could be almost 400 bytes for 100Mbs links



Assumptions

Within a single "SR Domain"

- **All devices in a path are "time aware systems"**
  - e.g., support 802.1AS/PtP
- **… and are in the same timing domain**
  - e.g., use the same grand master
- **… and share the same "cycle" duration phase**
  - e.g., a cycle starts at the same time for all participating devices

… then …

worst case delay = cycle duration

Multiple speeds?

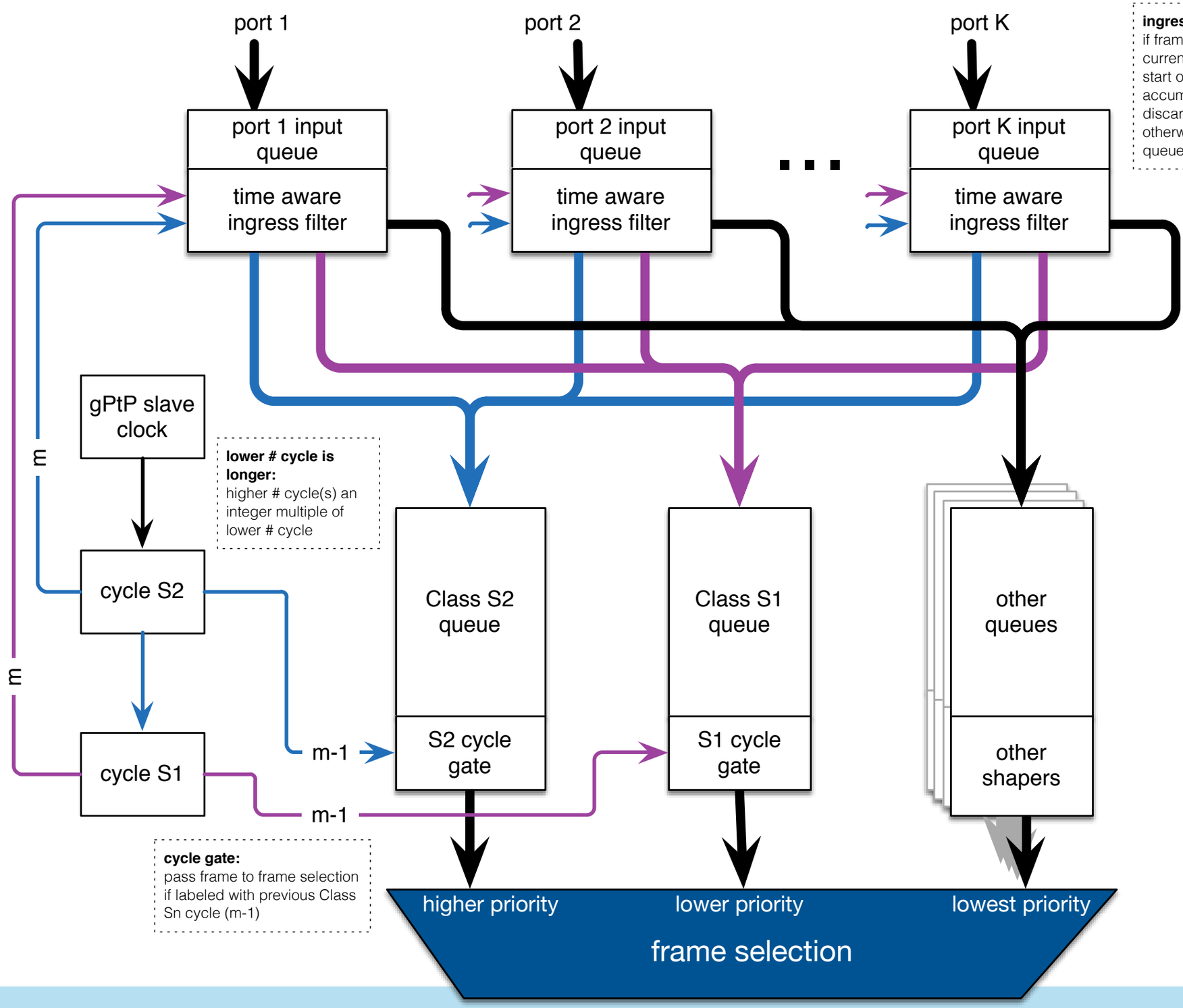
- Wrong question …
- Streams have two parameters of interest: bandwidth **and** worst-case delay
- **Right question: multiple traffic classes?**
  - Where multiple traffic classes correspond to multiple delay classes
  - E.g., Class A is 250us/hop

Multiple delay classes

- **Each traffic class / delay class corresponds to a different cycle duration**
  - Class cycle duration = worst case delay
- **A useful simplification** *(at least for me)* **is to assume all cycle durations are integer multiples of each other …**
  - all set by MIB, but validate by SRP domain
  - Class S1 = 250 µs cycle duration/delay default?
  - Class S2 = 25 µs cycle duration/delay?

It might look like …

Example bridge with two delay classes, S1 and S2



Bandwidth reservation

- **Bandwidth measurement same as Qav**
  - "observation interval" = cycle duration
  - … but …
- **Frame size limits are a bit complex:**
  - depends on lowest bandwidth on path
  - different streams with the same class will have different frame size limits depending on the path
- **Calculation of bandwidth limits per class**
  - possible to set *per link*
  - path reservation process needs to determine limits and report

Buffer implications
(and max frame sizes)

- Class bandwidth limit and cycle duration drive max buffering required per port
  - 75 Mb/s @ 250 µs ≈ 2500 bytes
  - 7.5 Gb/s @ 250 µs ≈ 250,000 bytes !!!
- **Going the other way …**
  - 7.5 Gb/s @ 2.5 µs ≈ 2500 bytes
  - 75 Mb/s @ 2.5 µs ≈ 25 bytes !!!

Good stuff

- Really simple to implement
- Really provides deterministic delays
- Really has fixed upper limit to buffers
- Really limits delivery jitter
- With appropriate defaults, is completely compatible with existing SRP and planned improvements

Bad stuff

- **Path dependent frame size limits possible**
  - Small delays and lower link speeds don't mix
    - (but you knew that already, so is that a problem?)
- **Can't automatically get shorter delays with faster links**
  - Need to use a shorter cycle duration/class, requiring shorter frames
  - Forcing shorter frames is already an issue

Improvements

- **Possible to run a single cycle duration for all classes**
  - Delays for lower class can reduced but the available bandwidth for that class gets reduced
    - remember, max bandwidth = cycle duration * link speed and cycle duration = delay
    - if cycle duration is constant, then bandwidth scales with link speed
    - if cycle duration becomes link dependent (gets shorter with link speed increase), then bandwidth for that cycle drops with link speed increase
  - But that might be OK!

Next steps

- **Is it important that we reduce delays for a particular class depending on per-hop link speed?**
  - If so, validate concepts for link speed dependent cycle duration
  - I will report back in a couple of weeks
- **No matter what, I think the peristaltic shaper is important**
  - Think about a PAR, or can we slip it into Qbv?
- **Need to evaluate interoperation with Qav**
  - Credit based shaper sucks, I'd like to deprecate it