



802.1Qbv Scheduled Traffic: Window Options



Rodney Cummings

National Instruments

Rodney.Cummings@ni.com

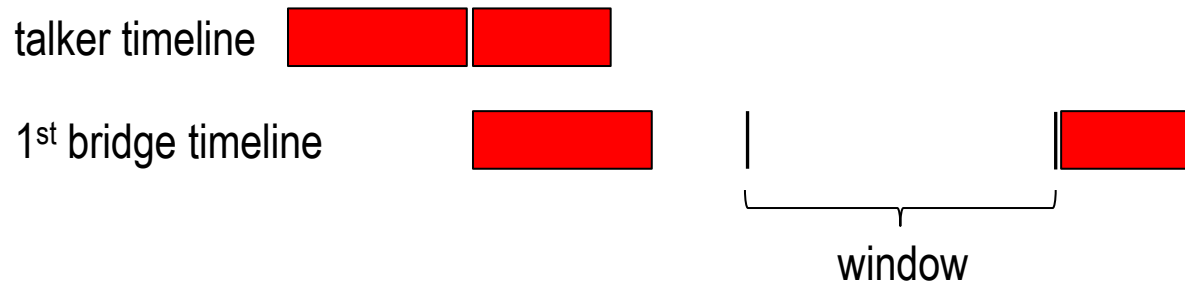


Conventions

- **A**: Class A configured as scheduled traffic
 - Previously known as “ultra-low latency”
- **Non-A**: All traffic that is not scheduled
 - Class B: credit-based shaper
 - Best-effort: strict-priority shapers
- Figures show store & forward

Start of Scheduling Window (1 of 3)

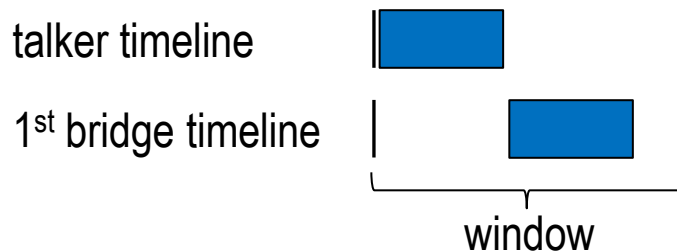
- Guard band prevents non-A from ending after start-of-window (t_0)



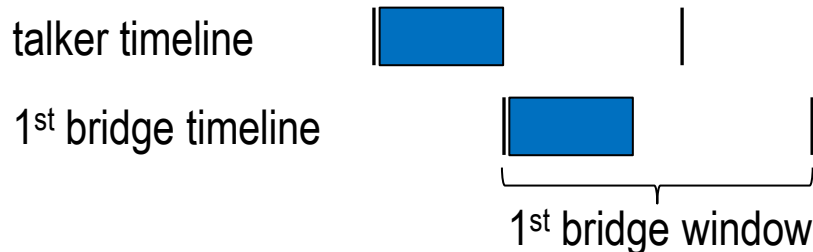
- Assume A prior to start of window will wait in queue
 - Reject instead? Policing... not covered in this presentation

Start of Scheduling Window (2 of 3)

- If talker window starts at same time as 1st bridge, we always have wasted bandwidth

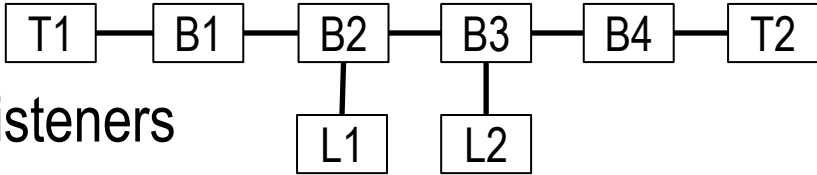


- Proposal: Talker windows start earlier than 1st bridge
 - Offset by length of talker's 1st A frame for that window

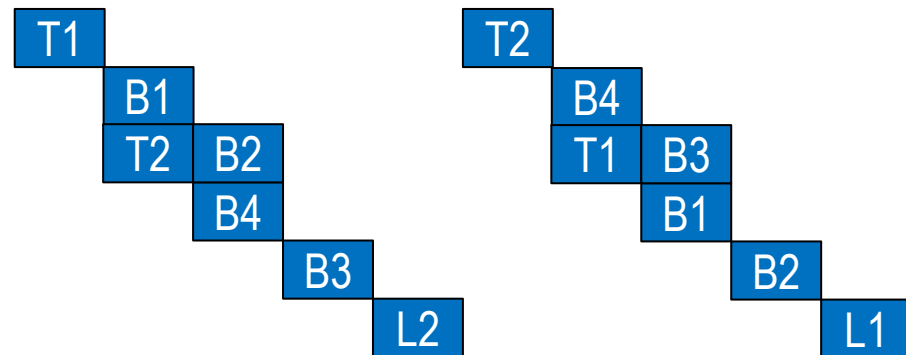


Start of Scheduling Window (3 of 3)

- Offset in a similar manner from 1st bridge to 2nd bridge?
 - Window configuration is distinct per egress shaper (direction)

- Let's look at an example 
 - Both talkers send to both listeners
 - Find offset back from each listener

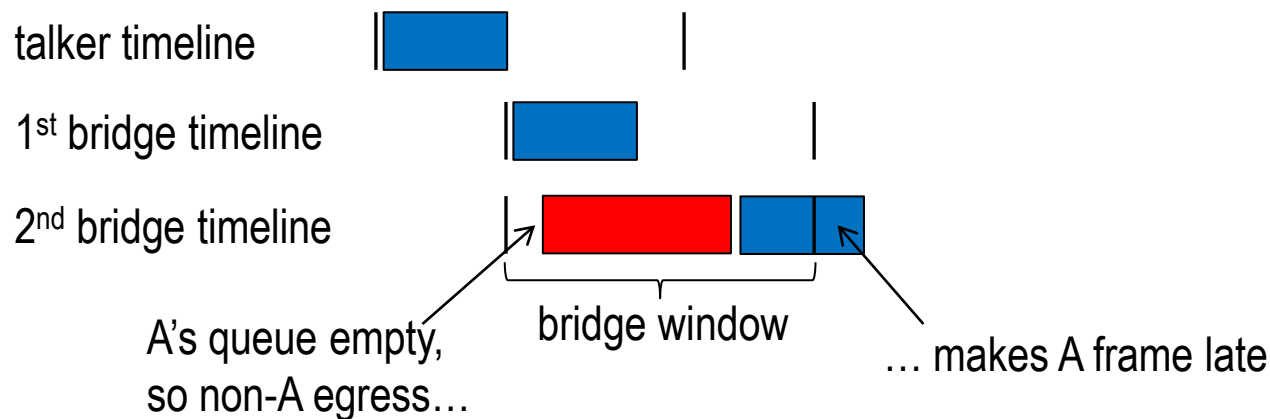
- If merge listener timelines, one bridge egress (e.g. B1 to B2) must repeat



- Conclusion: Offset in bridges requires multiple windows

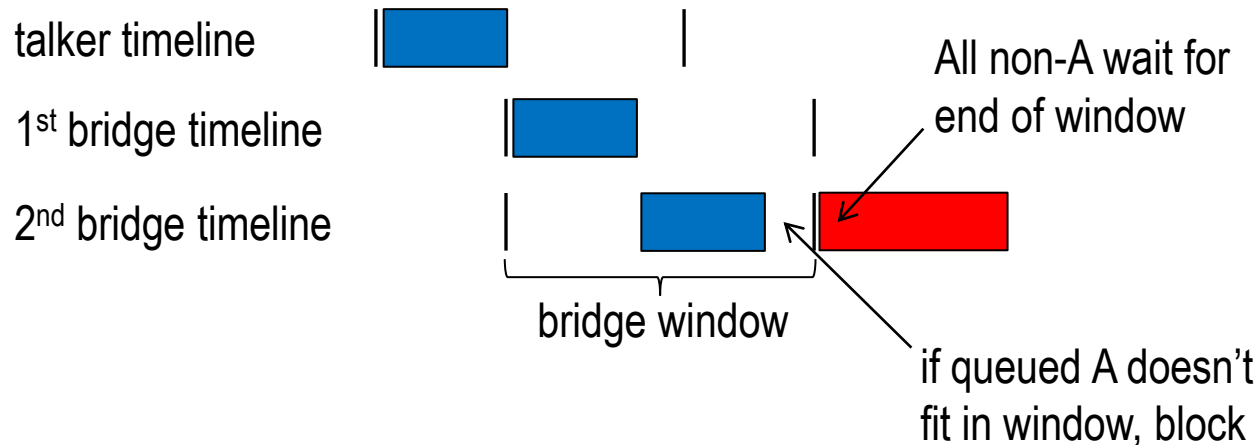
Inside Scheduling Window (1 of 2)

- March 2011 TABS presentation:
burst A frames until A's queue empty, then allow non-A
 - Pro: Non-A uses available bandwidth (like Gen 1)
 - Con: Doesn't work for scheduled traffic



Inside Scheduling Window (2 of 2)

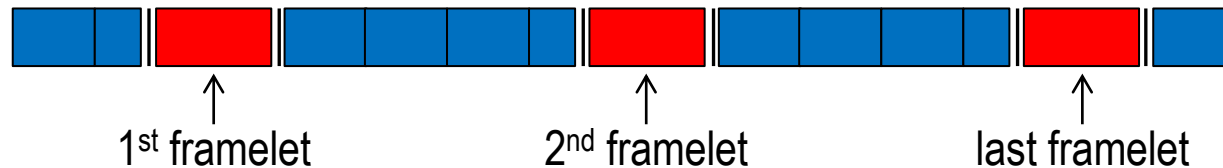
- Proposal: Allow only A inside window, not non-A
 - Non-A is blocked in queues until end of window
 - A uses a guard band for end of window



- Pro: Works for scheduled traffic
- Pro: Simple to calculate bandwidth for A (window)
- Con: Worse non-A bandwidth when window has idle time

End of Scheduling Window

- What happens when gap between windows is less than max frame length (1522 byte)?
 - Without preemption, max non-A is impossible
 - With preemption, max non-A preempted multiple times

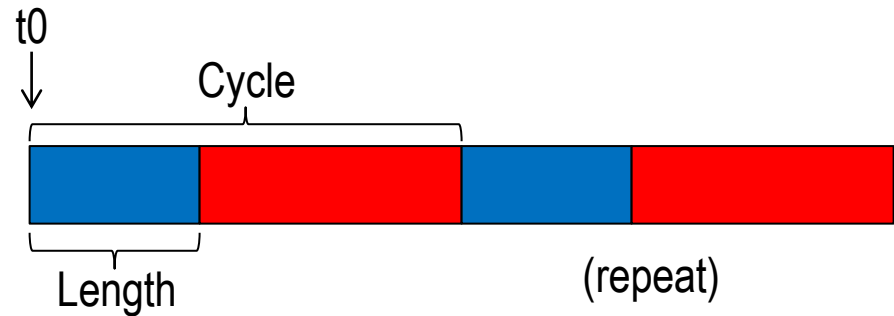
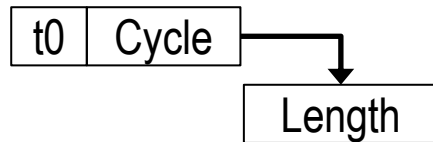


- Con: Adverse effects on non-A latency and bandwidth
 - Con: May complicate preemption design
- Proposal: Gap must be max frame length or more
 - Meets automotive & industrial requirements

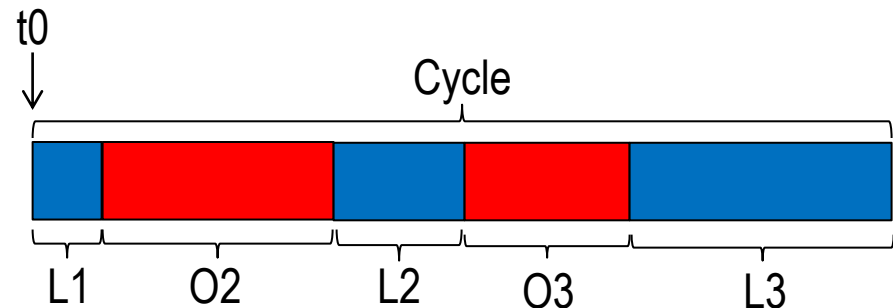
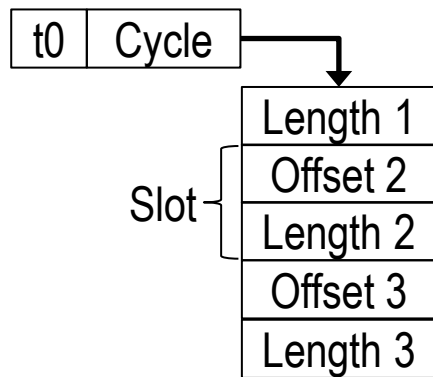
How Many Windows? (1 of 2)

- Focus on windows first
 - Mapping of streams to windows... second

- One window



- Multi window



How Many Windows? (2 of 2)

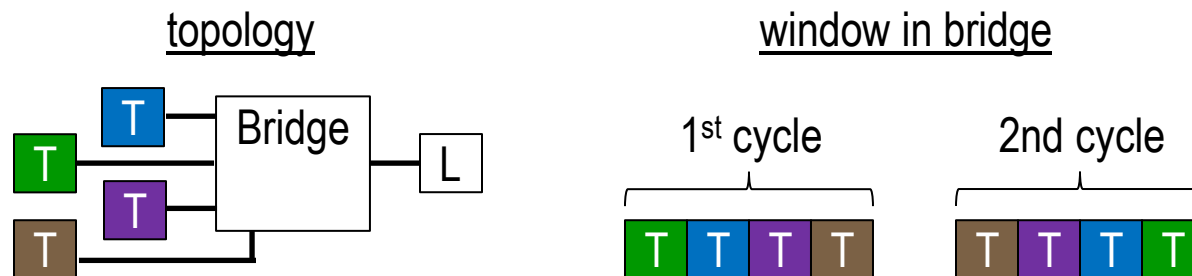
- Using the one and multi assumptions, I'll cover three options
 1. One-in-talker, one-in-bridge
 2. Multi-in-talker, one-in-bridge
 3. Multi-in-talker, multi-in-bridge

One-in-talker, one-in-bridge (1 of 2)

- Also known as “one slot all stream”
- All bridge windows start same time (shared t_0)
 - Previous conclusion:
Offset of bridge windows requires multi-in-bridge
- Talker windows offset earlier than bridge window
- Talkers deblock all queued A frames at start of window
 - Slot within window is considered multi-in-talker

One-in-talker, one-in-bridge (2 of 2)

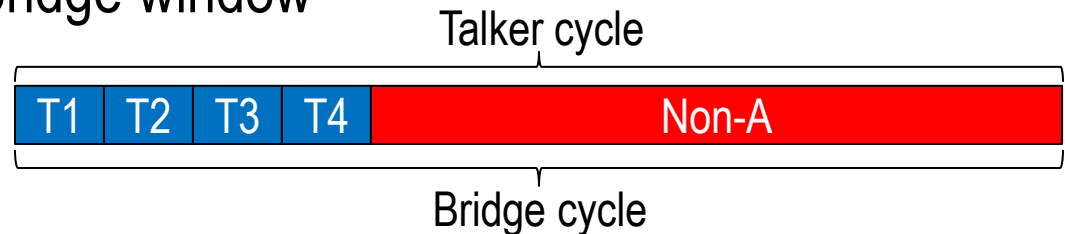
- Pro: Simple to implement
- Pro: Simple to configure
- Con: Latency and jitter close to window length



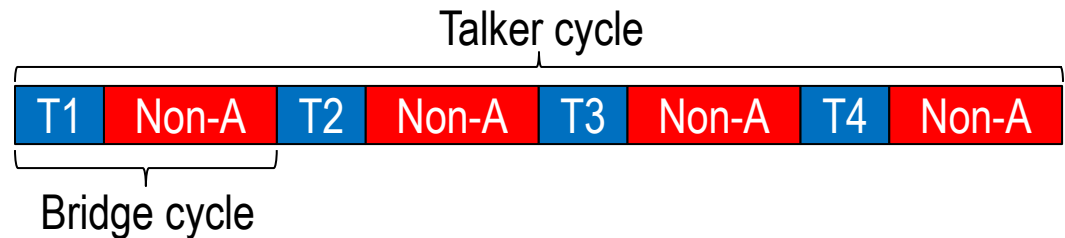
- Class A interference doesn't meet industrial/automotive reqs
 - <http://www.ieee802.org/1/files/public/docs2011/new-avb-boiger-meeting-gen2-latency-req-1111.pdf>
- Con: Large idle time in window for large hop counts
 - Decreases non-A bandwidth

Multi-in-talker, one-in-bridge (1 of 3)

- All bridge windows start at same time
- Two sub-options for talker windows (slots)
 - Slots subdivide bridge window



- Slots span multiple bridge windows

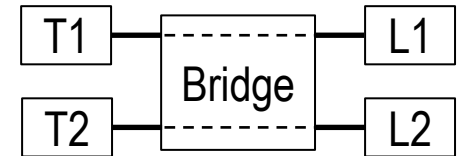


- Sub-options not mutually exclusive

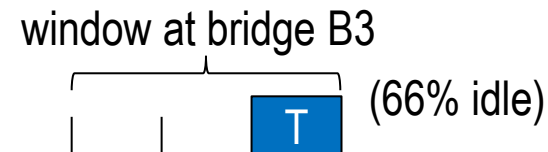
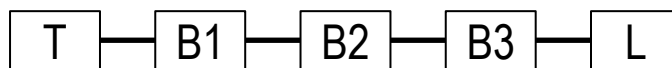


Multi-in-talker, one-in-bridge (2 of 3)

- Pro: Meets automotive/industrial requirements
 - Must engineer to ensure one stream per slot per egress
 - Avoid class A interference
 - Streams can share a slot as long as different egress
 - Example: stream $T1 \rightarrow L1$, stream $T2 \rightarrow L2$,
both streams can share a slot



- Pro: Supports ordering in talkers
- Con: End-station more complex than one-in-talker
- Con: Idle in bridge window; reduced non-A bandwidth

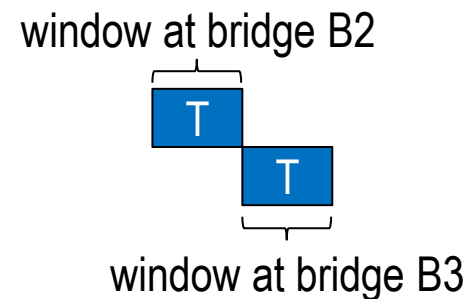
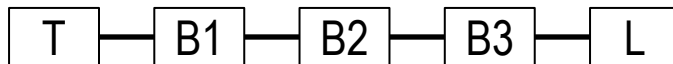


Multi-in-talker, one-in-bridge (3 of 3)

- For this option, slots must be specified in 802.1
 - Otherwise 802.1 doesn't meet requirements
 - Important part of talker's scheduled shaper
 - FlexRay Host Interface specifies that “message transmission operates on non-queued transmit buffers”, where each buffer schedules a slot in the window
 - Slot-level scheduling in FlexRay end-station chip

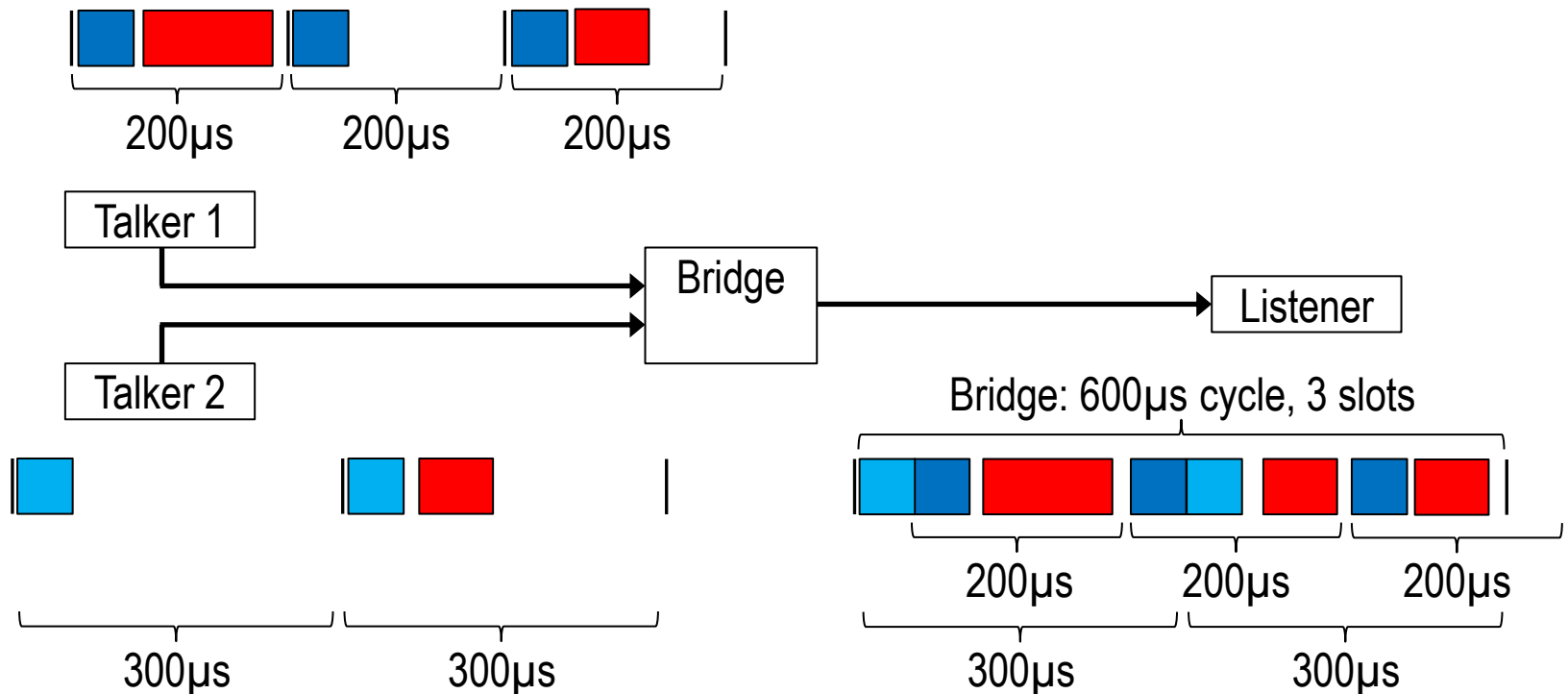
Multi-in-talker, multi-in-bridge (1 of 3)

- List of windows in talker and bridge
- List can be different in each talker and each bridge
- Not necessarily window-per-stream
 - Streams in different directions can share a window
- Pro: Removes idle; optimizes bandwidth usage



Multi-in-talker, multi-in-bridge (2 of 3)

- Pro: Non-harmonic stream rates



Multi-in-talker, multi-in-bridge (3 of 3)

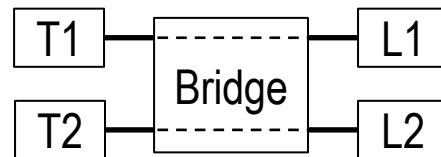
- Pro: Most flexible configuration
 - Can revert to one/one or multi/one
 - Add new windows with less impact to previous windows
 - No longer forced to share a window in bridges
- Pro: Consistent in talker and bridge
- Con: Talker and bridge more complex

How Many Windows?

- Proposal: Multi-in-talker, multi-in-bridge
 - Most pros with reasonable silicon complexity
 - One/one doesn't meet requirements
 - Multi/one wastes bandwidth
- Follow-up question:
Minimum number of windows required (i.e. PICS)?
 - 4 is useful
 - Non-harmonic example uses 3
 - 128 is closer to enabling stream-per-window
 - Common number of messages in CAN & FlexRay MACs

Forwarding/Filtering by Port (1 of 3)

- Proposal: Specify 'Domain' for scheduled shaper
 - Specify end-stations and bridges using scheduled shaper
 - Agree on window/slot configuration within domain
 - Scheduled traffic filtered outside domain
 - Similar concept to AVB Gen 1 domain
- Do scheduled talkers broadcast or multicast in domain?
 - For this example with streams $T1 \rightarrow L1$ and $T2 \rightarrow L2$



does L2 receive T1 (broadcast), or not (multicast)?

Forwarding/Filtering by Port (2 of 3)

- Broadcast has benefits
 - Typical for automotive / industrial / big-physics control
 - E.g. CAN and FlexRay
 - Simple to configure and failover
 - No Stream concept required
 - Simple implementation: VLAN filtering for Domain
- Multicast has benefits
 - More flexible than broadcast
 - Reduces filtering in listeners
 - E.g. CAN / FlexRay MACs provide filtering to mitigate broadcast
 - Consistent with Gen 1: Destination MAC filtering

Forwarding/Filtering by Port (3 of 3)

- Proposal: Both... multicast and broadcast
 - Meets requirements for variety of applications
 - Specify 'Stream' concept for multicast
 - Registration of talker & listeners
 - If needed to complete stream specification, broadcast could register stream as talker only
 - Stream doesn't need a Tspec
 - Implicitly specified by stream's window/slot
- Proposal: Allow multiple domains, which can overlap
 - Facilitates use of broadcast
 - Overlap implies sharing slots across multiple domains

Thank you