Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Institute of Computer Science
Chair of Communication Networks
Prof. Dr. Tobias Hoßfeld

Latency Model and Example Reservation Flow in RAP

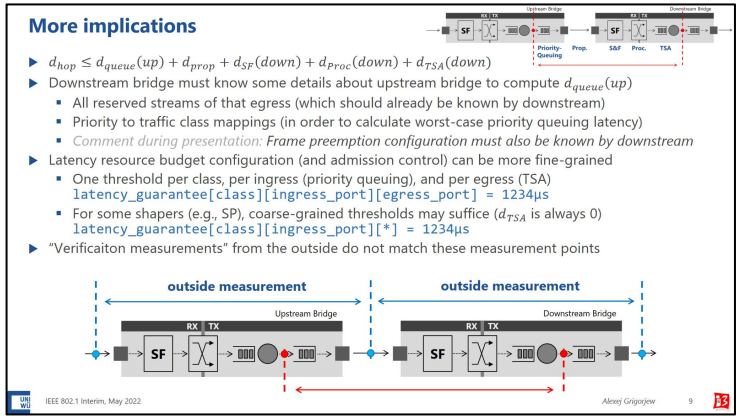
IEEE 802.1 Meeting, January 2023

Alexej Grigorjew
University of Wuerzburg
alexej.grigorjew@uni-wuerzburg.de

Feng Chen
Siemens AG
chen.feng@siemens.com

Overview for this Presentation

- **Recap: last presentation (measurement points for latency models)**
  - cf. dd-grigorjew-measurement-points-0522-v02.pdf
  - What are the measurement points (and resulting delay segments)?
  - What are the reasons for this change?
  - Some implications



- **More implications**
  - Visualization of delay segments
  - What happens at the Listener?
    - Suggestion: merge two configurations per delay segment
  - What happens with different Shapers?
    - Suggestion: communicate the behavior at the Priority Transmission Selection Queue

- **Example reservation flow**
  - Very simple scenario (2 switches, 2 streams)
  - Clarify general procedure
  - Clarify the implications of delay segments

if we have time

Recap: Last Presentation

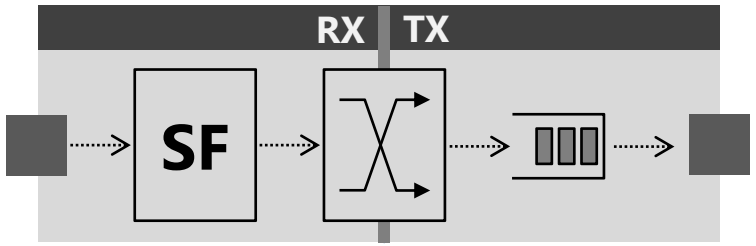
MEASUREMENT POINTS FOR LATENCY MODELS

Extended delay model, including transmission selection algorithm

previous presentation
dd-grigorjew-...-0522-v02.pdf

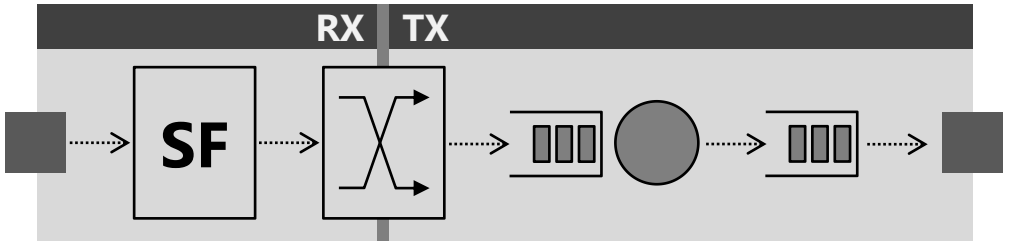
- Split “queuing” latency of formal latency models into…
  - Transmission Selection Algorithm (TSA)
  - Priority-Queuing, where only the eligible frames interfere

Previous model:



Queuing
(including TSA)

Extended model:

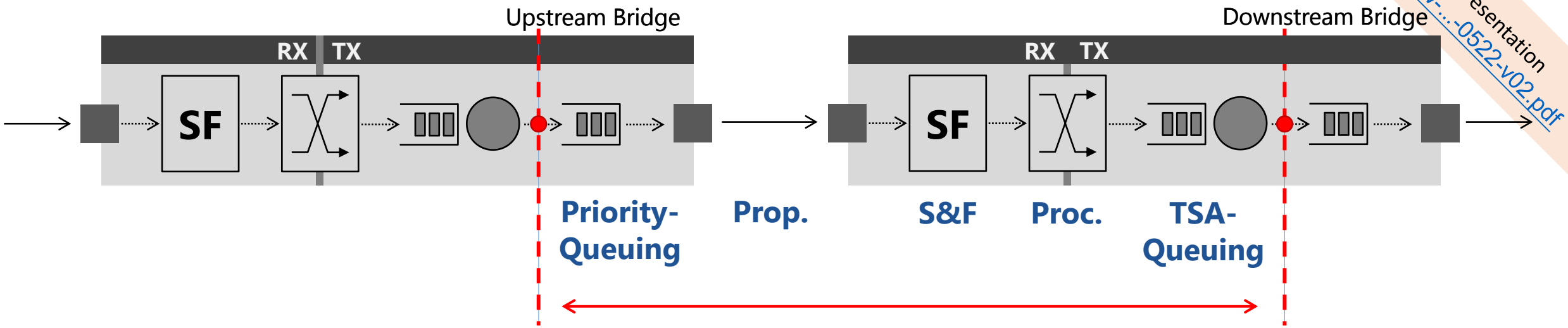


TSA-Queuing **Priority-Queuing**

- Add measurement point during queuing when frame **becomes eligible for transmission**
  - SP: Immediately after enqueuing
  - CBSA: When credits >= 0, the head of the queue becomes eligible for transmission
  - ATS: When the defined eligibility time for that frame is reached (cf. Qcr)
  - CQF: When queues swap roles (receive → send), all frames in the send queue become eligible

Suggestion: Use ATS measurement points for all shapers in RAP

previous presentation dd-grigorjew-...-0522-v02.pdf

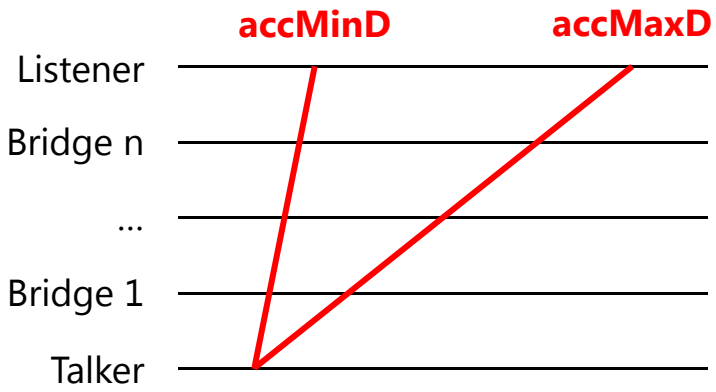


- Suggestion: Use the ATS measurement points for all TSAs & latency models in RAP
- Per-hop latency is given by…
  - Queuing after eligibility time was reached (upstream) // queuing for priority transmission selection
  - Propagation
  - Store-and-Forward (downstream)
  - Processing (downstream)
  - Queuing until eligibility time is reached (downstream) // queuing for transmission selection algorithm
- *Comment during presentation: PHY can often introduce a delay after priority queuing. The simple suggestion is to account for it as part of the upstream processing delay, even if it technically occurs after the measurement point.*

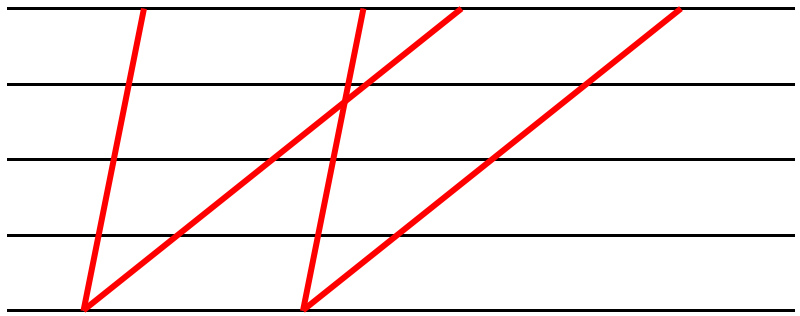
Why is shaper-to-shaper latency beneficial?

previous presentation dd-grigorjew-...-0522-v02.pdf

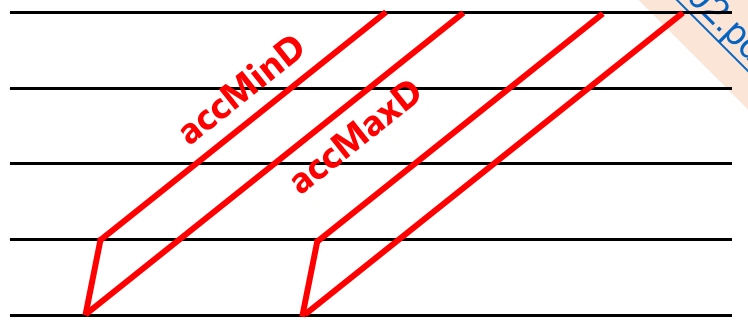
Distributed latency model:



CQF (edge to edge measurement):



CQF (shaper to shaper):

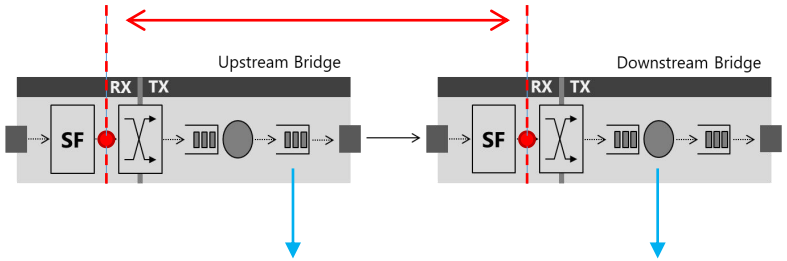


dd-grigorjew-strict-priority-latency-0320-v02.pdf

Generally:

- Minimum delay and maximum delay accumulated per hop
- Accumulating bursts are calculated based on *(accMaxD - accMinD)*
- A lower latency variance is better for downstream delay computation

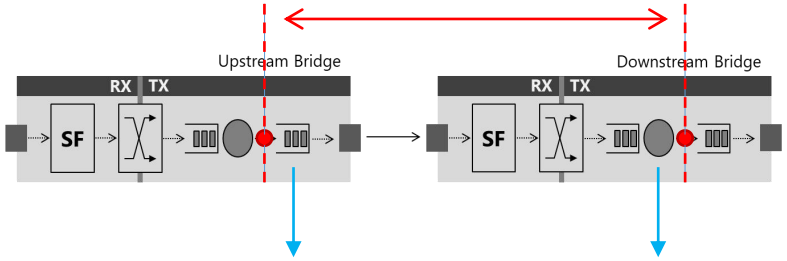
Fully-received to fully-received:



Upstream priority queue is a major **source of jitter**

CQF removes this jitter – but only **after** the measurement point

Shaper to shaper:

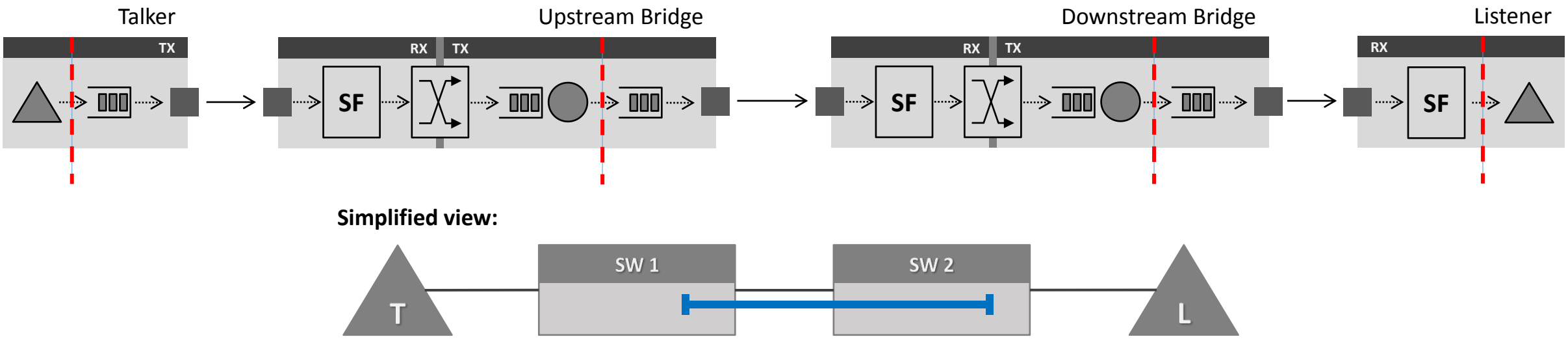


All sources of jitter can be removed; **Well-defined traffic pattern, as intended by the TSA, is measured directly after the TSA**.

Visualization and new Suggestions

MORE IMPLICATIONS

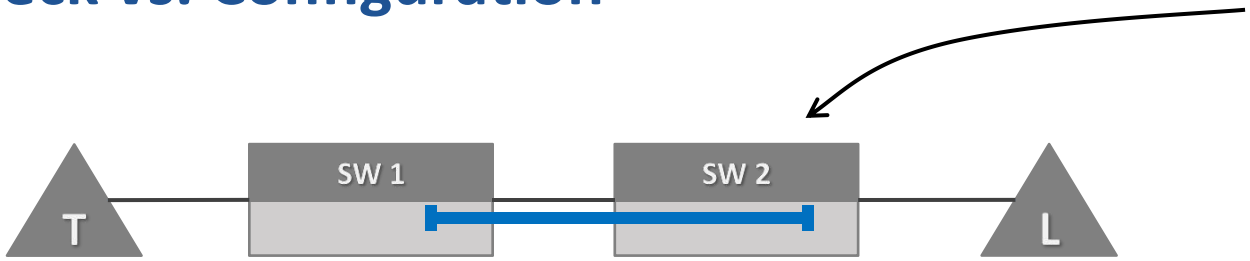
Full End-to-end Path with Delay Segments



- One delay segment includes
  - TX of upstream bridge (SW 1)
  - RX of downstream bridge (SW 2)
  - TX of downstream bridge (SW 2)
- Downstream bridge (SW 2) performs the bounds check during reservation
- But where does the configuration (delay threshold) come from? SW 1 or SW 2?
- General problem: on any path with **n bridges** (2 bridges), we have **n+1 delay segments** (3 delay segments)

Bounds Check vs. Configuration

Ingress Port	Egress Port	Traffic Class	Delay Threshold
1	2	7	150 µs
1	2	6	500 µs
…			



- Initial suggestion: SW 2 performs bounds check **and** contains the delay threshold config
  - But: we don‘t really like the fact that SW 1 has no say, although it is involved in the delay segment

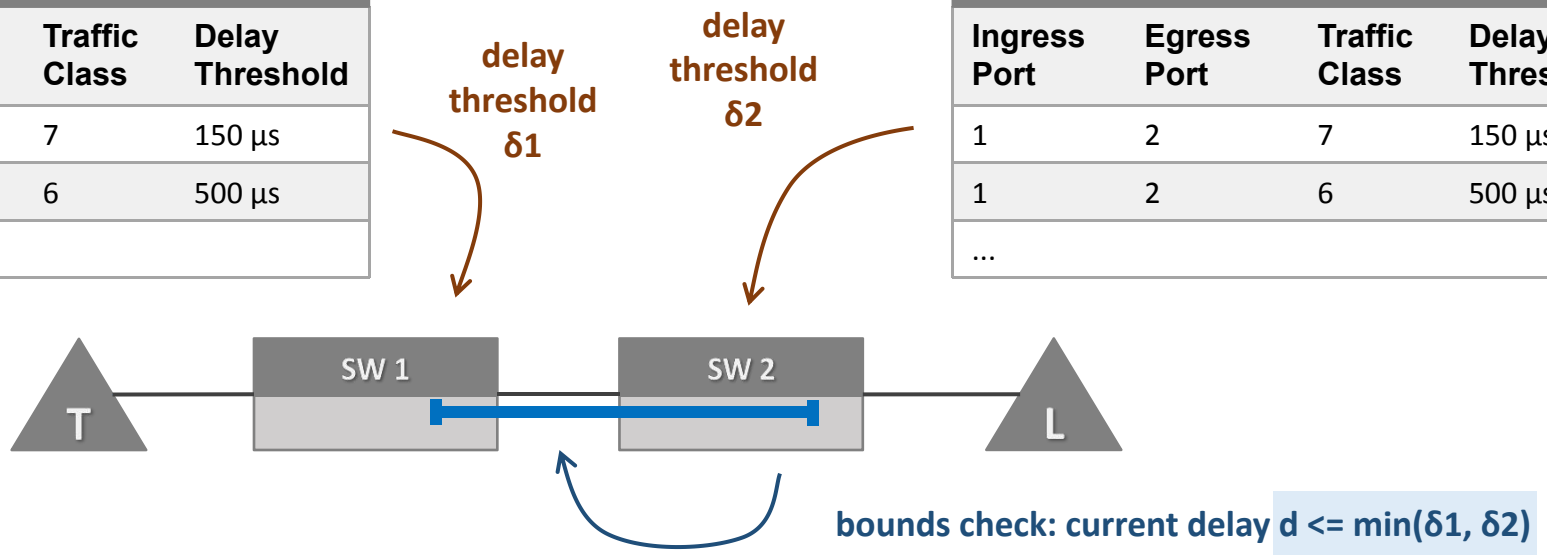
- In addition: what happens at the Listener?

  - It can perform bounds checks
  - But we don‘t really want to **configure** that aspect in our end devices
  - (Config sources can be: default configuration, profile, CLI, Network Management System)

Suggestion: Both Devices Suggest a Delay Threshold

Outgoing Delay Segments		
Egress Port	**Traffic Class**	**Delay Threshold**
2	7	150 µs
2	6	500 µs
…		

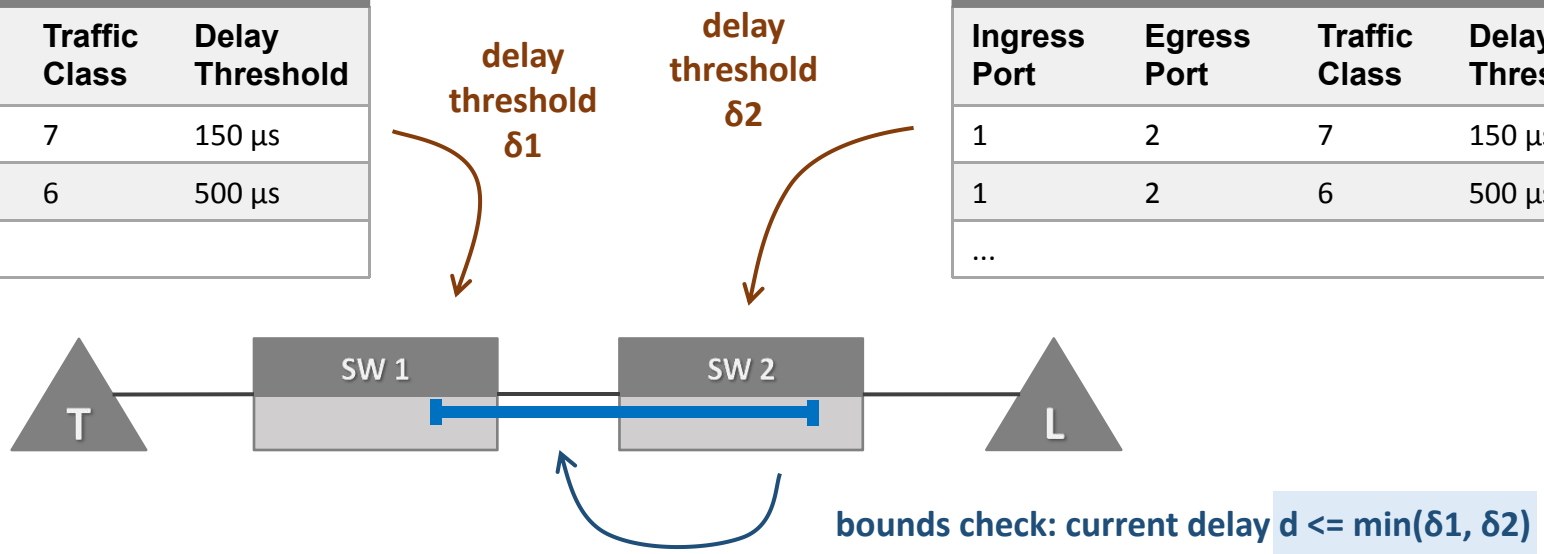
Incoming Delay Segments			
Ingress Port	**Egress Port**	**Traffic Class**	**Delay Threshold**
1	2	7	150 µs
1	2	6	500 µs
…			



- Suggestion: split threshold configuration for each delay segment into two configs
  - Upstream bridge (SW 1) has one config **for each egress port** and traffic class
  - Downstream bridge (SW 2) has one config **for each in ingress/egress port pair** and traffic class
- Each bridge will have two delay config tables: one for outgoing delay segments, one for incoming segments
- Upstream (SW 1) communicates the outgoing δ1 with the downstream neighbor (SW 2)
  - Downstream aggregates both configurations and selects **the minimum of both** for bounds checking

Implications of Having two Delay Threshold Tables

Outgoing Delay Segments		
Egress Port	**Traffic Class**	**Delay Threshold**
2	7	150 µs
2	6	500 µs
…		



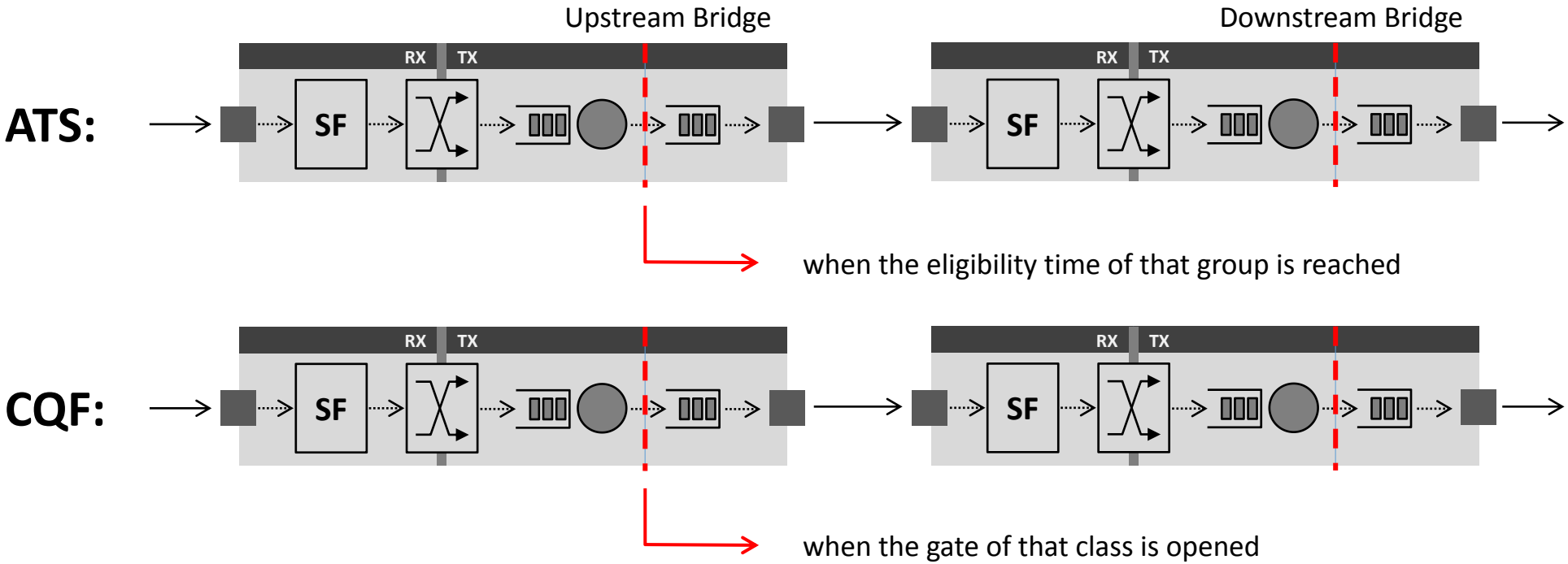
Incoming Delay Segments			
Ingress Port	**Egress Port**	**Traffic Class**	**Delay Threshold**
1	2	7	150 µs
1	2	6	500 µs
…			

- The listener no longer needs a delay threshold configuration
  - It can simply use δ1 of upstream (SW 2 in that case)
  - It **can** still specify its own δ2 where necessary (e.g., **routers** are listeners from layer 2 RAP point of view)
- When optimizing a network‘s configuration (e.g., via NMS), simply use the same value for δ1 and δ2
  - It is the same delay segment after all
- Upstream (SW 1) could specify “*don‘t care*“ in order to prevent unnecessary resource constraints
  - Technically, it still **needs** a valid outgoing δ1 config in case an end device connects to that port

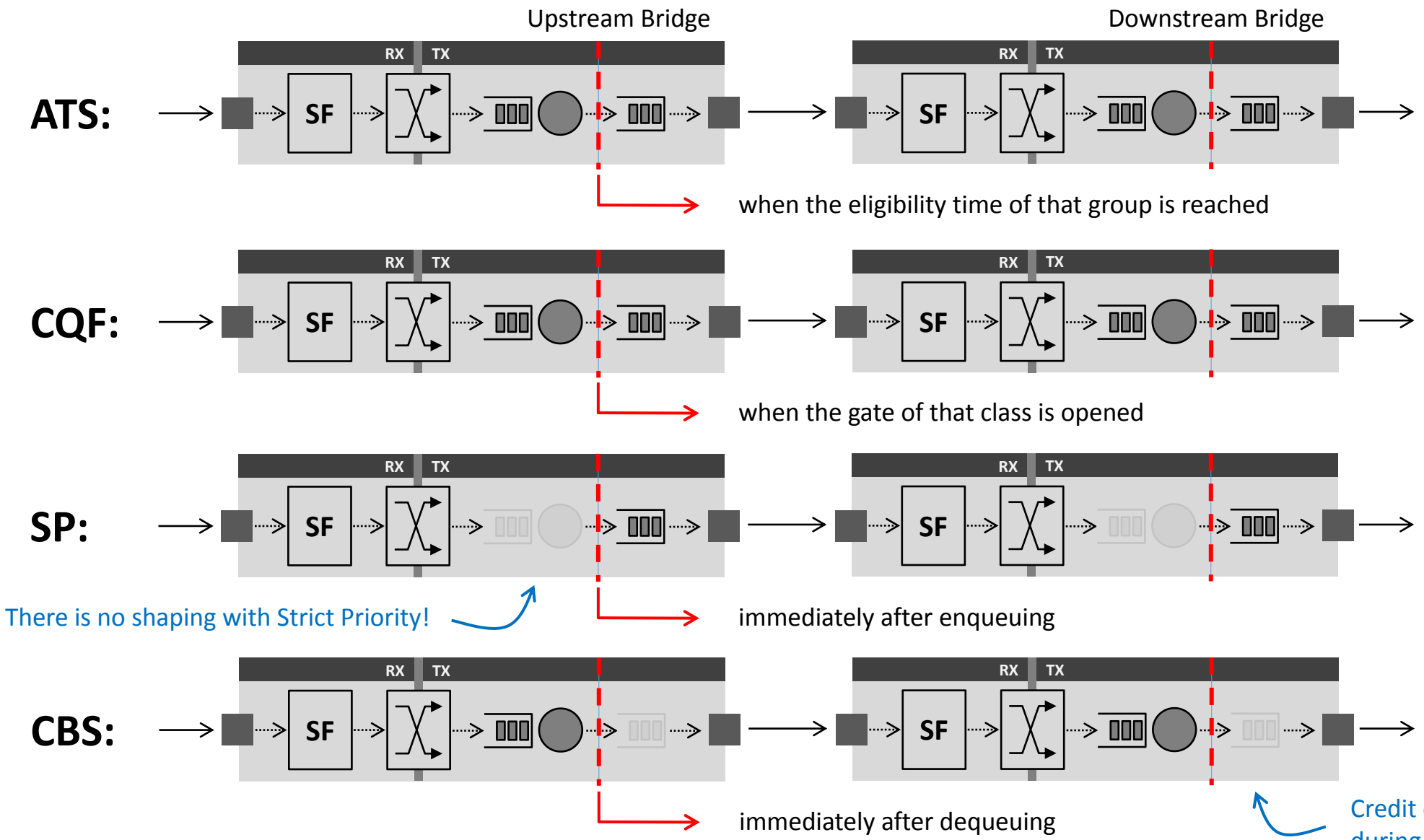
A Closer Look at Delay Segments with Different Shapers

- Recap old presentation: delay segments begin when the frame "becomes eligible for transmission"
- **More specifically**, we want delay segments to be tied to the events that change the **shaper's state**
- This ensures that the shaper has the intended effect on the latency model
- This is simple for ATS and CQF:

CQF (shaper to shaper):

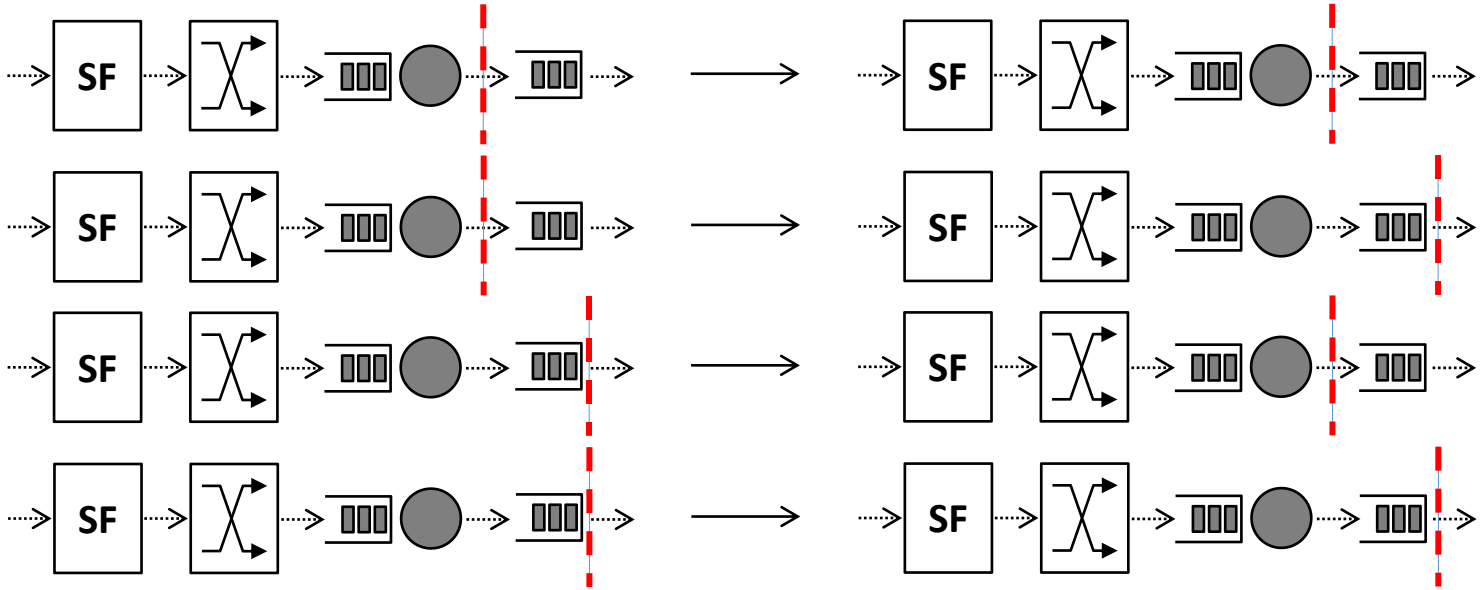


Not All Shapers Use All Delay Segments



Suggestion: Communicate Whether the Last Queue is Part of the Delay Segment

- Suggestion: Instead of fixed, shaper-specific behavior, introduce a variable that indicates whether the (pure) **priority transmission selection is part of the next delay segment**
- For heterogeneous networks, this creates four scenarios:



- Each device can now specify which delay segment the last transmission queue belongs to
  - ATS and CQF do not include it
  - CBS does always include it
  - SP can now decide! (this can help with the vast zoo of end devices)

This includes the bounds check and the accMaxLatency field

TAs, LAs, Bounds Checks, Example Values

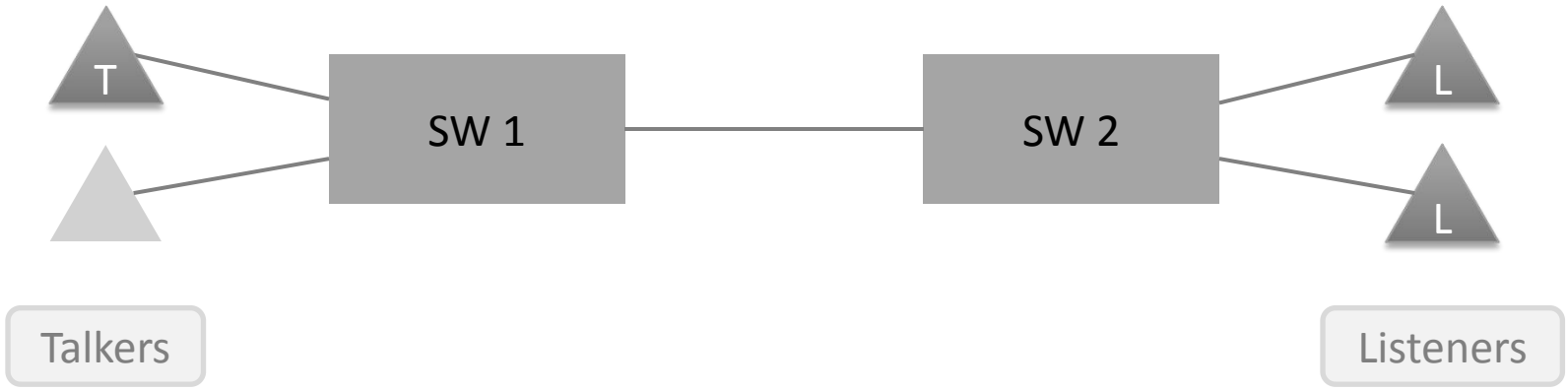
EXAMPLE RESERVATION PROCESS

Disclaimer

- Just a simple example!
- Many things are simplified
- Some things are only suggestions
- Some things are subject to change in the standard

- See this as a means for easy introduction
- Please do not cling to the details

Example Topology Overview



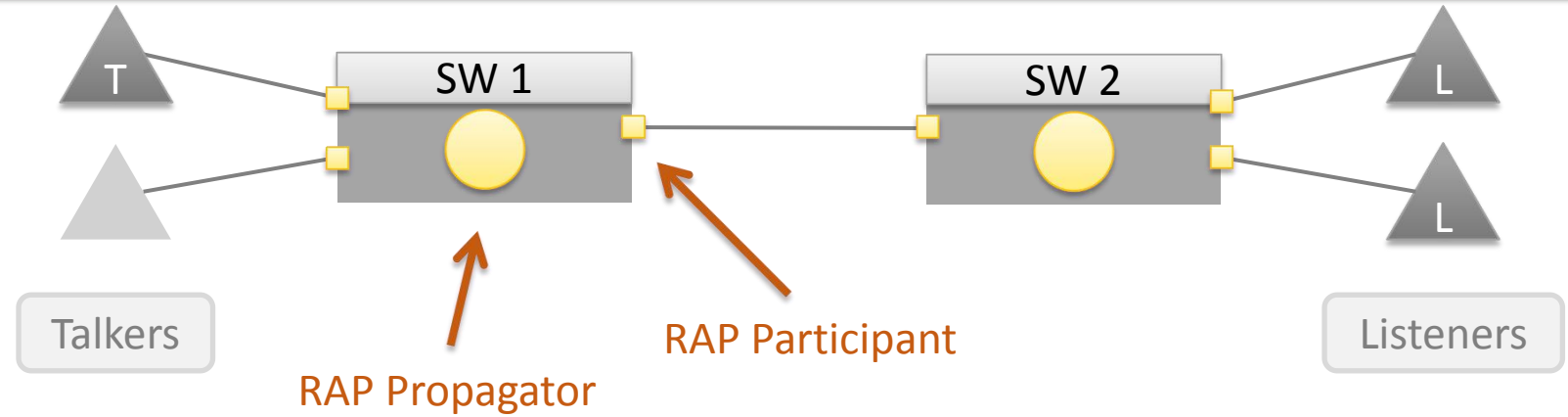
Stream 1

Stream 2

Configuration

RA Classes:

ID=7 , Priority=7, RTID=00-80-C2-01 (TSA=SP, TSpec=TokenBucket)
ID=6 , Priority=6, RTID=00-80-C2-01 (TSA=SP, TSpec=TokenBucket)
ID=5 , Priority=5, RTID=00-80-C2-01 (TSA=SP, TSpec=TokenBucket)
ID=4 , Priority=4, RTID=00-80-C2-01 (TSA=SP, TSpec=TokenBucket)



only this is important here

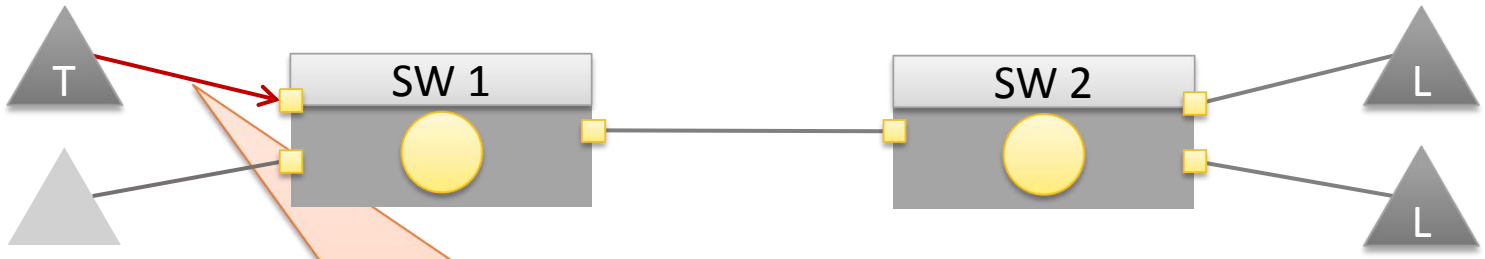
Max delays (every port pair):

Class 7: max 150 µs
Class 6: max 500 µs
Class 5: max 10 ms
Class 4: max 50 ms

Max data rates (every port):

Class 7: max 50 Mbits/s
Class 6: max 100 Mbits/s
Class 5: max 100 Mbits/s
Class 4: max 200 Mbits/s

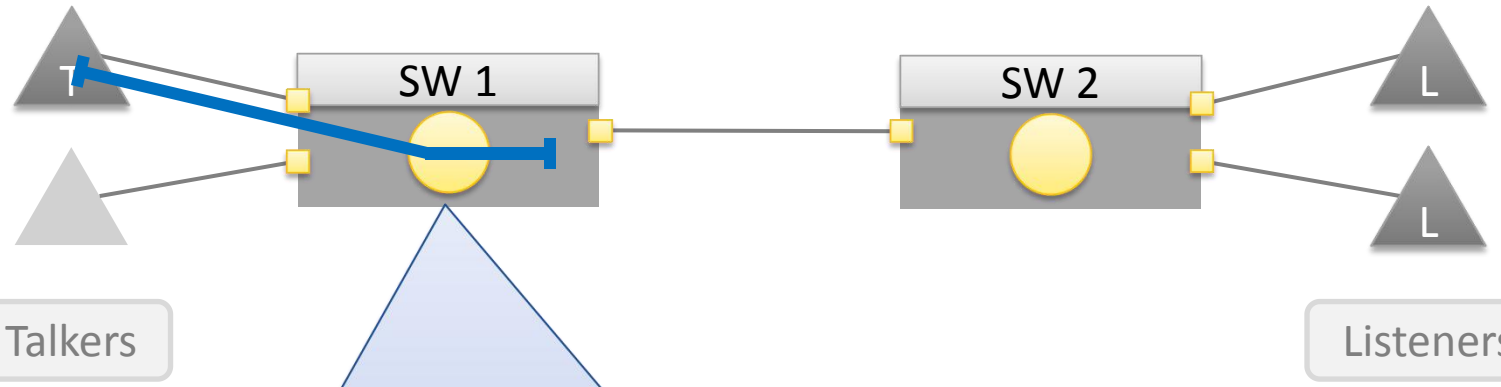
Stream 1




```
TalkerAnnounce:
    StreamId: "B7:77:19:07:B4:18:00:01"
    StreamRank: 1
    AccMaxLatency: 0 ns
    AccMinLatency: 0 ns
    DataFrameParams:
        DestinationMacAddress: "01:00:5E:00:00:01"
        Priority: 7
        VID: 5
    TSpec:
        MaxTransmittedFrameLength: 1000 Bytes
        MinTransmittedFrameLength: 64 Bytes
        CommittedBurstSize: 8160 bits
        CommittedInformationRate: 220000 bits/s
    FailureInfo:
        None
```

These values depend on the talker and are not specified in RAP yet

Bounds Check on SW 1 (EgressPort: SW 2)



configured threshold

Max delays (every port pair):
Class 7: max 150 µs
Class 6: max 500 µs
Class 5: max 10 ms
Class 4: max 50 ms

DelayBoundsCheck:

Traffic Class	MaxHopLatency	ComputedDelay	Success
4	50000000 ns	31497 ns	True
5	10000000 ns	22697 ns	True
6	500000 ns	20607 ns	True
7	150000 ns	20497 ns	True

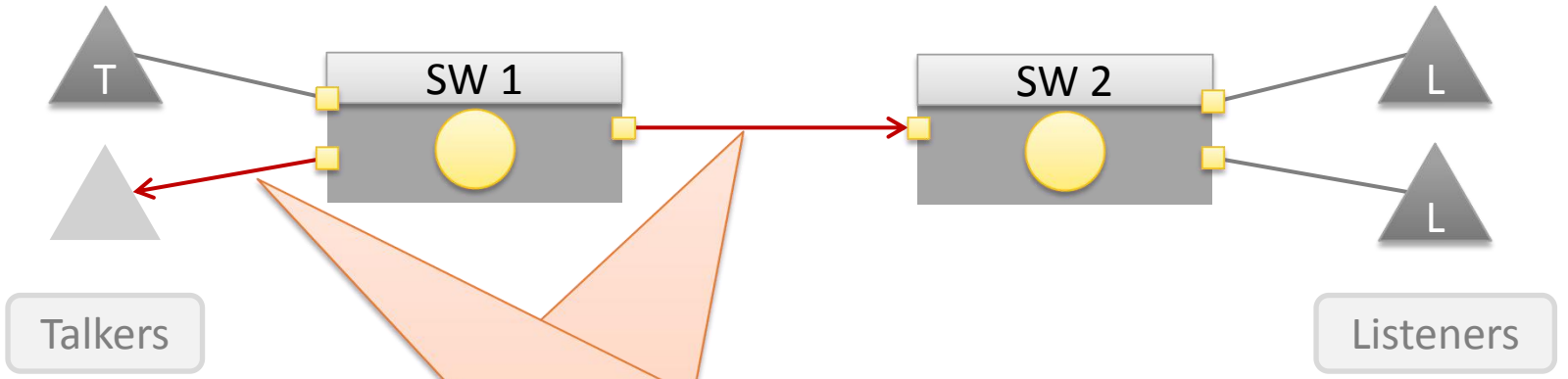
current delay bound for all reserved streams plus the new stream

(latency math/model)

BandwidthBoundsCheck (not shown here) → **Success: True**

InternalResourcesCheck (not shown here) → **Success: True**

Adjusted TA is Propagated to other Ports



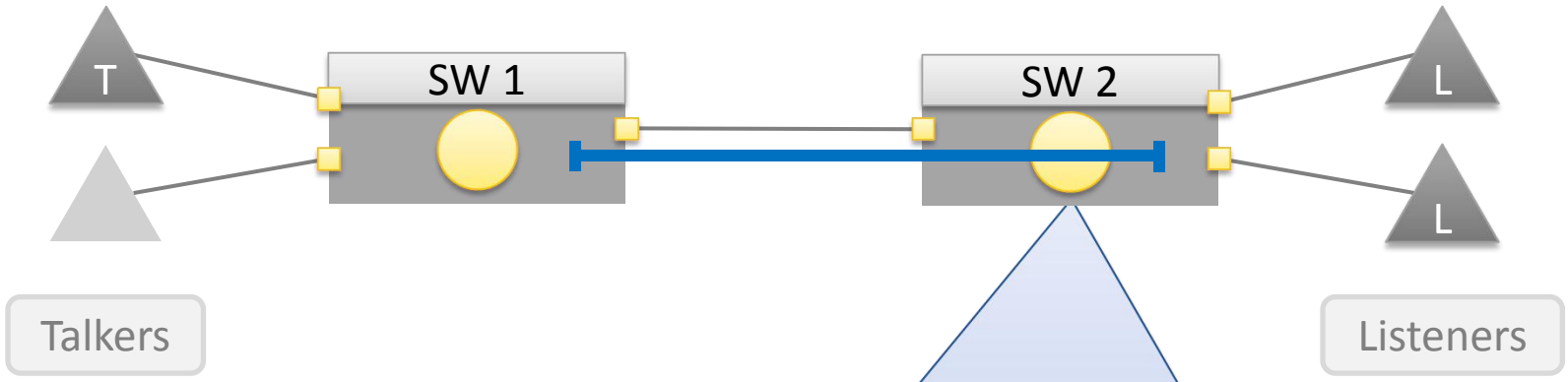

```
TalkerAnnounce:
    StreamId: "B7:77:19:07:B4:18:00:01"
    StreamRank: 1
    AccMaxLatency: 150000 ns
    AccMinLatency: 10512 ns
    DataFrameParams:
        DestinationMacAddress: "01:00:5E:00:00:01"
        Priority: 7
        VID: 5
    TSpec:
        MaxTransmittedFrameLength: 1000 Bytes
        MinTransmittedFrameLength: 64 Bytes
        CommittedBurstSize: 8160 bits
        CommittedInformationRate: 220000 bits/s
    FailureInfo:
        None
```

MaxDelay

Max delays (every port pair):
Class 7: max 150 µs
Class 6: max 500 µs
Class 5: max 10 ms
Class 4: max 50 ms

MinFrameLength / LinkSpeed
+ MinBridgeProcessingDelay (10µs)
+ MinPropagationDelay (0)
(used for more accurate latency calculations)

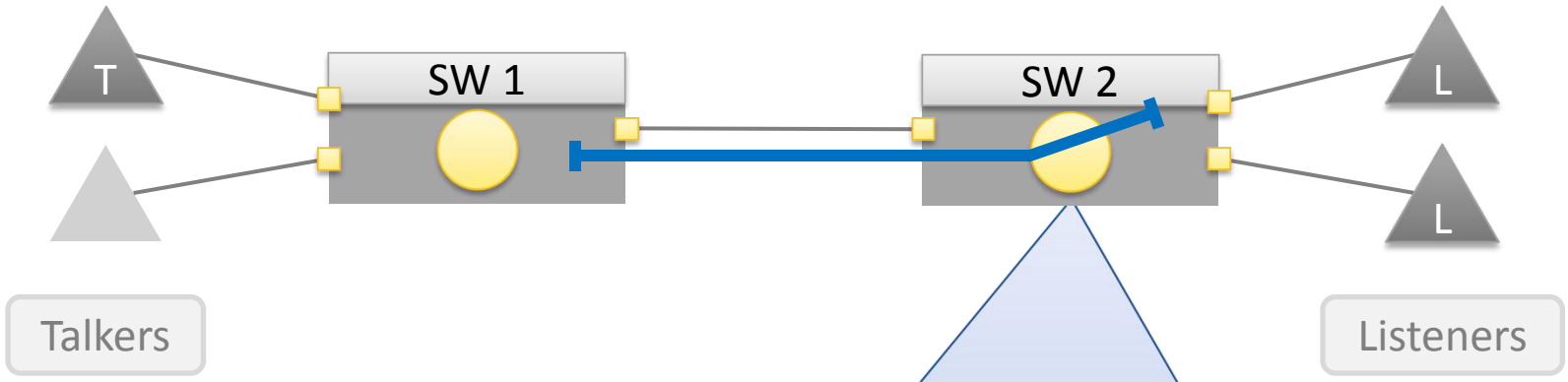
Bounds Check on SW 2 (EgressPort: Listener 2)



DelayBoundsCheck:

Traffic Class	MaxHopLatency	ComputedDelay	Success
4	50000000 ns	31529 ns	True
5	10000000 ns	22729 ns	True
6	500000 ns	20639 ns	True
7	150000 ns	20529 ns	True

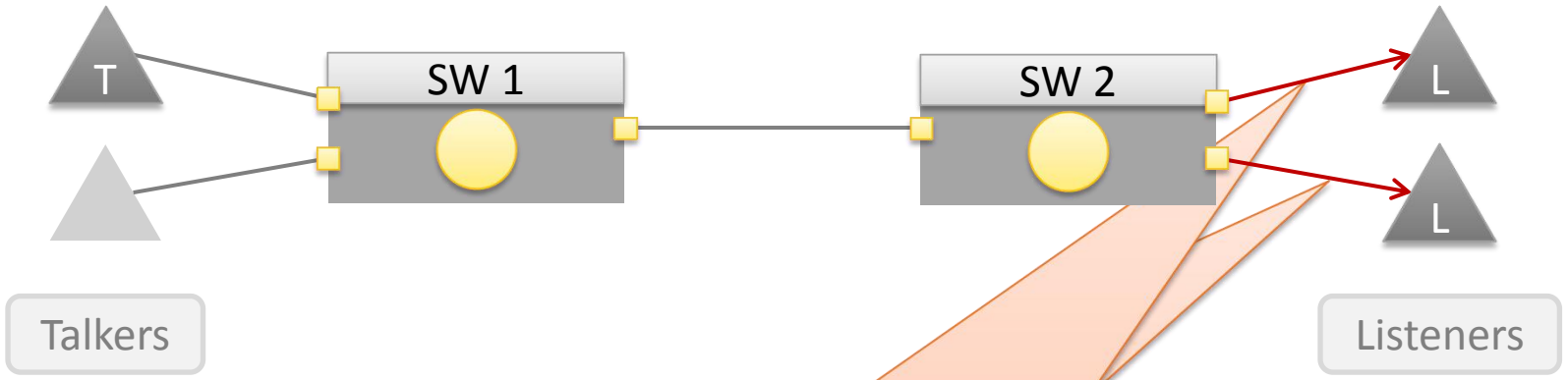
Bounds Check on SW 2 (EgressPort: Listener 1)



DelayBoundsCheck:

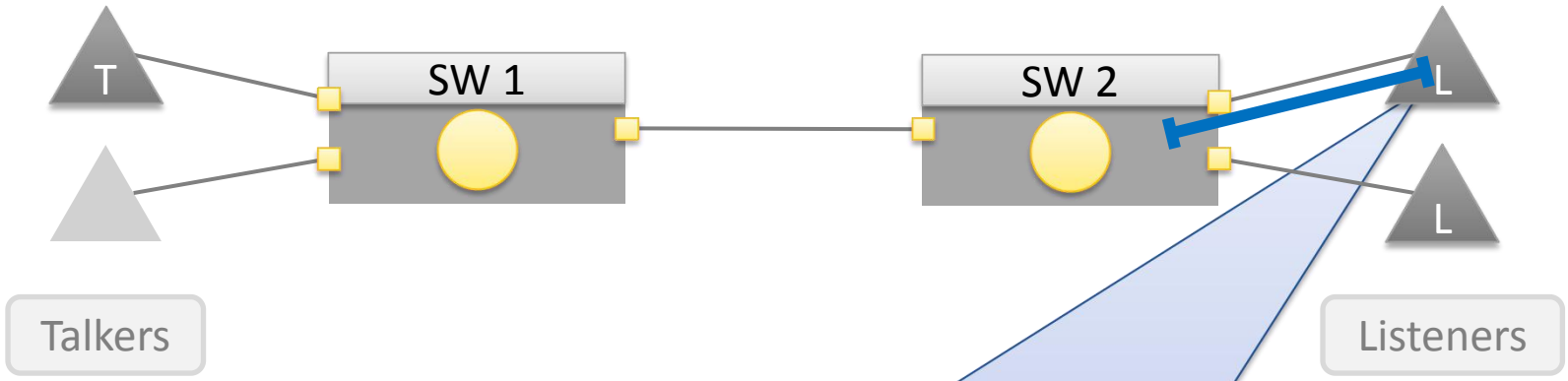
Traffic Class	MaxHopLatency	ComputedDelay	Success
4	50000000 ns	31529 ns	True
5	10000000 ns	22729 ns	True
6	500000 ns	20639 ns	True
7	150000 ns	20529 ns	True

Adjusted TA is Propagated to other Ports




```
TalkerAnnounce:
    StreamId: "B7:77:19:07:B4:18:00:01"
    StreamRank: 1
    AccMaxLatency: 300000 ns
    AccMinLatency: 21024 ns
    DataFrameParams:
        DestinationMacAddress: "01:00:5E:00:00:01"
        Priority: 7
        VID: 5
    TSpec:
        MaxTransmittedFrameLength: 1000 Bytes
        MinTransmittedFrameLength: 64 Bytes
        CommittedBurstSize: 8160 bits
        CommittedInformationRate: 220000 bits/s
    FailureInfo:
        None
```

Bounds Check on Listener 1

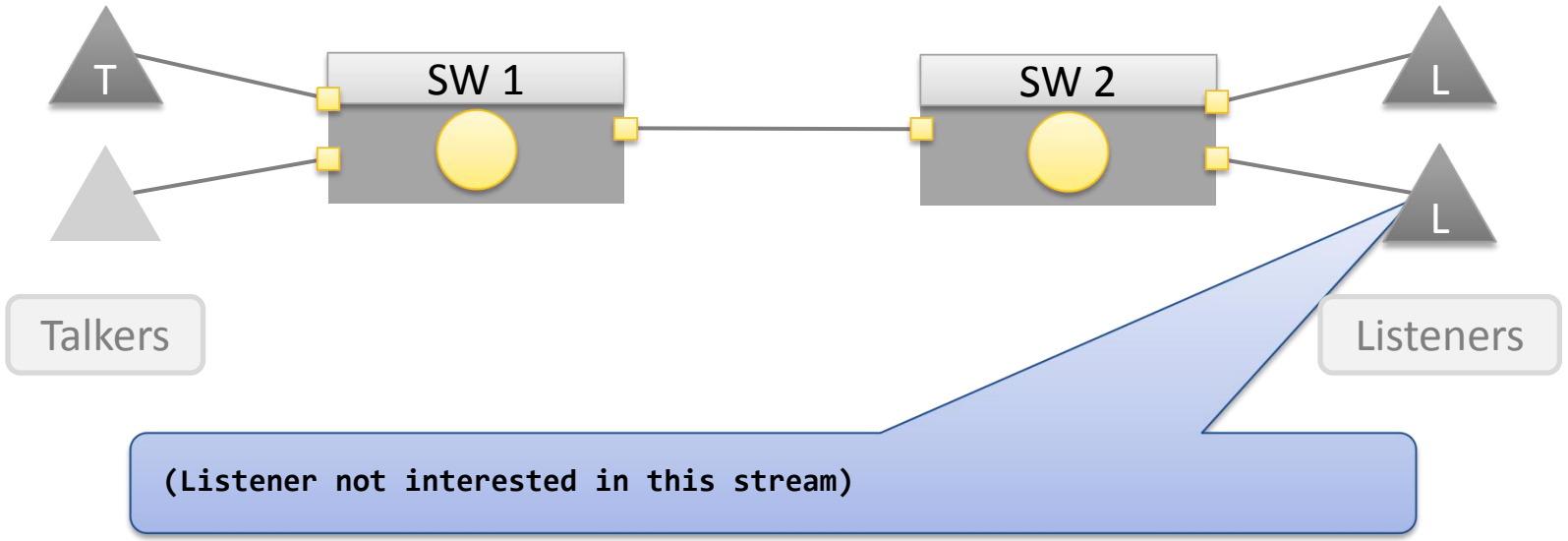


(not specified for end devices in RAP yet)

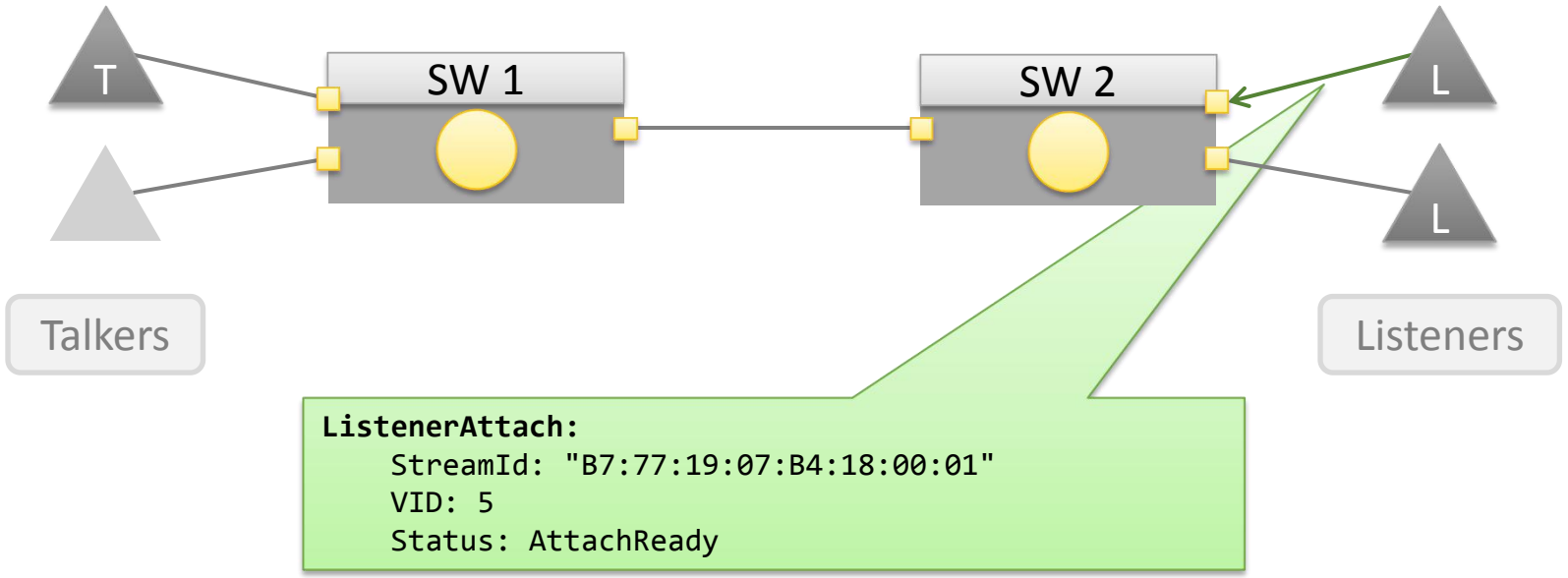
DelayBoundsCheck:

Traffic Class	MaxHopLatency	ComputedDelay	Success
4	50000000 ns	31562 ns	True
5	10000000 ns	22762 ns	True
6	500000 ns	20672 ns	True
7	150000 ns	20562 ns	True

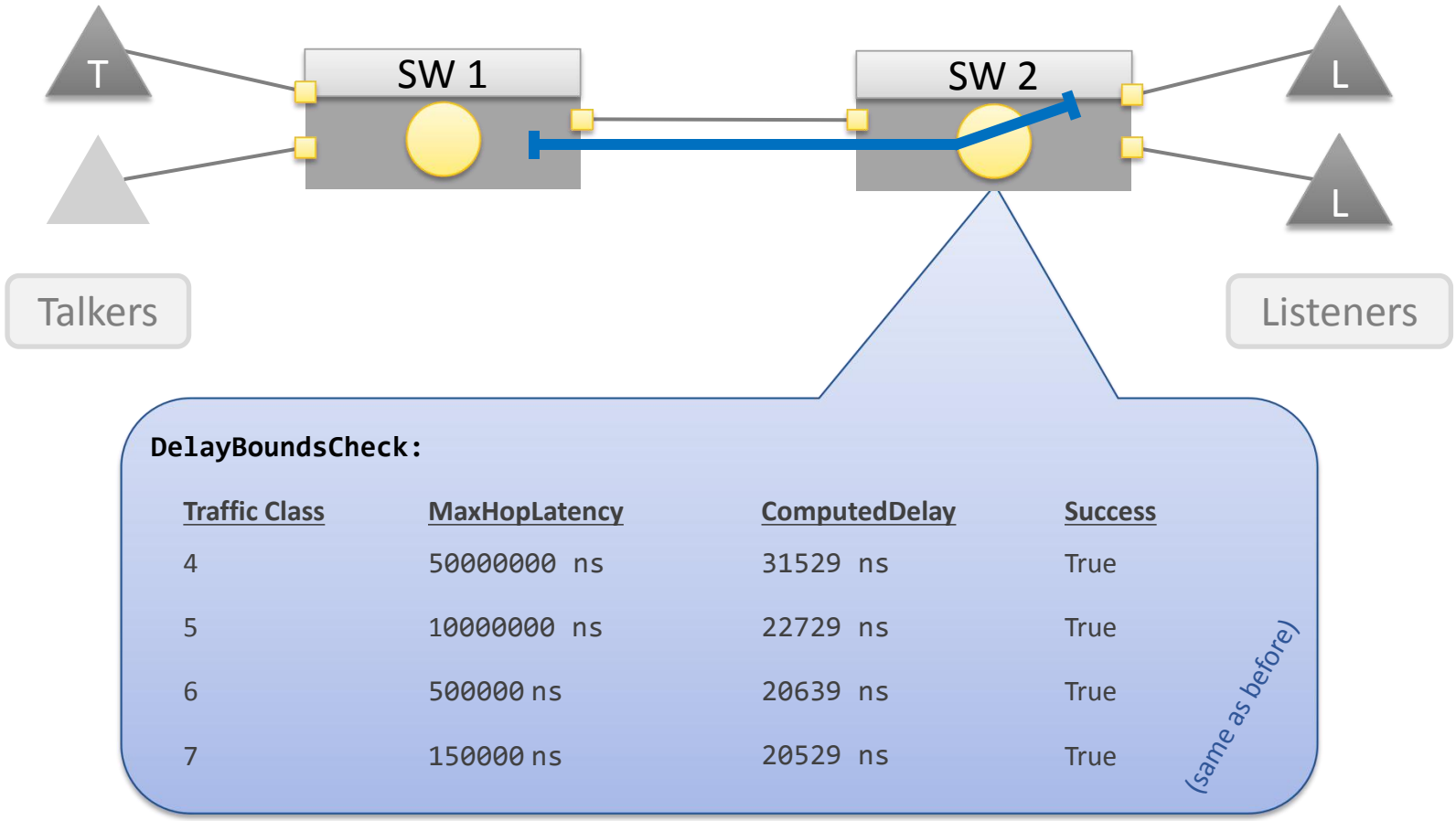
Listener 2 is not Attaching



Listener 1 sends LA



Bounds Check on SW 2 (EgressPort: Listener 1)

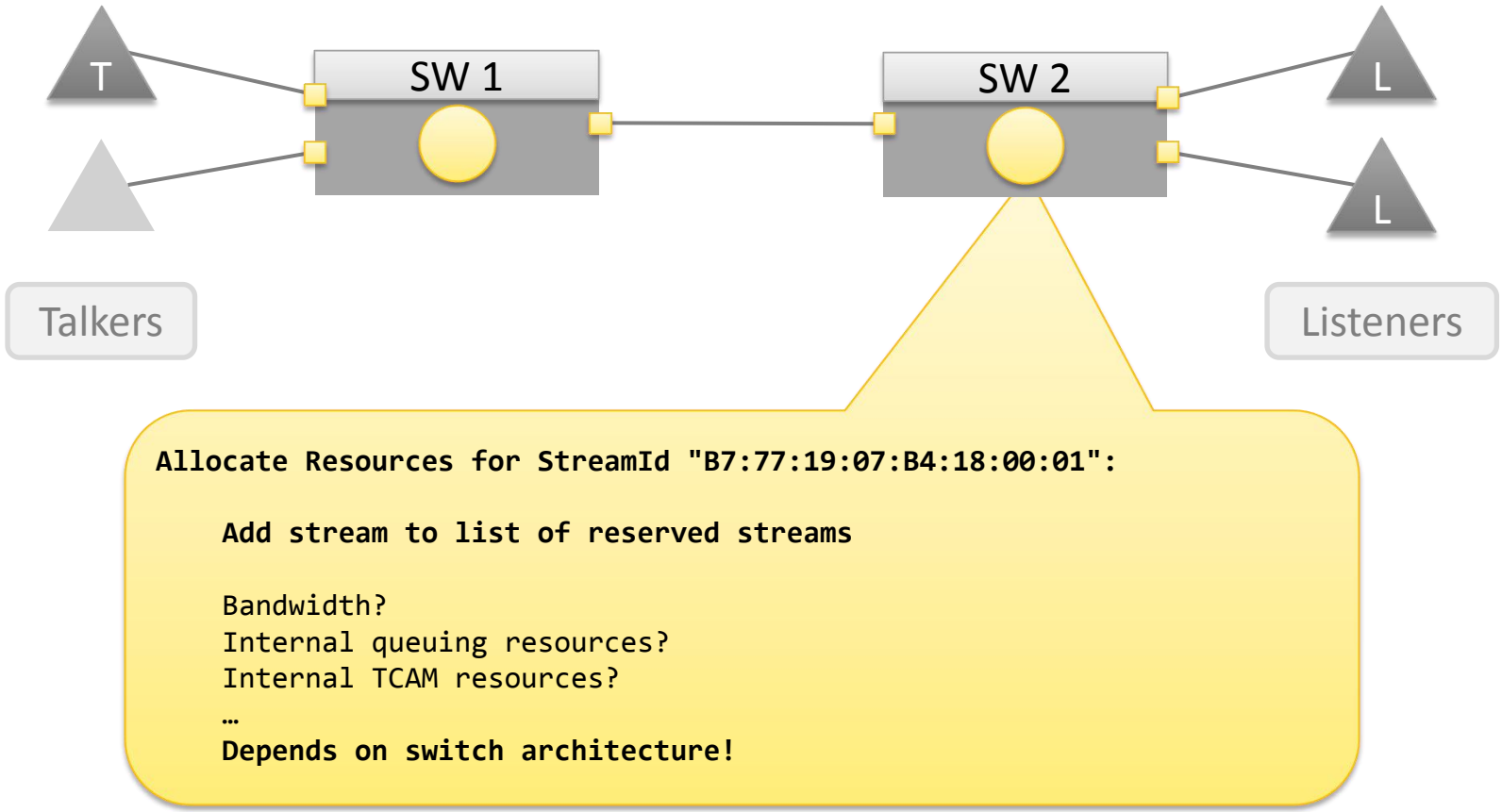


DelayBoundsCheck:

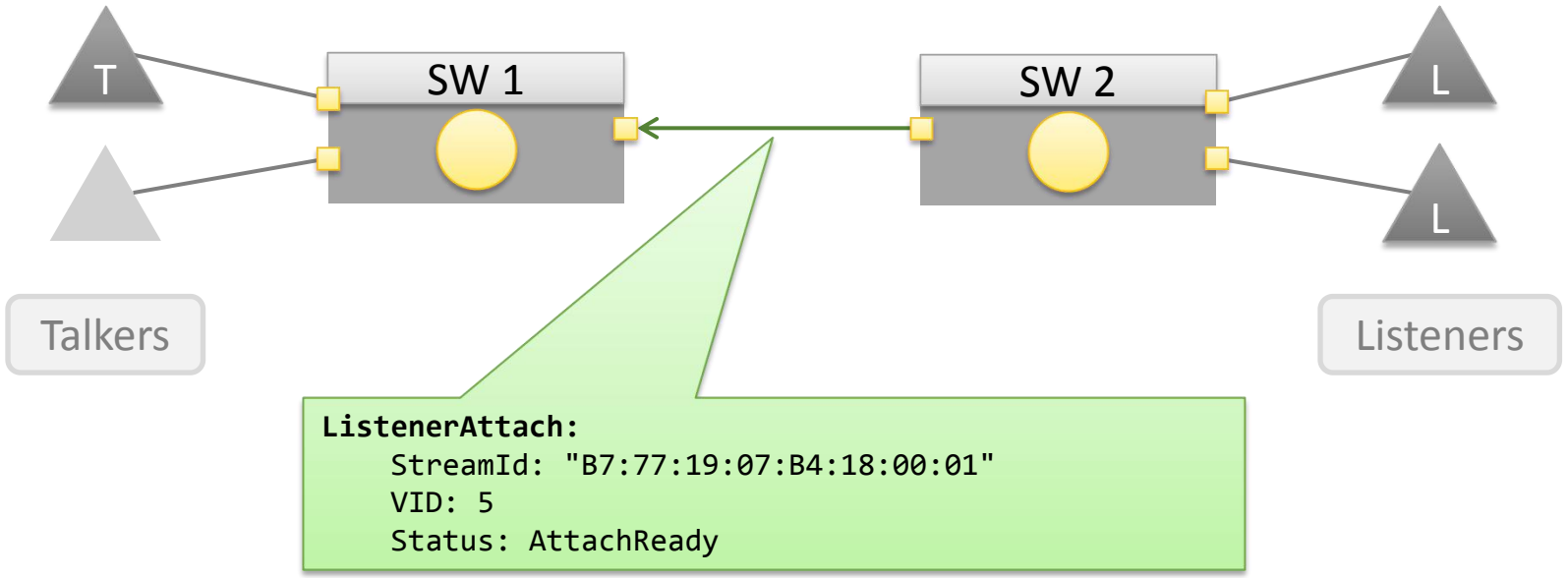
Traffic Class	MaxHopLatency	ComputedDelay	Success
4	50000000 ns	31529 ns	True
5	10000000 ns	22729 ns	True
6	500000 ns	20639 ns	True
7	150000 ns	20529 ns	True

(same as before)

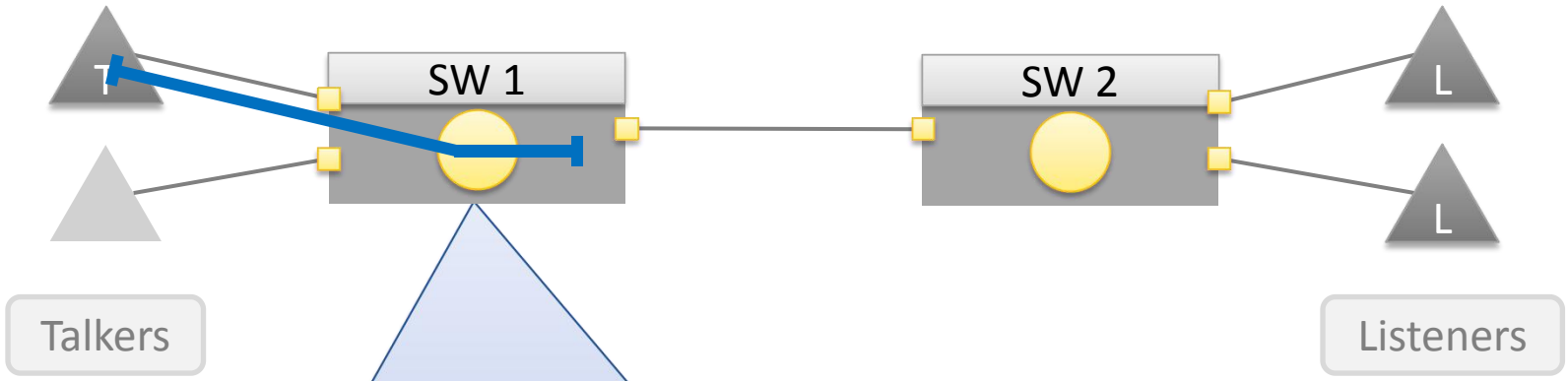
Reservation on SW 2 Successful



SW 2 Forwards the LA to SW 1



Bounds Check on SW 1 (EgressPort: SW 2)

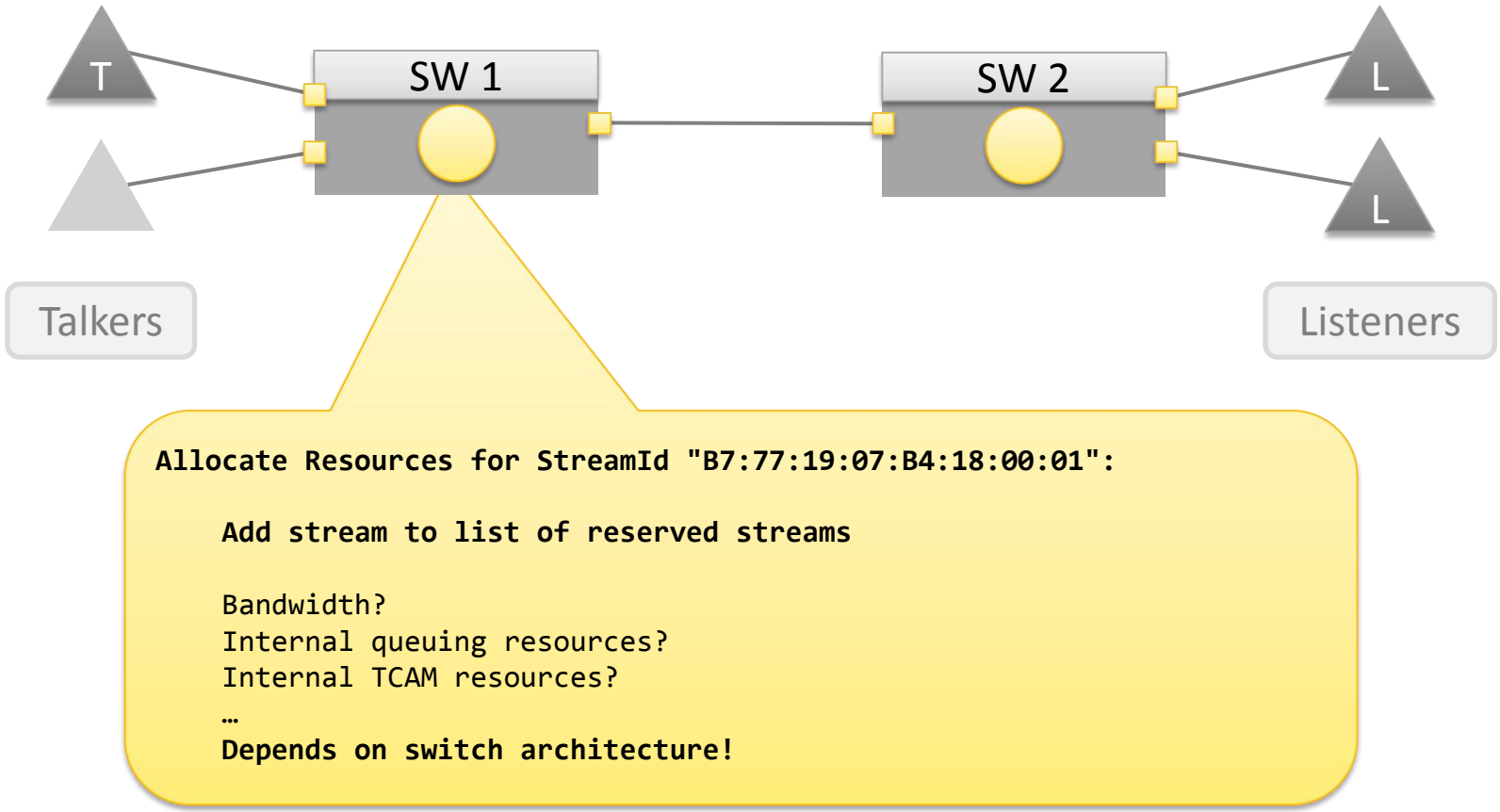


DelayBoundsCheck:

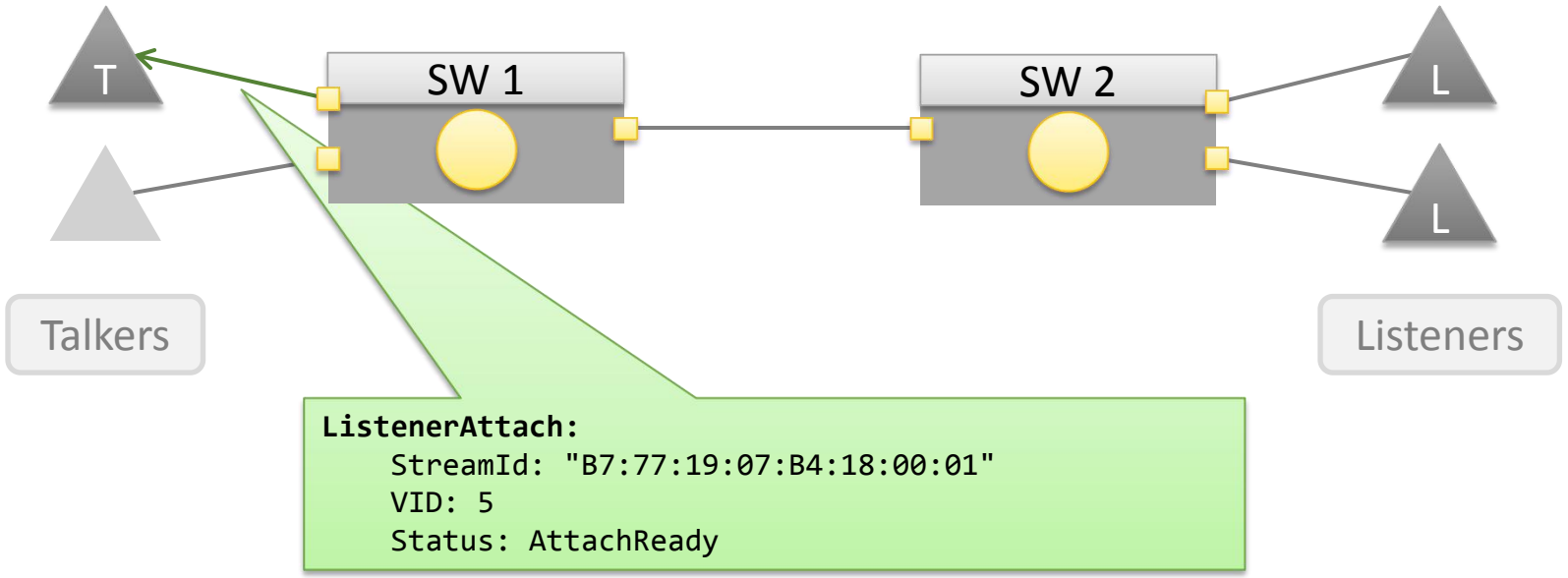
Traffic Class	MaxHopLatency	ComputedDelay	Success
4	50000000 ns	31497 ns	True
5	10000000 ns	22697 ns	True
6	500000 ns	20607 ns	True
7	150000 ns	20497 ns	True

(same as before)

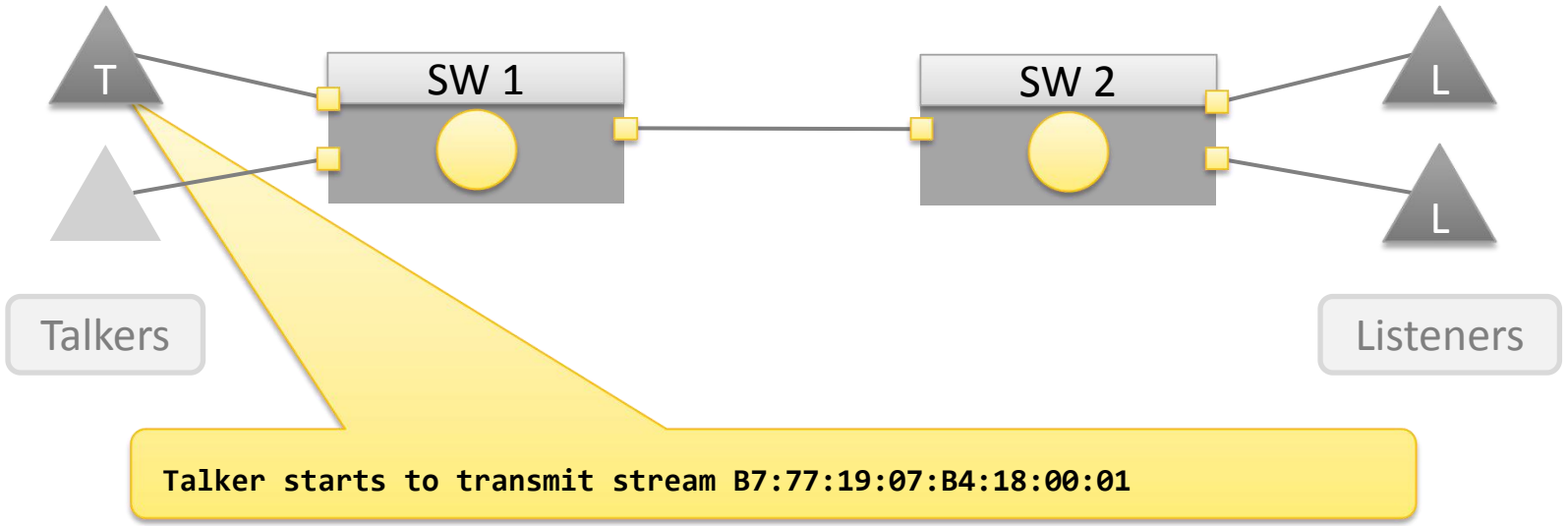
Reservation on SW 1 Successful



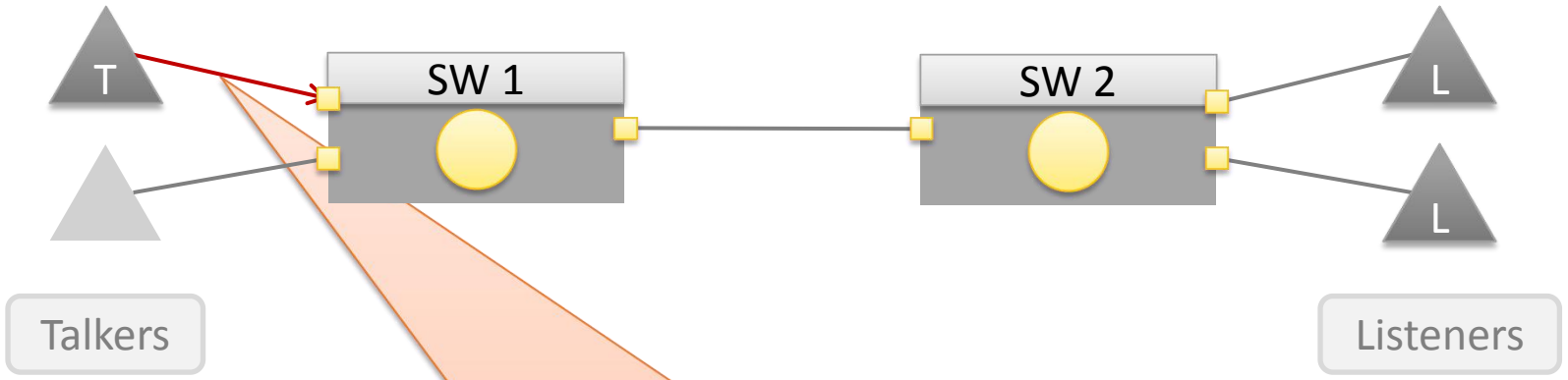
SW 1 Forwards the LA to the Talker



Talker Received Successful LA and Starts to Transmit

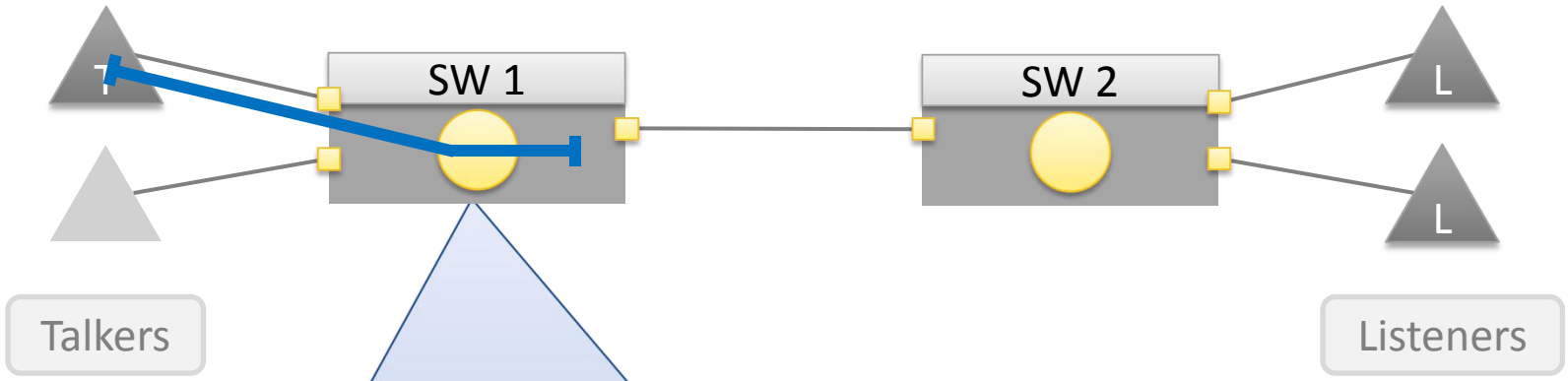


Stream 2




```
TalkerAnnounce:
    StreamId: "55:BB:FD:68:07:10:00:00"
    StreamRank: 1
    AccMaxLatency: 0 ns
    AccMinLatency: 0 ns
    DataFrameParams:
        DestinationMacAddress: "01:00:5E:00:00:00"
        Priority: 7
        VID: 5
    TSpec:
        MaxTransmittedFrameLength: 230 Bytes
        MinTransmittedFrameLength: 100 Bytes
        CommittedBurstSize: 2000 bits
        CommittedInformationRate: 10000000 bits/s
    FailureInfo:
        None
```

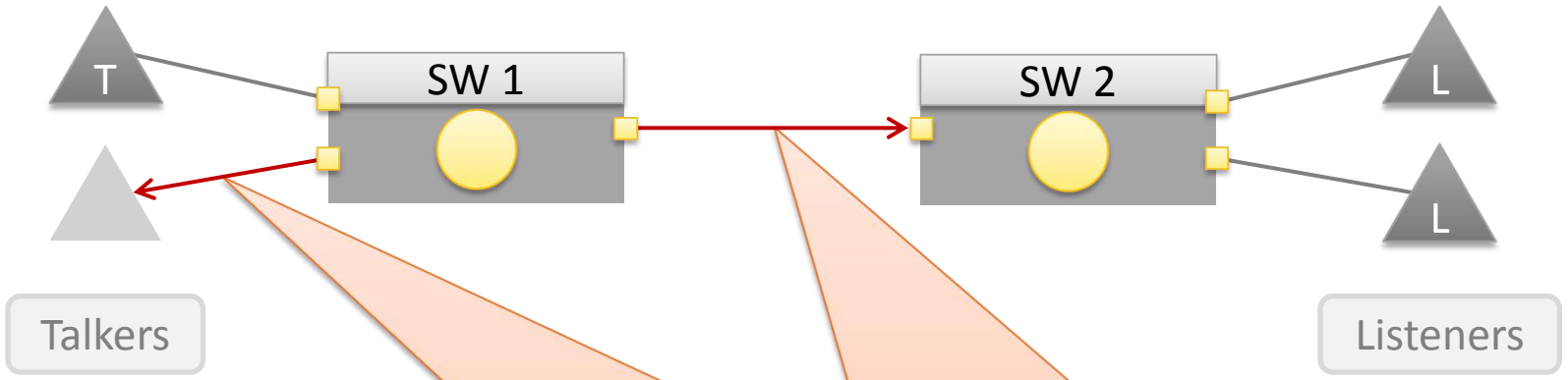
Bounds Check on SW 1 (EgressPort: SW 2)



DelayBoundsCheck:

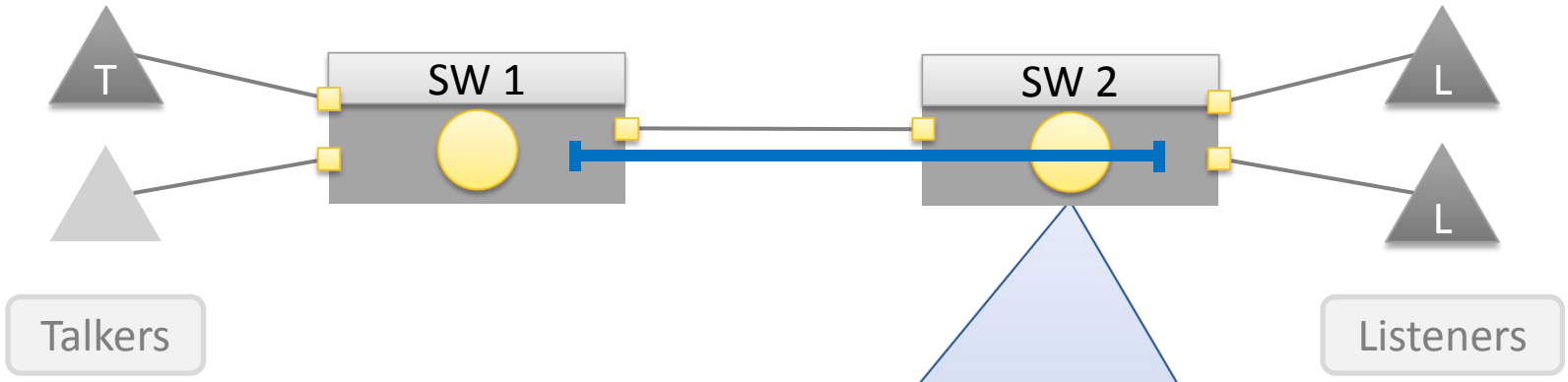
Traffic Class	MaxHopLatency	ComputedDelay	Success
4	50000000 ns	533505 ns	True
5	10000000 ns	124705 ns	True
6	500000 ns	27615 ns	True
7	150000 ns	22505 ns	True

Adjusted TA is Propagated to other Ports




```
TalkerAnnounce:
    StreamId: "55:BB:FD:68:07:10:00:00"
    StreamRank: 1
    AccMaxLatency: 150000 ns
    AccMinLatency: 10800 ns
    DataFrameParams:
        DestinationMacAddress: "01:00:5E:00:00:00"
        Priority: 7
        VID: 5
    TSpec:
        MaxTransmittedFrameLength: 230 Bytes
        MinTransmittedFrameLength: 100 Bytes
        CommittedBurstSize: 2000 bits
        CommittedInformationRate: 10000000 bits/s
    FailureInfo:
        None
```

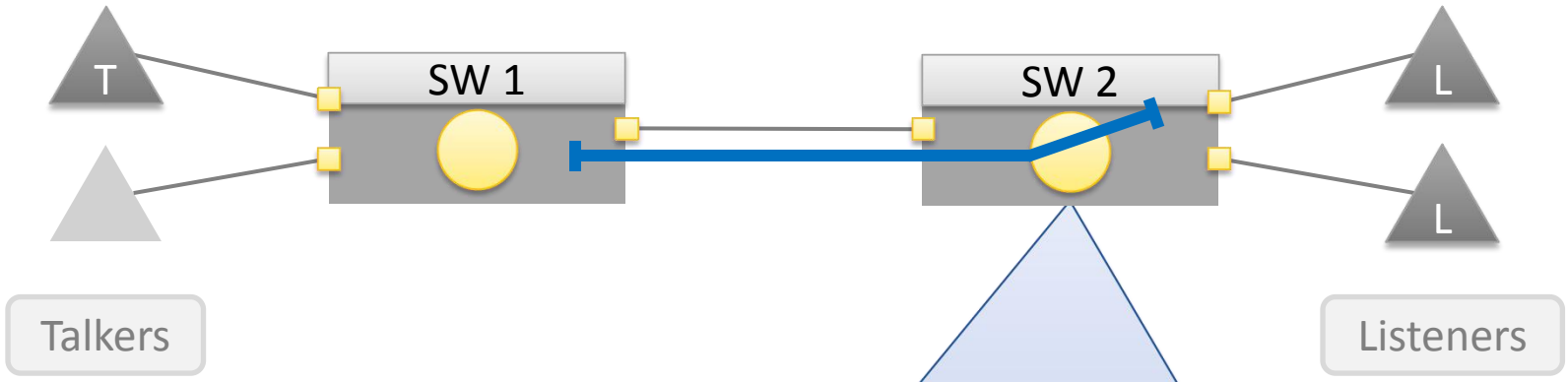
Bounds Check on SW 2 (EgressPort: Listener 2)



DelayBoundsCheck:

Traffic Class	MaxHopLatency	ComputedDelay	Success
4	50000000 ns	535029 ns	True
5	10000000 ns	126229 ns	True
6	500000 ns	29139 ns	True
7	150000 ns	24029 ns	True

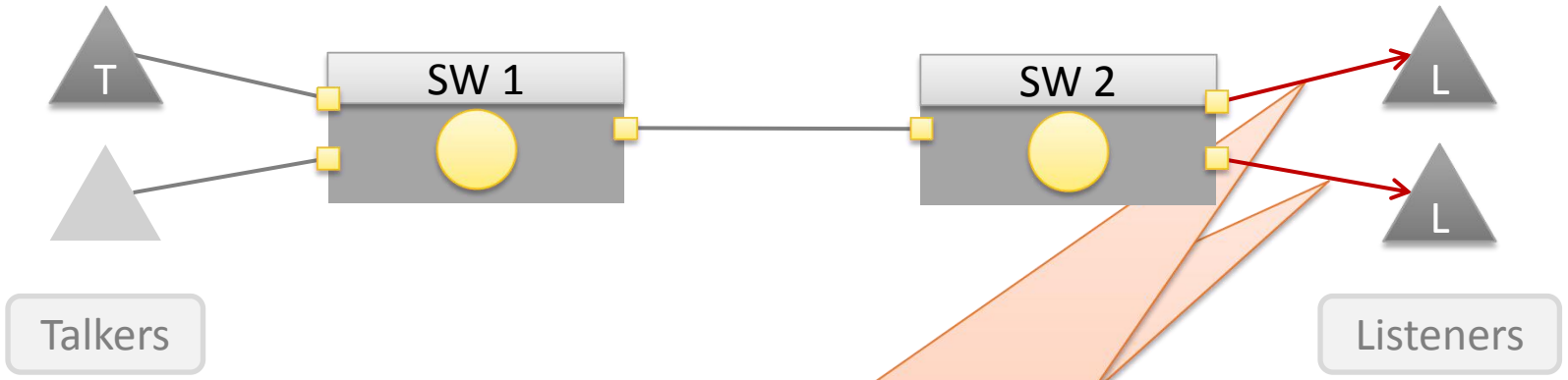
Bounds Check on SW 2 (EgressPort: Listener 1)



DelayBoundsCheck:

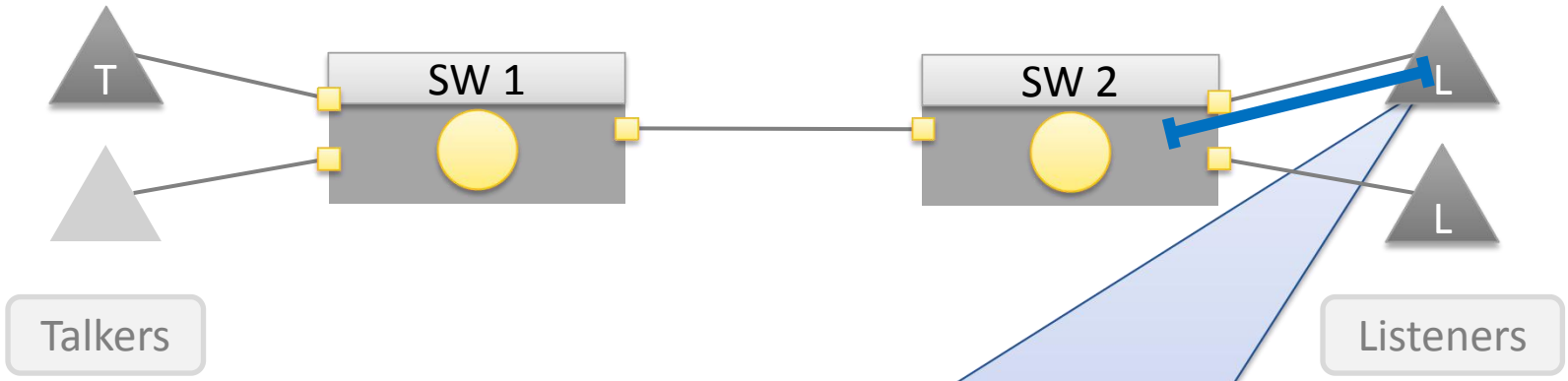
Traffic Class	MaxHopLatency	ComputedDelay	Success
4	50000000 ns	535029 ns	True
5	10000000 ns	126229 ns	True
6	500000 ns	29139 ns	True
7	150000 ns	24029 ns	True

Adjusted TA is Propagated to other Ports




```
TalkerAnnounce:
    StreamId: "55:BB:FD:68:07:10:00:00"
    StreamRank: 1
    AccMaxLatency: 300000 ns
    AccMinLatency: 21600 ns
    DataFrameParams:
        DestinationMacAddress: "01:00:5E:00:00:00"
        Priority: 7
        VID: 5
    TSpec:
        MaxTransmittedFrameLength: 230 Bytes
        MinTransmittedFrameLength: 100 Bytes
        CommittedBurstSize: 2000 bits
        CommittedInformationRate: 10000000 bits/s
    FailureInfo:
        None
```

Bounds Check on Listener 1

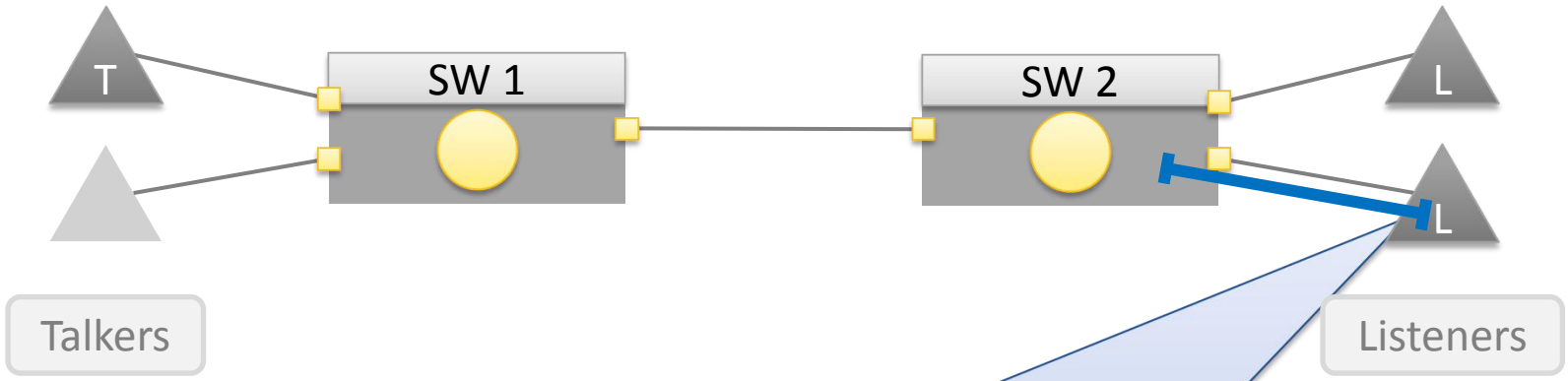


(not specified for end devices in RAP yet)

DelayBoundsCheck:

Traffic Class	MaxHopLatency	ComputedDelay	Success
4	50000000 ns	536554 ns	True
5	10000000 ns	127754 ns	True
6	500000 ns	30664 ns	True
7	150000 ns	25554 ns	True

Bounds Check on Listener 2

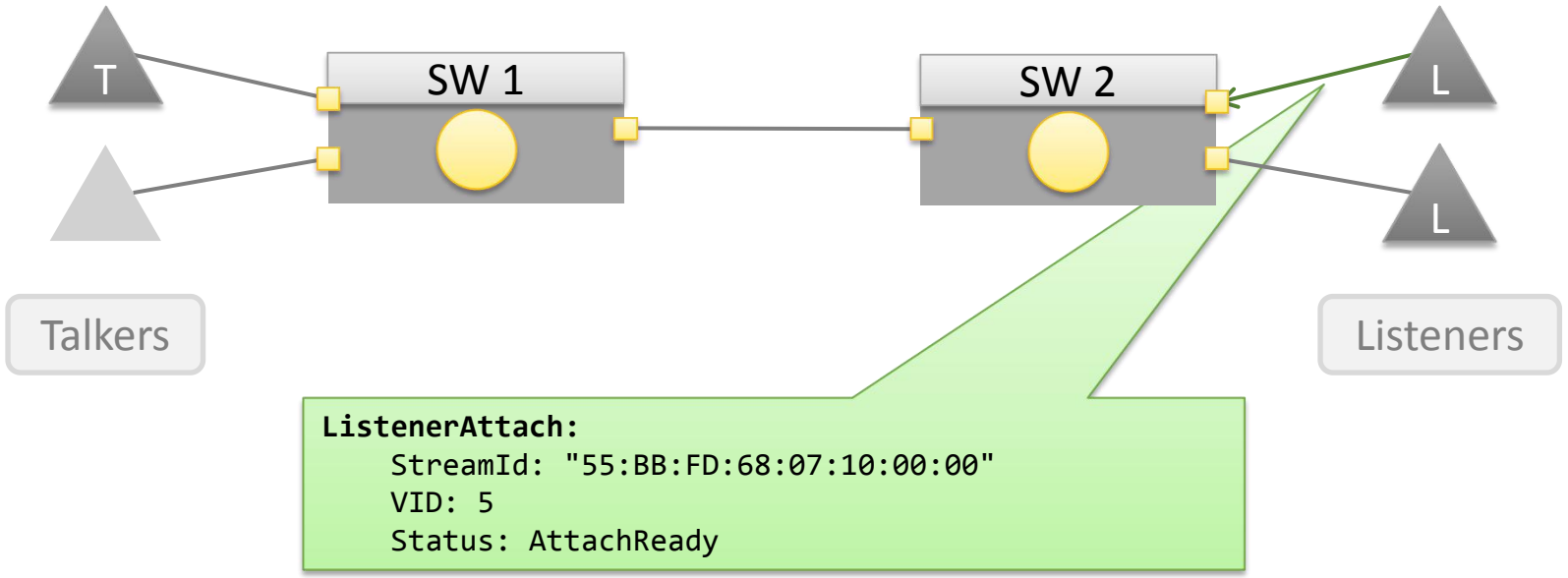


(not specified for end devices in RAP yet)

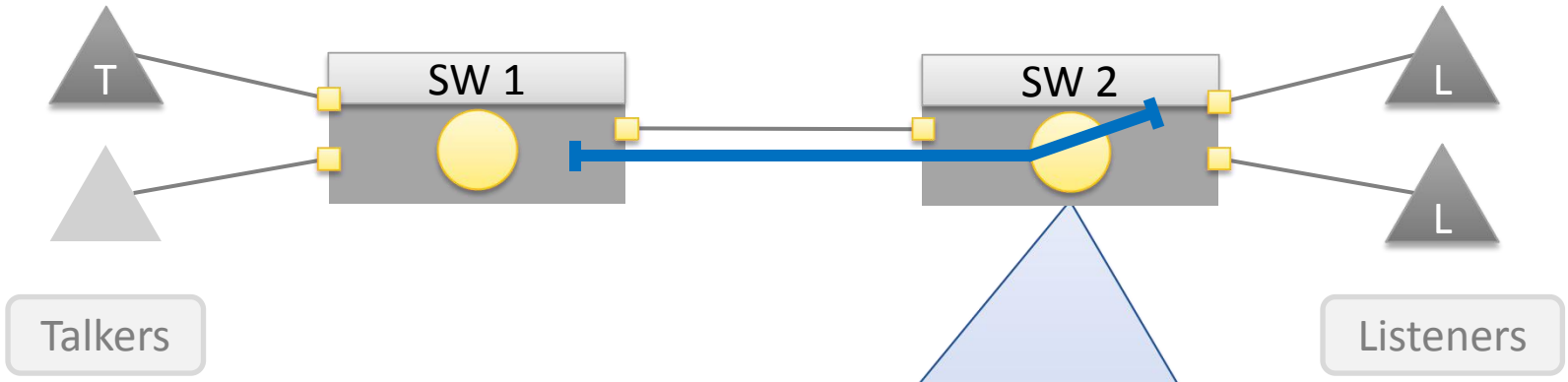
DelayBoundsCheck:

Traffic Class	MaxHopLatency	ComputedDelay	Success
4	50000000 ns	517328 ns	True
5	10000000 ns	117328 ns	True
6	500000 ns	22328 ns	True
7	150000 ns	17328 ns	True

Listener 1 sends LA



Bounds Check on SW 2 (EgressPort: Listener 1)

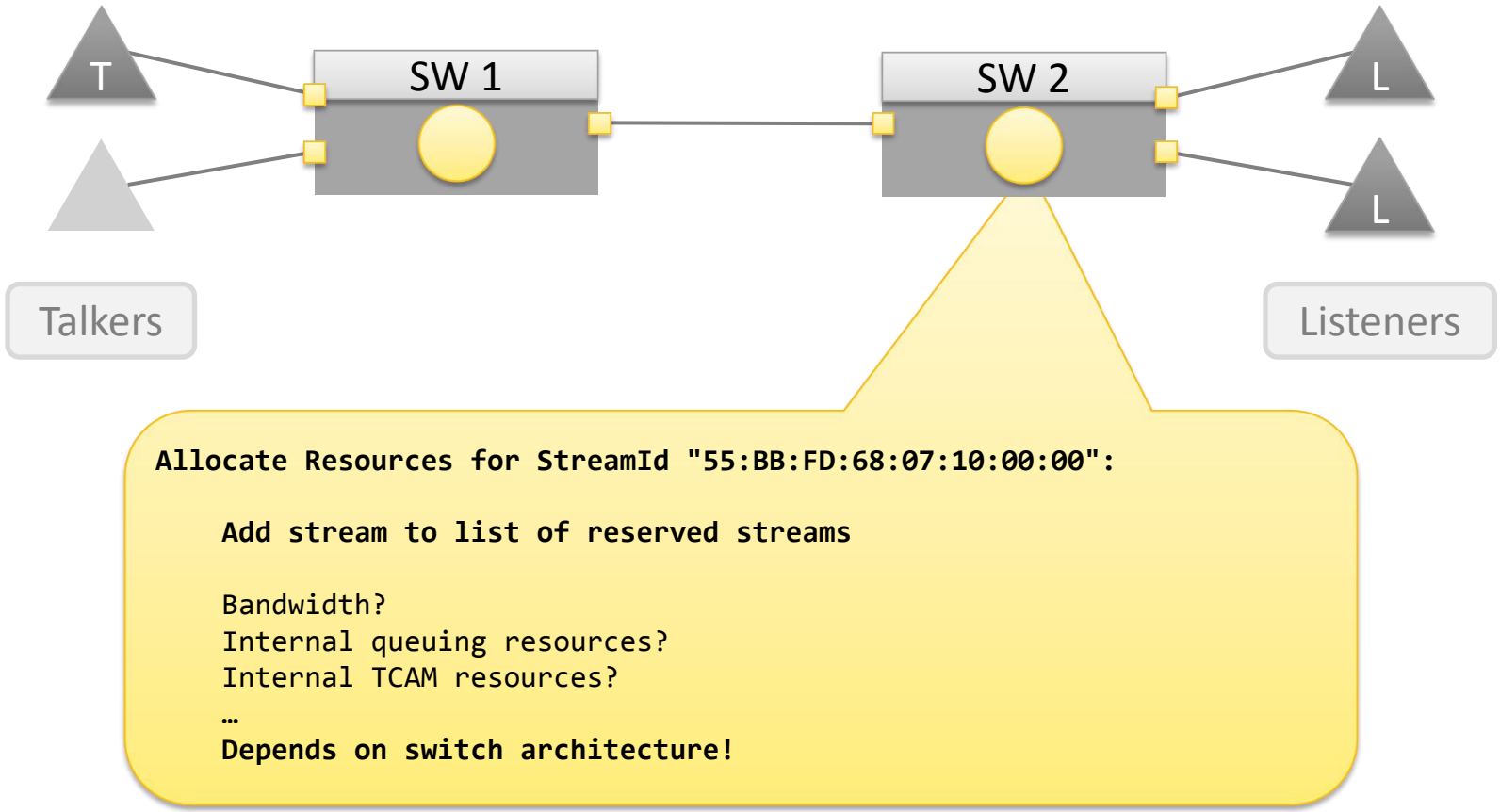


DelayBoundsCheck:

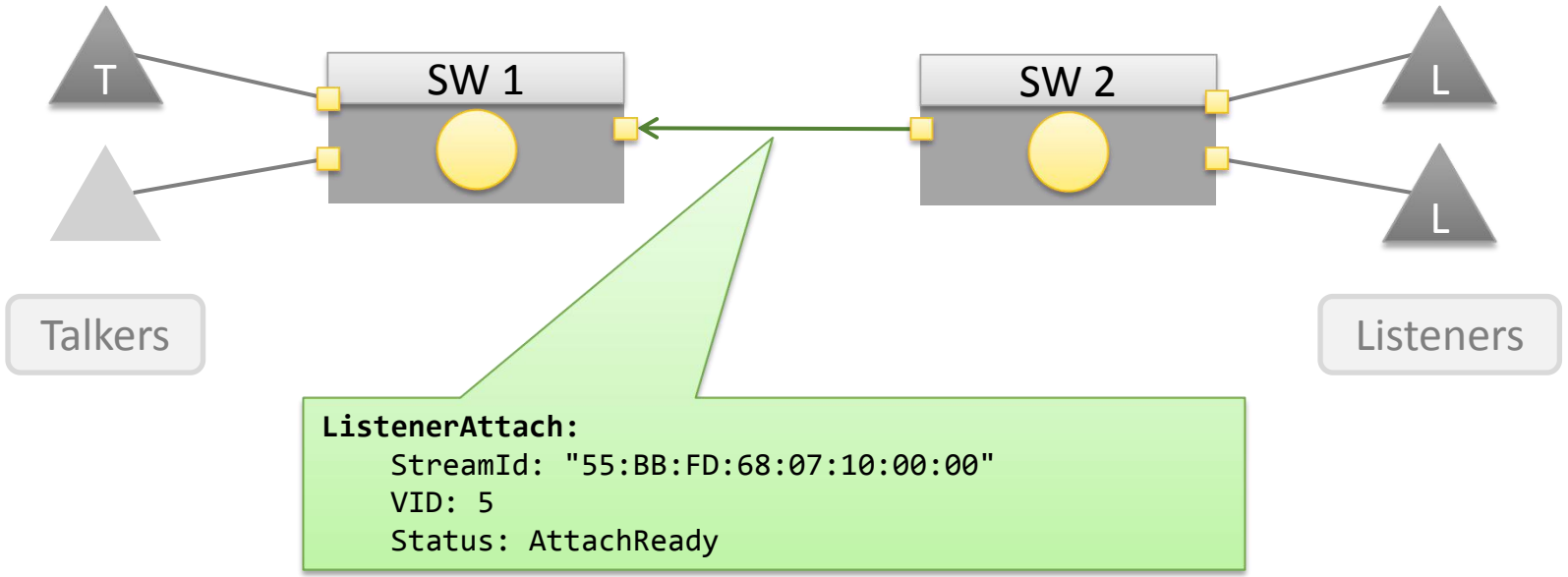
Traffic Class	MaxHopLatency	ComputedDelay	Success
4	50000000 ns	535029 ns	True
5	10000000 ns	126229 ns	True
6	500000 ns	29139 ns	True
7	150000 ns	24029 ns	True

(same as before)

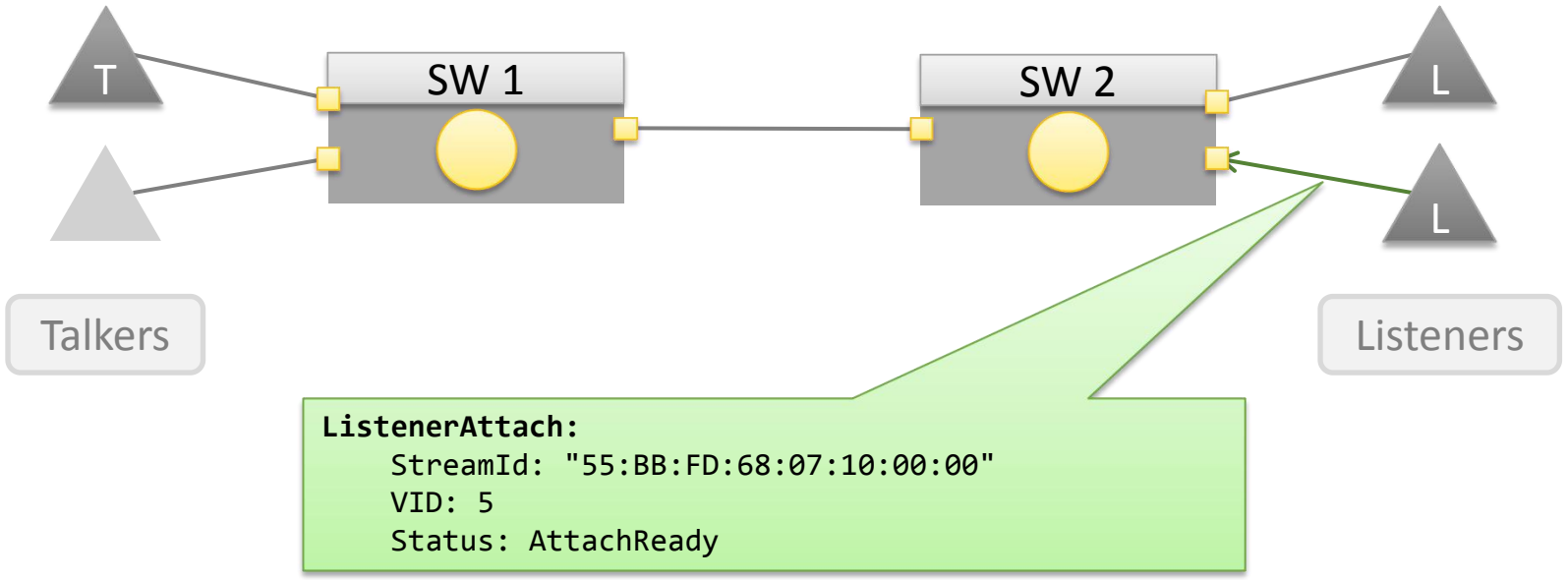
Reservation on SW 2 (Port 1) Successful



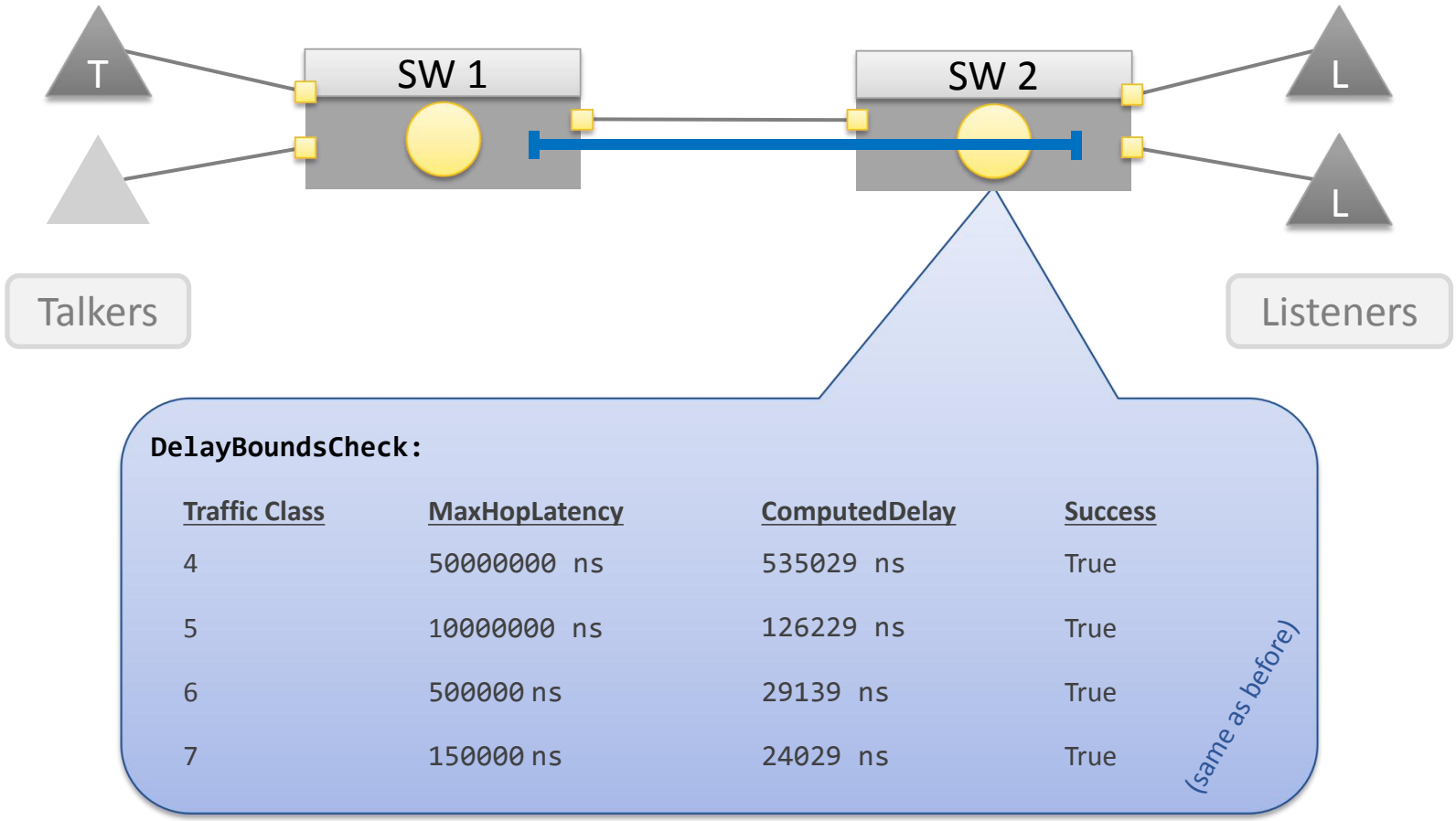
SW 2 Forwards the LA to SW 1



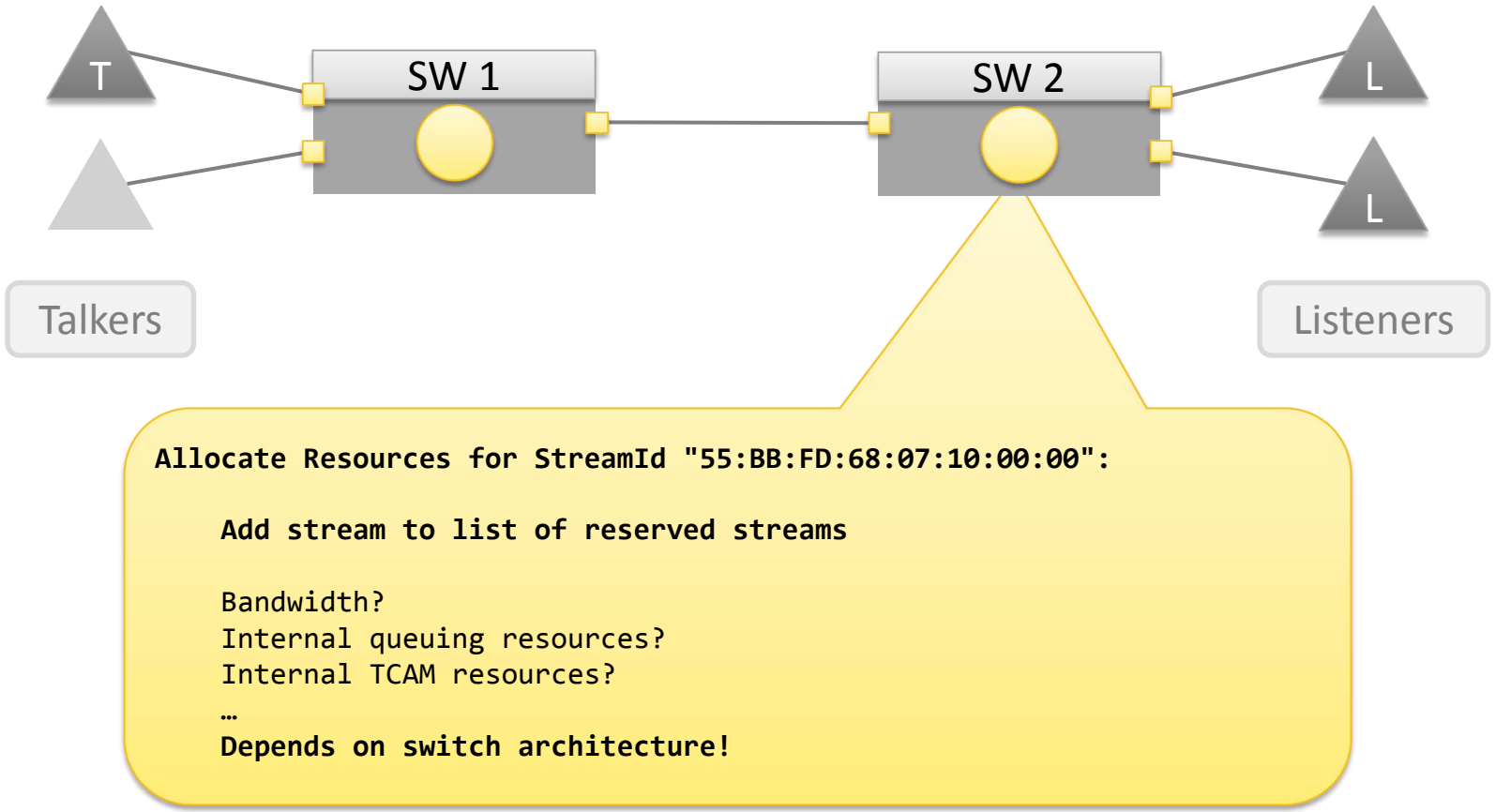
Listener 2 sends LA



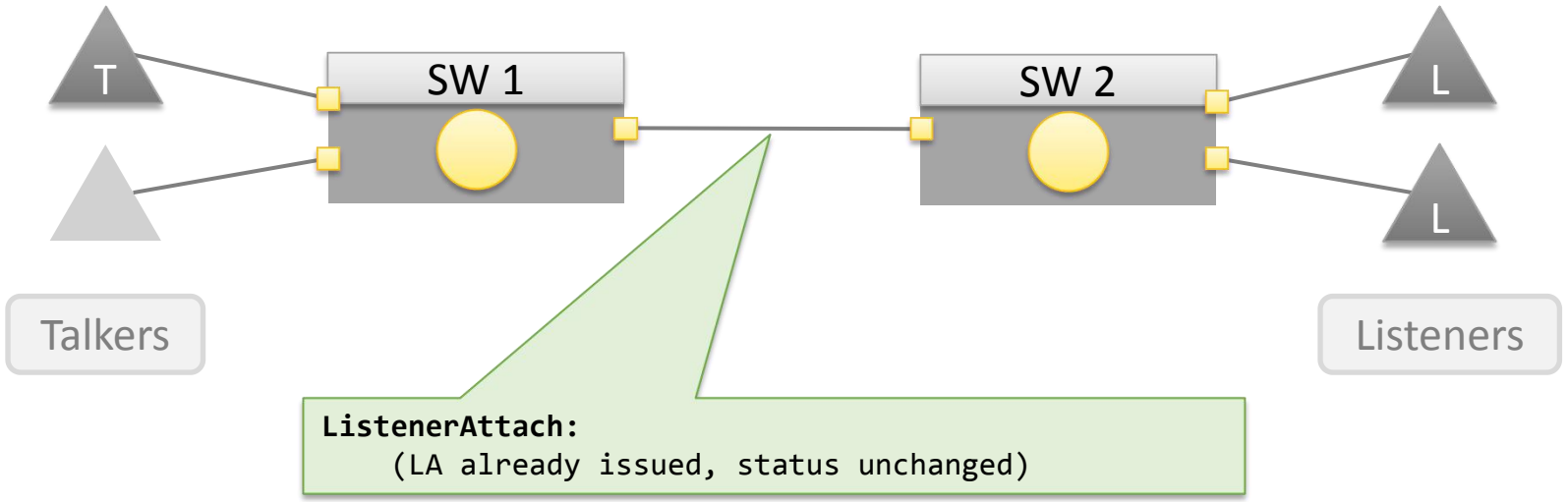
Bounds Check on SW 2 (EgressPort: Listener 2)



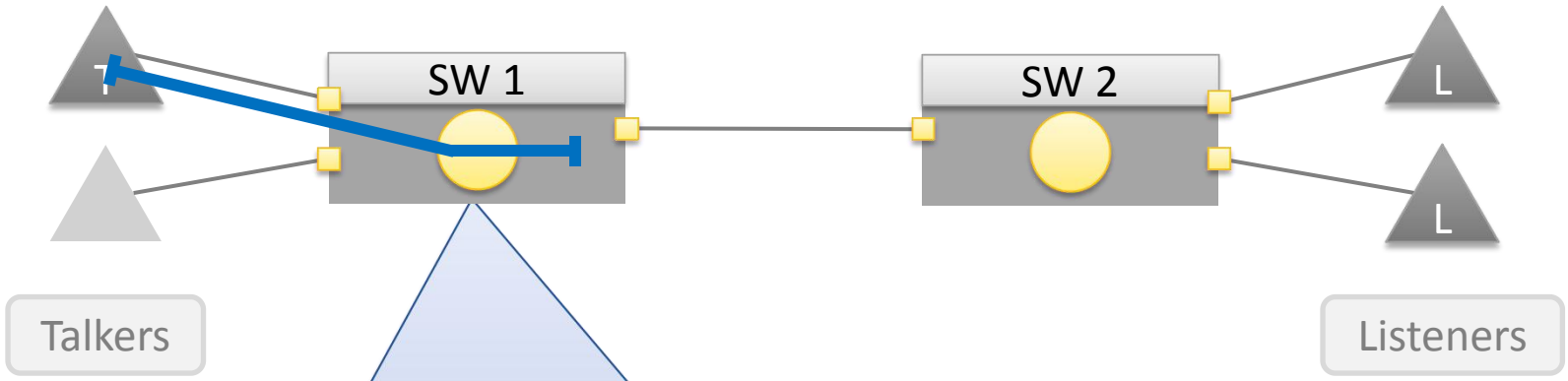
Reservation on SW 2 (Port 2) Successful



SW 2 does Nothing, Existing LA for this Stream Unchanged



Bounds Check on SW 1 (EgressPort: SW 2)

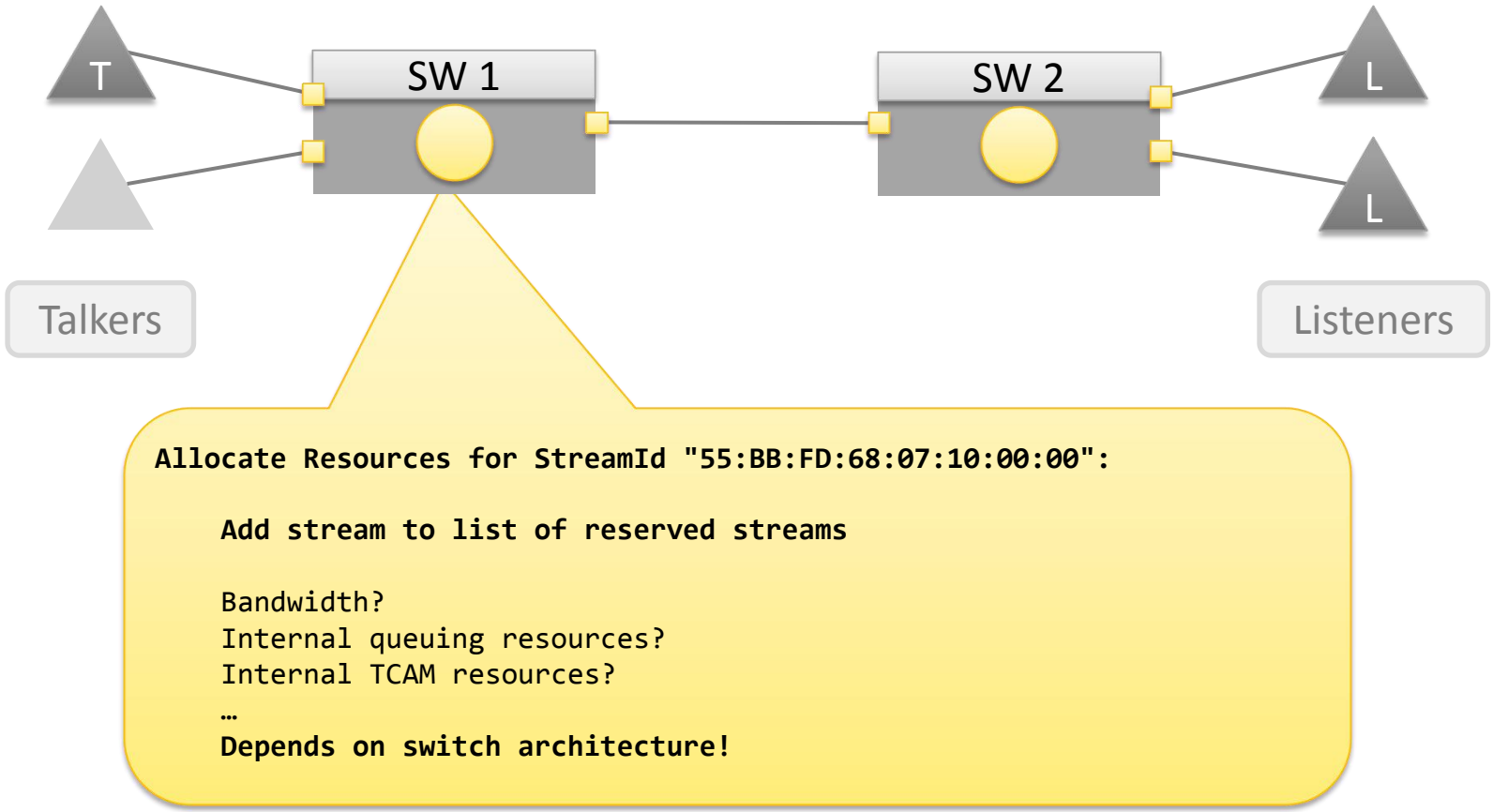


DelayBoundsCheck:

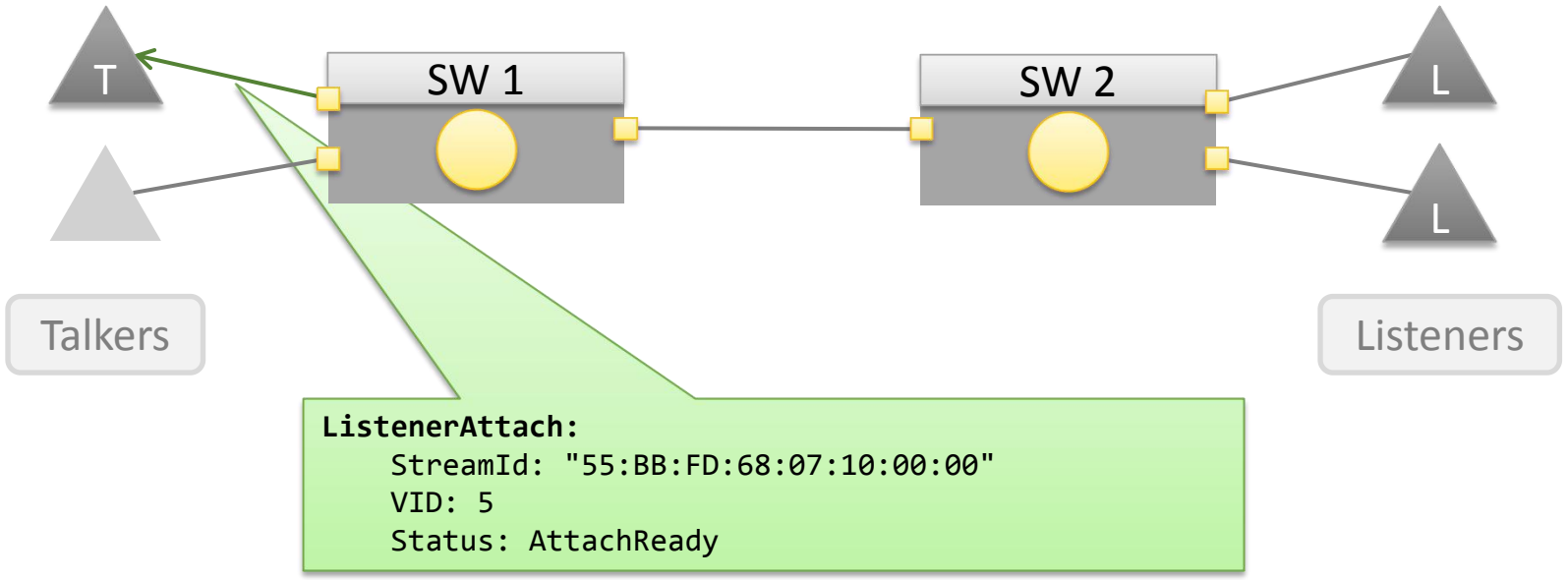
Traffic Class	MaxHopLatency	ComputedDelay	Success
4	50000000 ns	533505 ns	True
5	10000000 ns	124705 ns	True
6	500000 ns	27615 ns	True
7	150000 ns	22505 ns	True

(same as before)

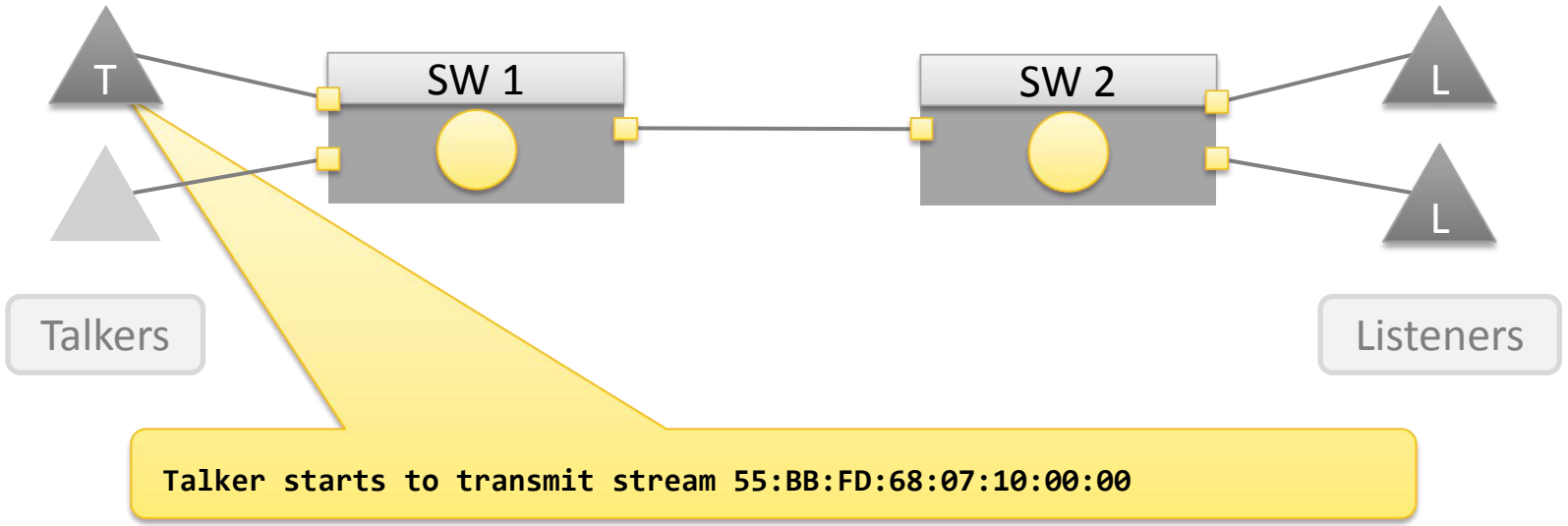
Reservation on SW 1 Successful



SW 1 Forwards the LA to the Talker



Talker Received Successful LA and Starts to Transmit



```
topo = Topology()
    switches = [topo.add_node(Switch("sw1"))]
    switches.append(topo.create_and_add_links(switches[-1], Switch("sw2"), 1e9))
    talkers = [topo.create_and_add_links(switches[0], Host("talker1"), 1e9),
               topo.create_and_add_links(switches[0], Host("talker2"), 1e9)]
    listeners = [topo.create_and_add_links(switches[-1], Host("listener1"), 1e9),
                 topo.create_and_add_links(switches[-1], Host("listener2"), 1e9)]

    # Config
    #              prio = (0,   1,   2,   3,   4,    5,    6,     7    )
    per_hop_guarantees = (inf, inf, inf, inf, 50e6, 10e6, 500e3, 150e3)
    topo.update_guarantees_all_links(per_hop_guarantees)

    # Streams
    streams = []

    stream2 = Stream(label="s1",
                     path=topo.shortest_path(talkers[0], listeners[0]),
                     priority=7,
                     rate=220e3,  # in bits / s
                     burst=1020 * 8,  # bits
                     minFrameSize=64 * 8,  # bits
                     maxFrameSize=1000 * 8)  # bits
    streams.append([stream2])

    # TODO: temporary
    topo.add_stream(stream2)
    for link, tup in apply_model_to_topology(topo, "sp_simple").items():
        print(f"{link.name}: \t  {tup}")

    stream1 = Stream(label="s0",
                     path=topo.shortest_path(talkers[0], listeners[0]),
                     priority=7,
                     rate=10e6,  # in bits / s
                     burst=250 * 8,  # bits
                     minFrameSize=100 * 8,  # bits
                     maxFrameSize=230 * 8)  # bits
    streams.append([stream1])

    # Dirty hack for multicast streams
    stream1_2 = same_stream_different_listener(topo, stream1, listeners[1])
    streams[-1].append(stream1_2)
```

```
def main():
    # Topology
    topo = Topology()
    switches = [topo.add_node(Switch("sw1"))]
    switches.append(topo.create_and_add_links(switches[-1], Switch("sw2"), 1e9))
    talkers = [topo.create_and_add_links(switches[0], Host("talker1"), 1e9),
               topo.create_and_add_links(switches[0], Host("talker2"), 1e9)]
    listeners = [topo.create_and_add_links(switches[-1], Host("listener1"), 1e9),
                 topo.create_and_add_links(switches[-1], Host("listener2"), 1e9)]

    # Config
    #              prio = (0,   1,   2,   3,   4,    5,    6,     7    )
    per_hop_guarantees = (inf, inf, inf, inf, 50e6, 10e6, 500e3, 150e3)
    topo.update_guarantees_all_links(per_hop_guarantees)

    # Streams
    streams = []

    stream2 = Stream(label="s1",
                     path=topo.shortest_path(talkers[0], listeners[0]),
                     priority=7,
                     rate=220e3,  # in bits / s
                     burst=1020 * 8,  # bits
                     minFrameSize=64 * 8,  # bits
                     maxFrameSize=1000 * 8)  # bits
    streams.append([stream2])

    # TODO: temporary
    topo.add_stream(stream2)
    for link, tup in apply_model_to_topology(topo, "sp_simple").items():
        print(f"{link.name}: \t  {tup}")

    stream1 = Stream(label="s0",
                     path=topo.shortest_path(talkers[0], listeners[0]),
                     priority=7,
                     rate=10e6,  # in bits / s
                     burst=250 * 8,  # bits
                     minFrameSize=100 * 8,  # bits
                     maxFrameSize=230 * 8)  # bits
    streams.append([stream1])

    # Dirty hack for multicast streams
    stream1_2 = same_stream_different_listener(topo, stream1, listeners[1])
    streams[-1].append(stream1_2)

    # TODO: temporary
    print("-")
    topo.add_stream(stream1)
    topo.add_stream(stream1_2)
    for link, tup in apply_model_to_topology(topo, "sp_simple").items():
        print(f"{link.name}: \t  {tup}")
```

Stream 1:

listener2-sw2:	(inf, inf, inf, inf, 12336, 12336, 12336, 12336)
talker1-sw1:	(inf, inf, inf, inf, 31527, 22727, 20637, 20527)
sw1-talker2:	(inf, inf, inf, inf, 12336, 12336, 12336, 12336)
sw1-sw2:	(inf, inf, inf, inf, 31558, 22758, 20668, 20558)
talker2-sw1:	(inf, inf, inf, inf, 12336, 12336, 12336, 12336)
sw2-listener1:	(inf, inf, inf, inf, 31589, 22789, 20699, 20589)
listener1-sw2:	(inf, inf, inf, inf, 12336, 12336, 12336, 12336)
sw2-listener2:	(inf, inf, inf, inf, 12336, 12336, 12336, 12336)
sw2-sw1:	(inf, inf, inf, inf, 12336, 12336, 12336, 12336)
sw1-talker1:	(inf, inf, inf, inf, 12336, 12336, 12336, 12336)

-

Stream 2:

listener2-sw2:	(inf, inf, inf, inf, 12336, 12336, 12336, 12336)
talker1-sw1:	(inf, inf, inf, inf, 534927, 126127, 29037, 23927)
sw1-talker2:	(inf, inf, inf, inf, 12336, 12336, 12336, 12336)
sw1-sw2:	(inf, inf, inf, inf, 536350, 127550, 30460, 25350)
talker2-sw1:	(inf, inf, inf, inf, 12336, 12336, 12336, 12336)
sw2-listener1:	(inf, inf, inf, inf, 537773, 128973, 31883, 26773)
listener1-sw2:	(inf, inf, inf, inf, 12336, 12336, 12336, 12336)
sw2-listener2:	(inf, inf, inf, inf, 518520, 118520, 23520, 18520)
sw2-sw1:	(inf, inf, inf, inf, 12336, 12336, 12336, 12336)
sw1-talker1:	(inf, inf, inf, inf, 12336, 12336, 12336, 12336)

THANK YOU!

Questions, comments, suggestions?

Alexej Grigorjew

University of Wuerzburg

Chair of Communication Networks

Email: alexej.grigorjew@uni-wuerzburg.de