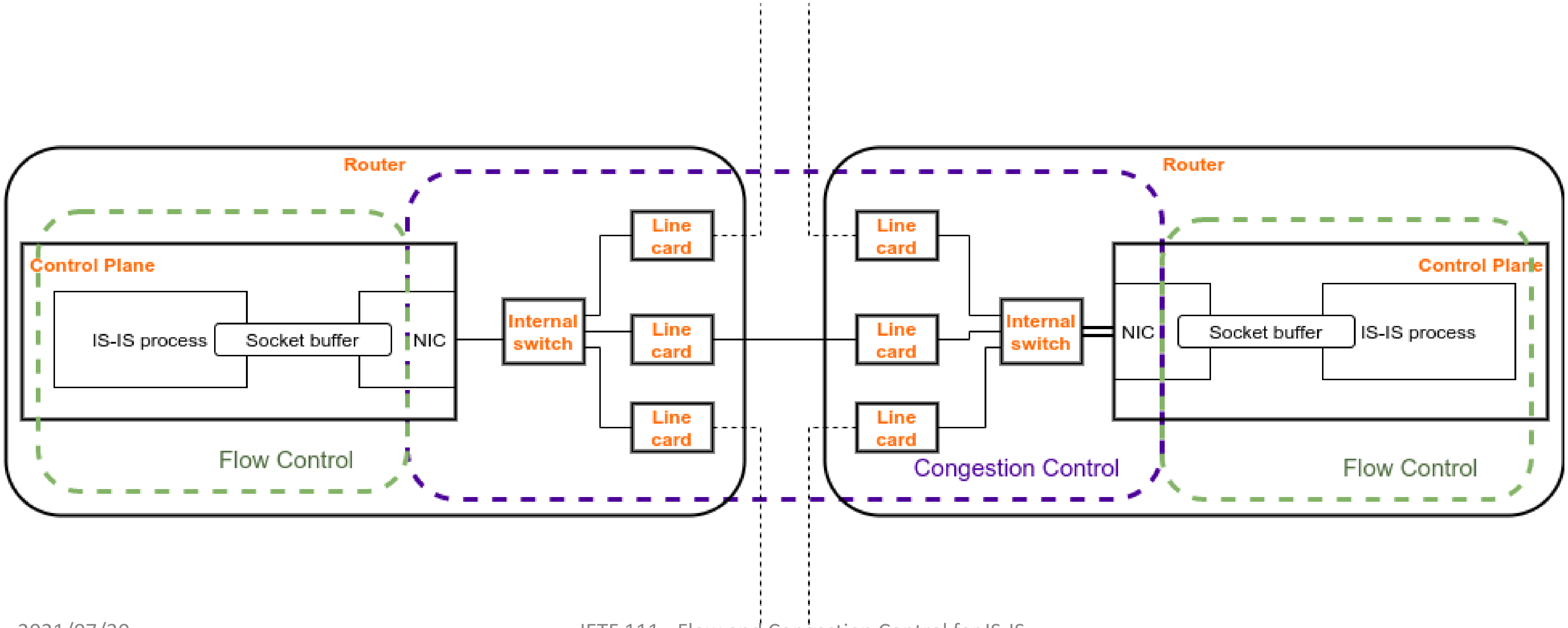
Flow and Congestion Control for IS-IS

Guillaume Solignac – Bruno Decraene

draft-decraene-lsr-isis-flooding-speed

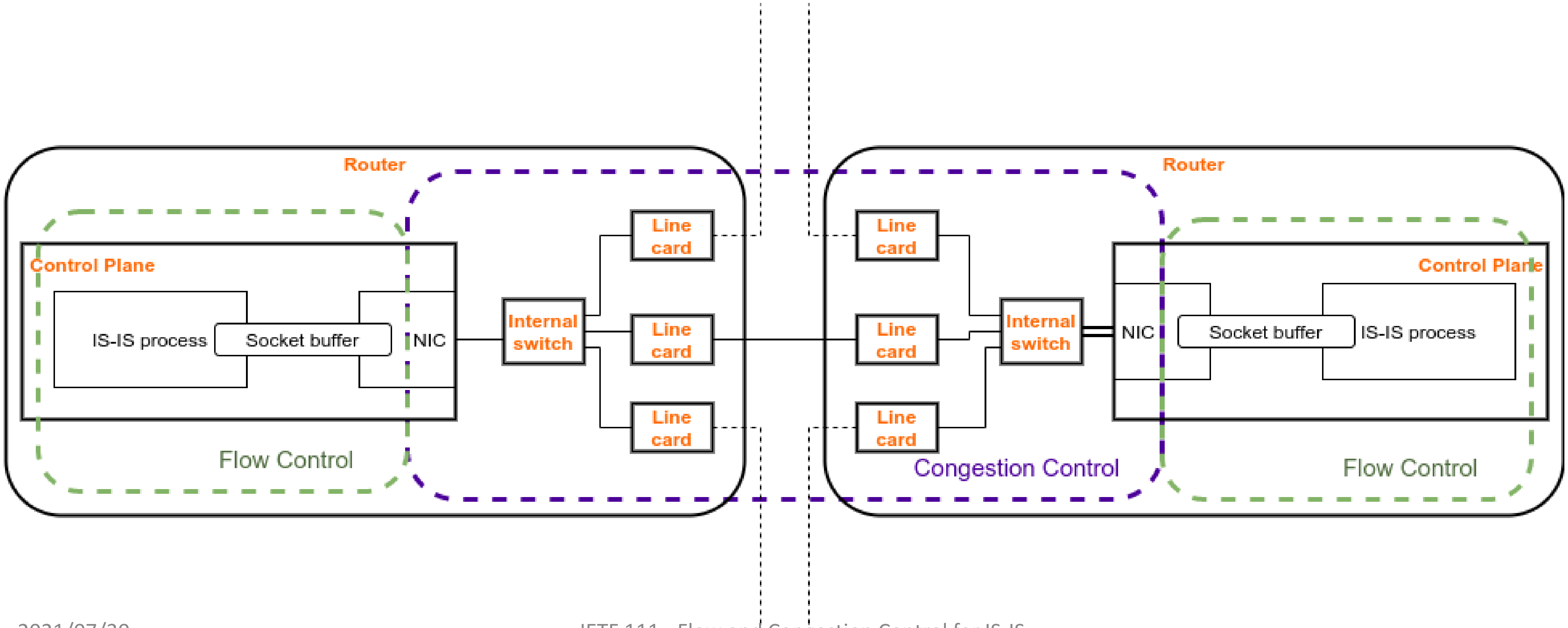
2-problem space

- CPU speed & socket buffer
- IO path



2 solutions

- Flow Control – Receive Window (RWin) for CPU "congestion"
- Congestion Control algorithm for IO congestion



Goals

- ❑ Faster flooding when the receiver has free cycles
- ❑ Slower flooding when the receiver is busy/congested
- ❑ Avoiding/minimizing the parameters the network operator has to tune
- ❑ Avoiding/minimizing the loss of LSPs
- ❑ Robust to a wide variety of conditions (Good & bad ones)
- ❑ Simplicity of implementation

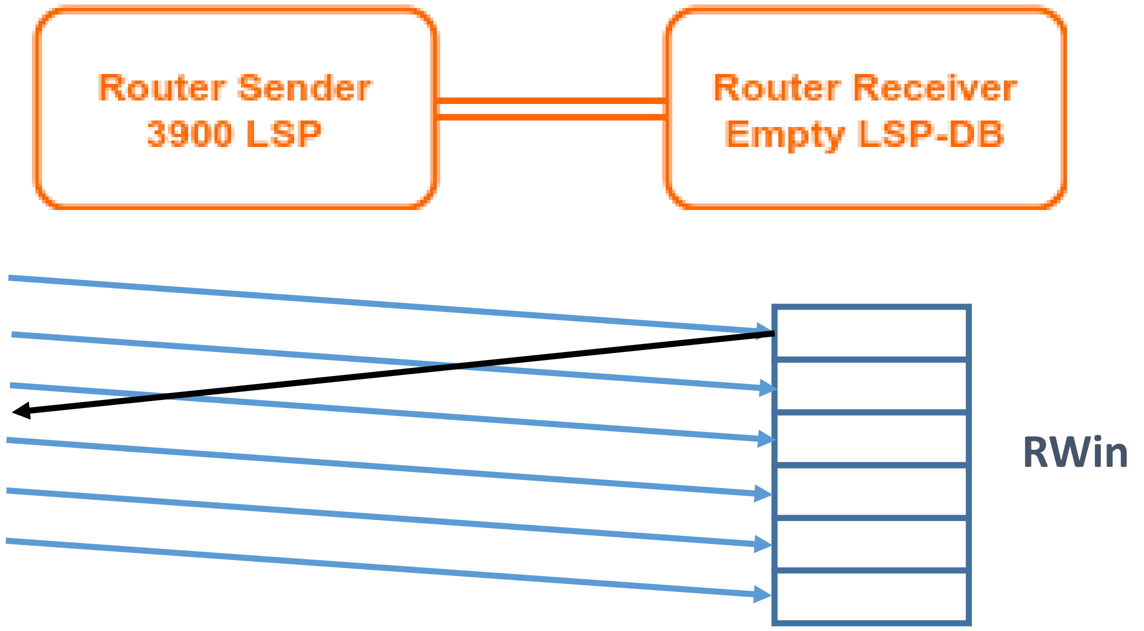
1. Flow control with RWin

Flow Control

- Goal : Avoid loosing LSPs at the receiving side. Mostly for p2p interface.
- Classic algorithm : Receiver window (RWin). Sender will never send more than RWin unacknowledged LSPs.
- When receiving LSP or PSNP:
  - Send_n_LSP(RWin – Unacked_lsps)

➢ Advertise Receive Window in Hello PDU

➢ Change PSNP behaviour to get faster feedback (otherwise only every n seconds)



How to choose RWin ?

By order of preference :

1. Socket size / 2
2. Use TCP value (used by BGP)
3. Conservative value (10) – Existing default in a popular implementation

<u>Software</u> parameter, hardware-independent.

When to send a PSNP ?

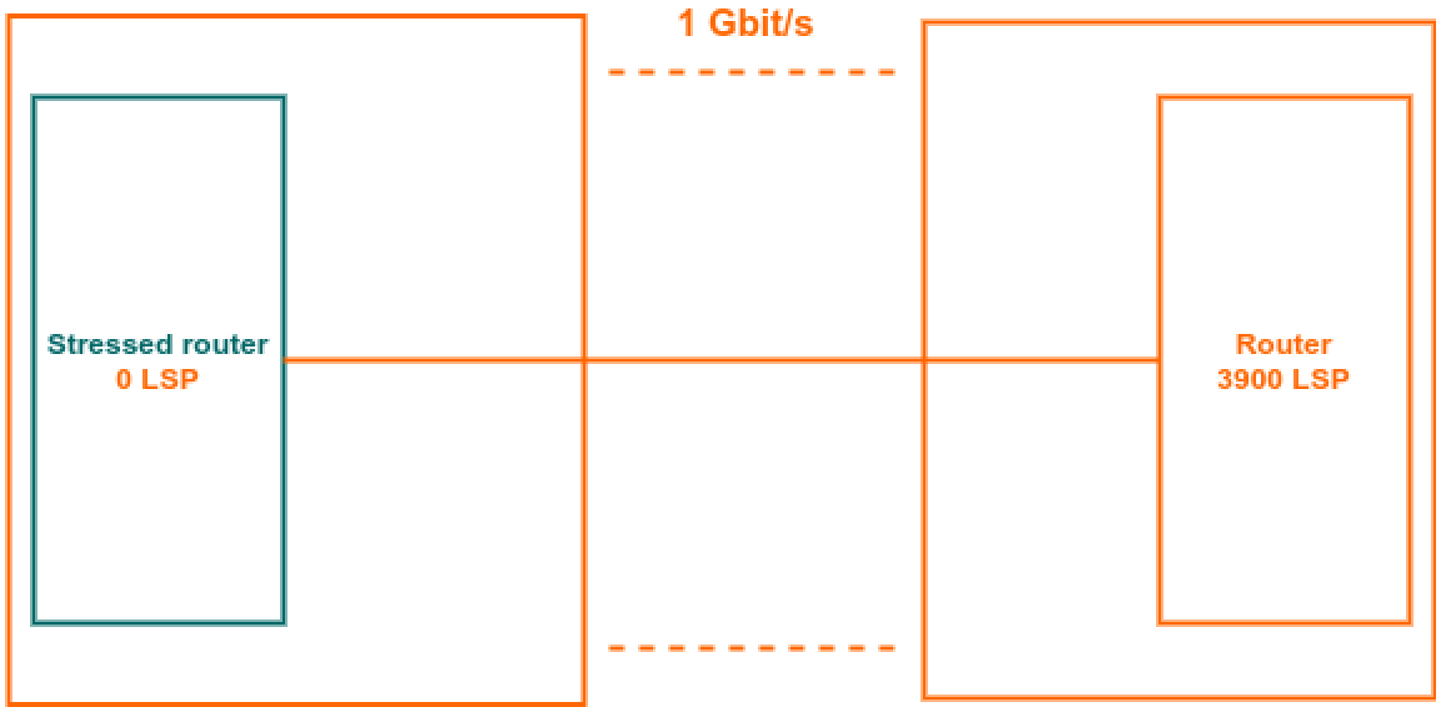
- Every **LPP** LSPs received. LPP = LSPs per PSNP
- Timeout

Flow Control

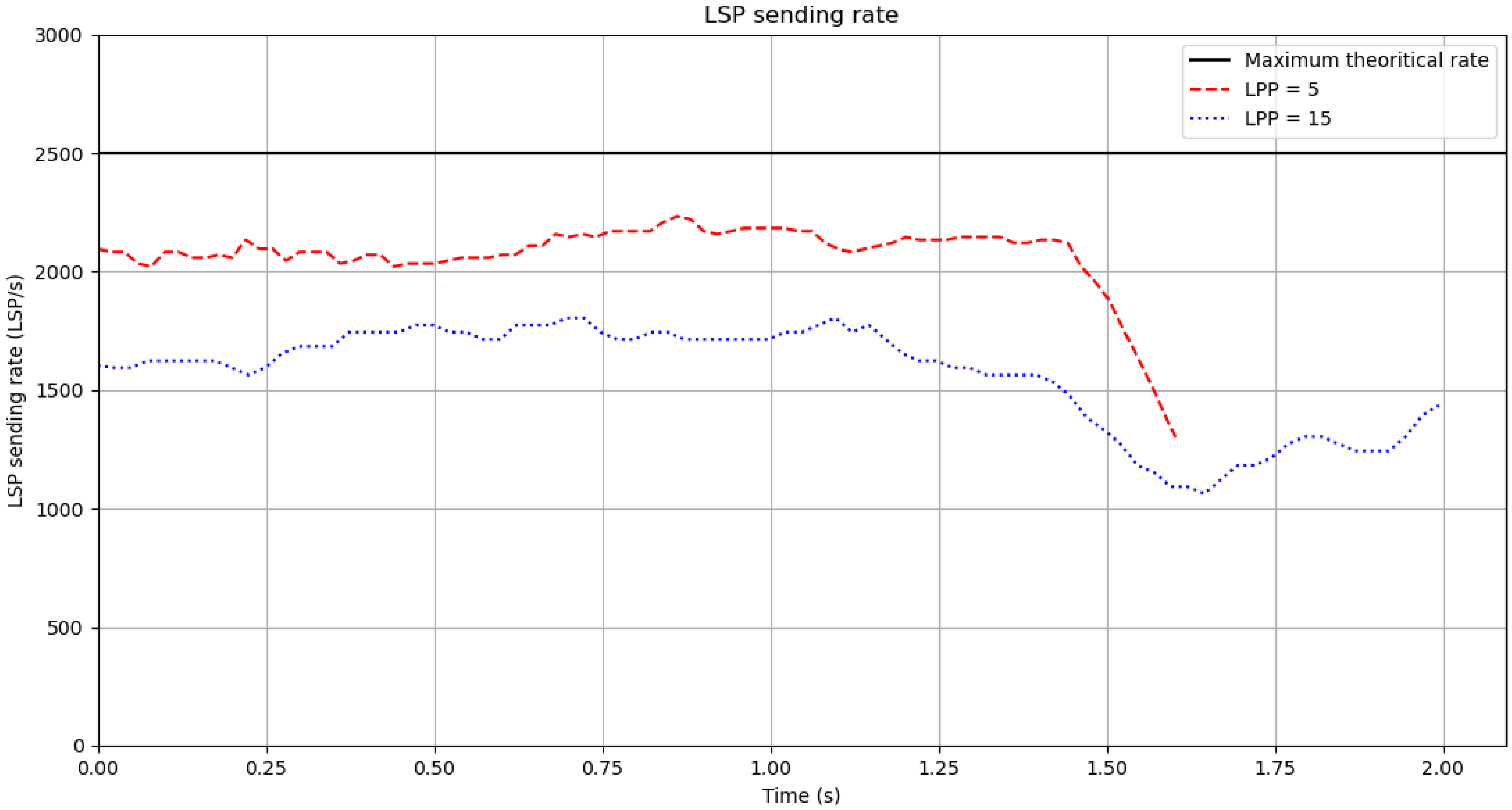
- Relies on one static information : size of socket buffer
- Multiple identified parameters influence behavior :
  - RWin
  - RTT
  - Number of LSP per PSNP (LPP)

$$rate < \frac{\mathrm{RWin}}{RTT} = \text{Theoritical rate}$$

Experimental setup – 1vs1

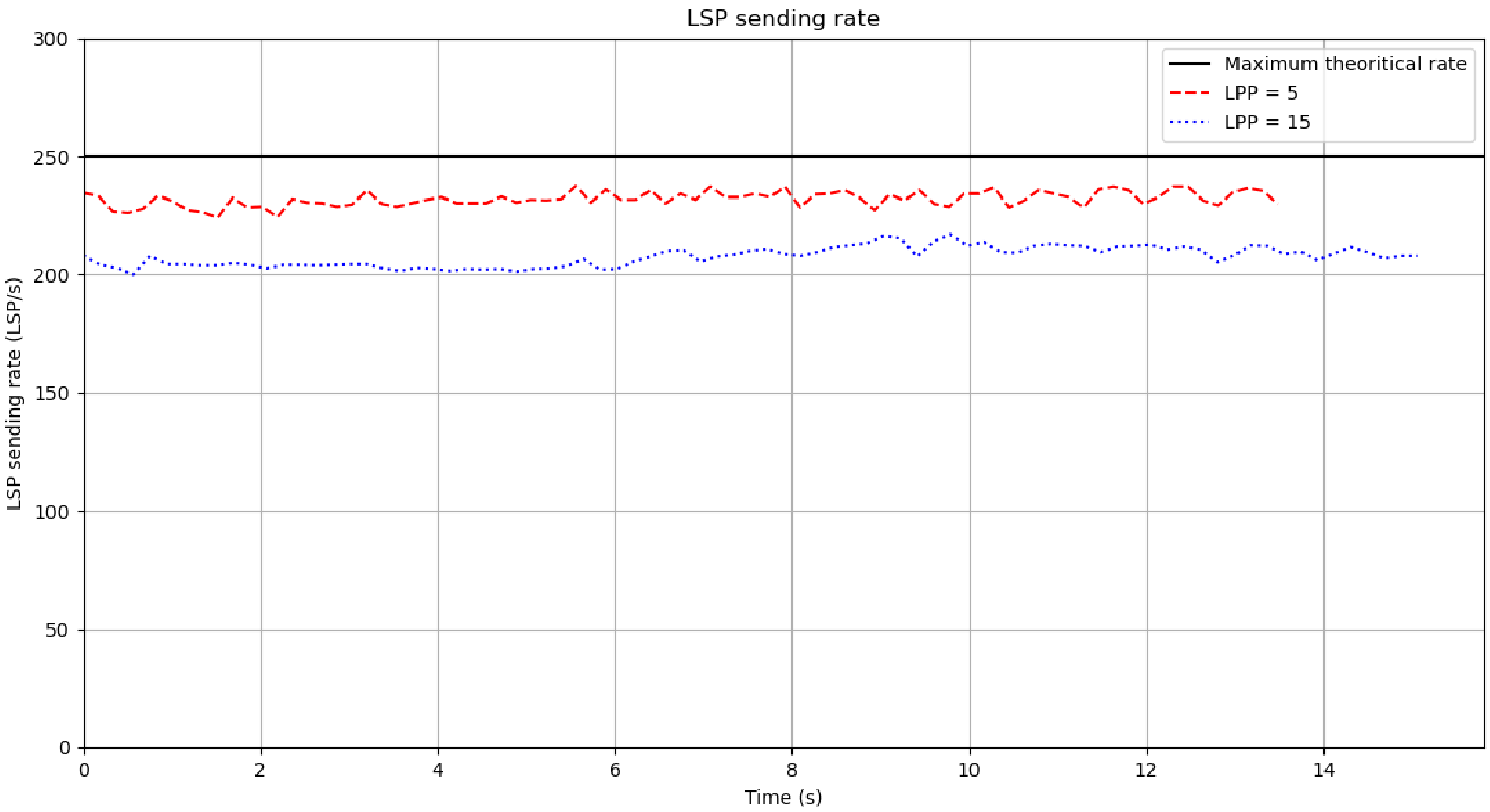


Flow Control – Results 1vs1 – RTT 20ms – RWin 50



- Close but lower than theoretical rate (50/20ms = 2500 LSP/s), more on that later
- No LSP loss

Flow Control – Results 1vs1 – RTT 200ms – RWin 50

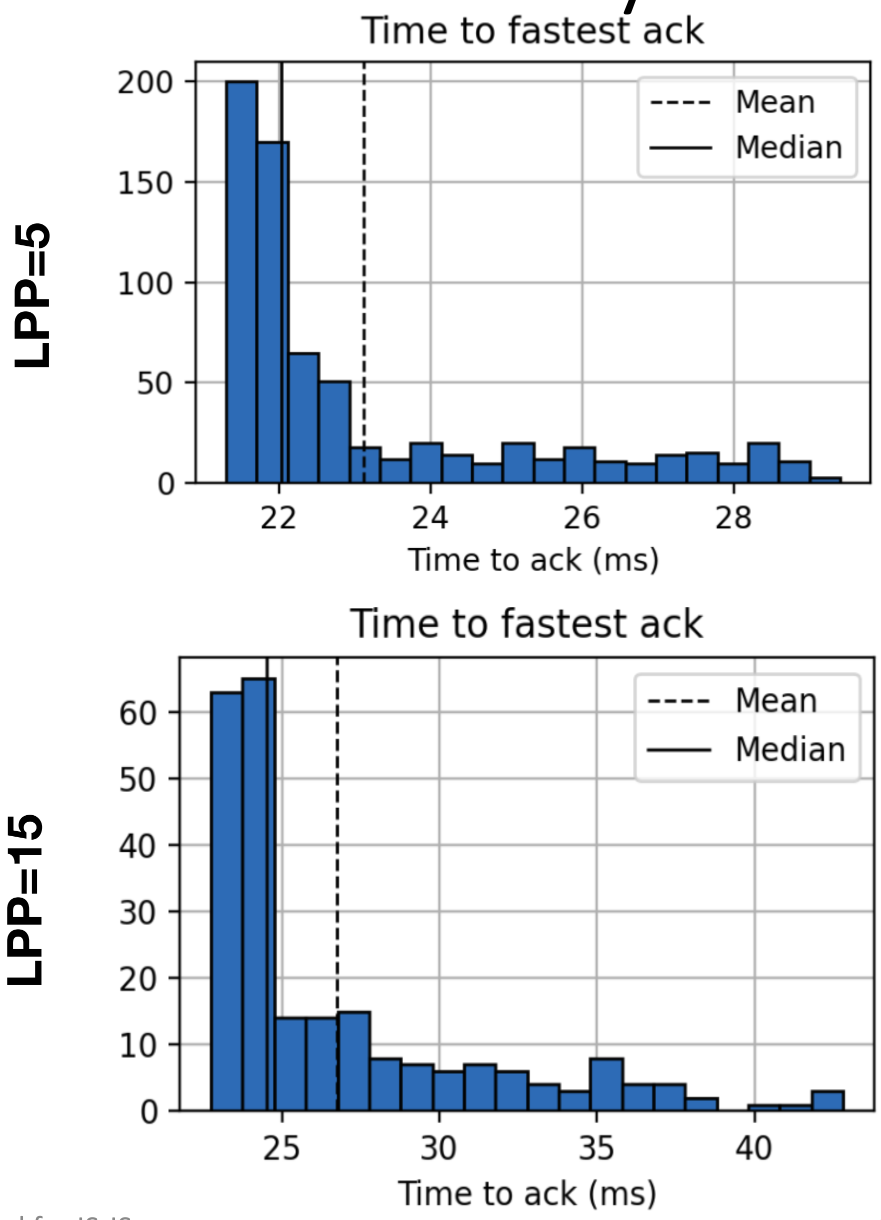


- Close to theoretical rate (50/200ms = 250 LSP/s)
- No LSP loss

Flow Control – Results 1vs1 – PSNP delay

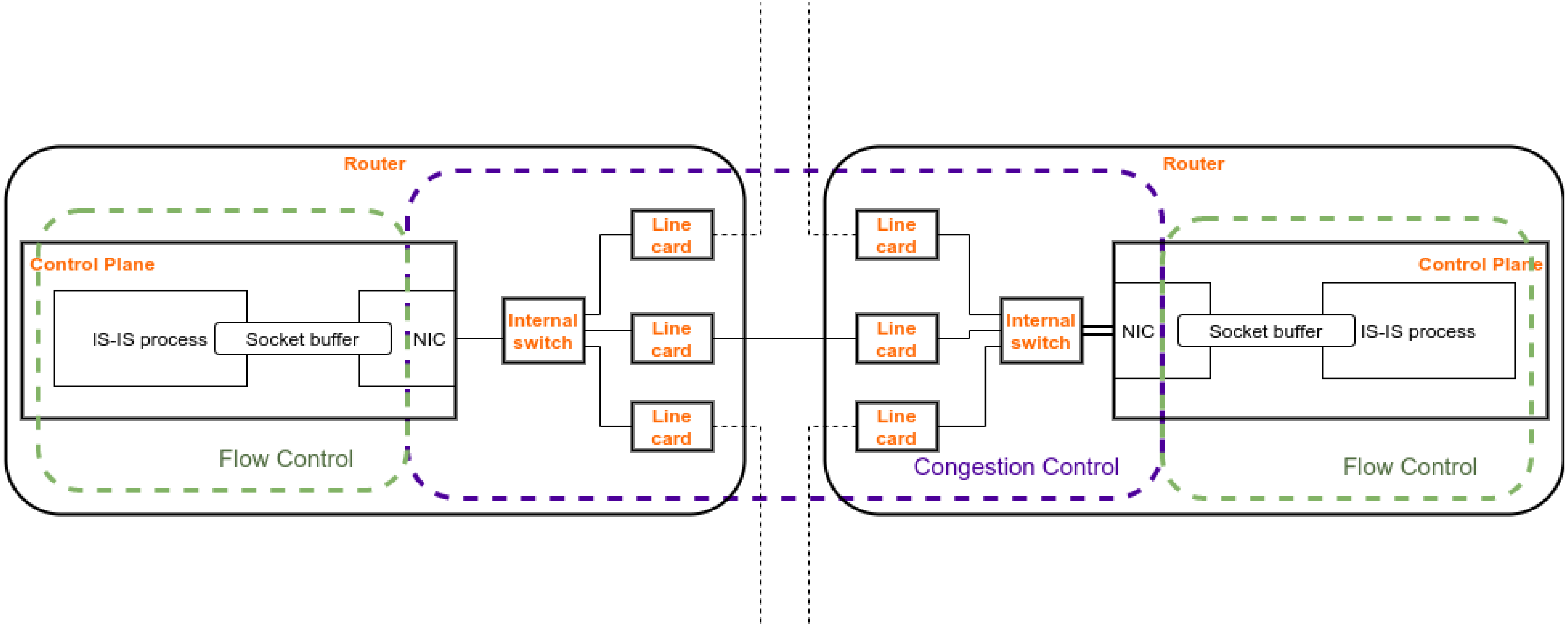
- Metric : inside a PSNP, time to latest included LSP

- Effective RTT is higher than 20ms due to computation time of LSPs & PSNP crafting
- RWin should be a multiple of LPP
    - Avoids delaying unfilled PSNP

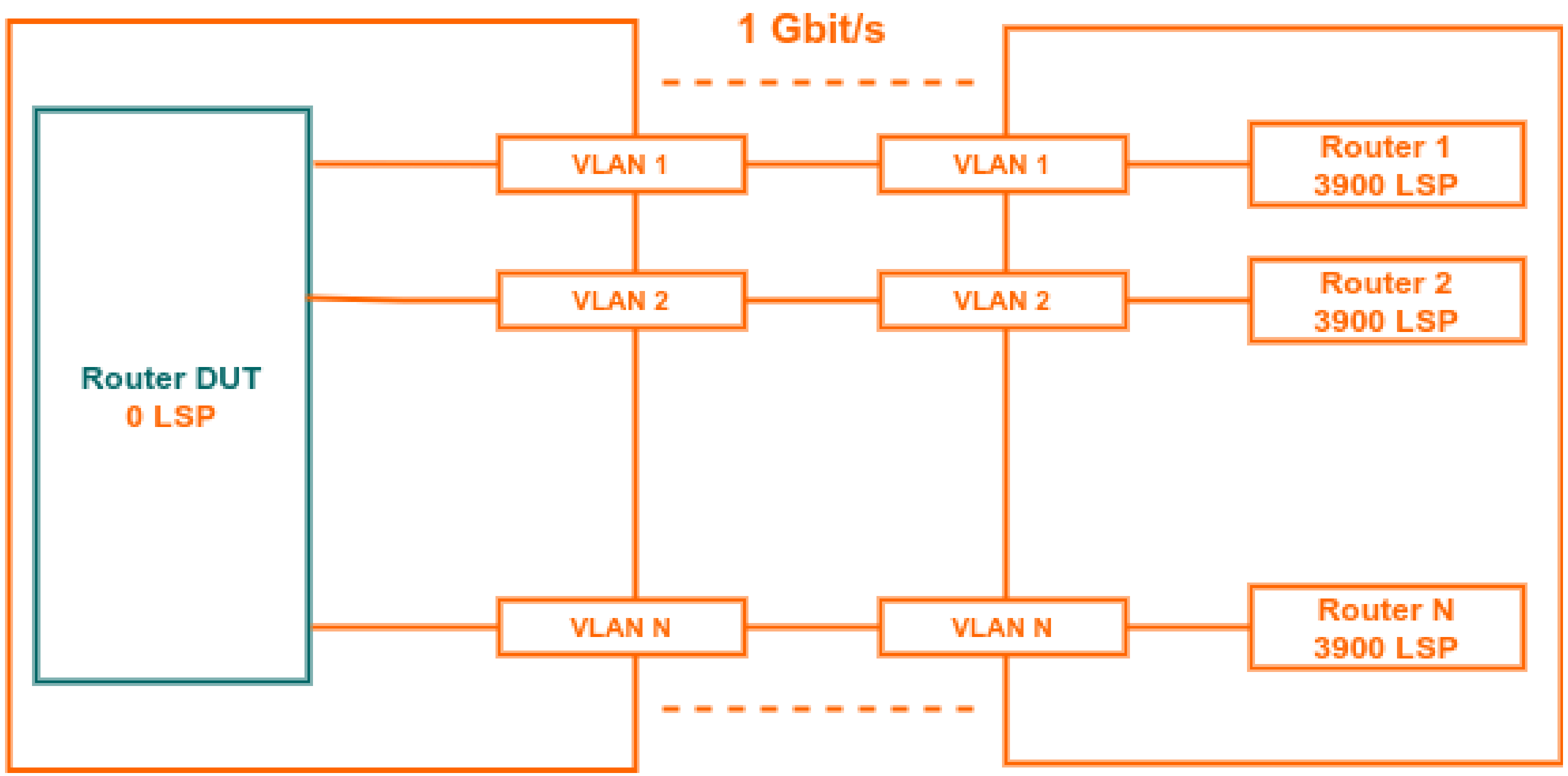


Increased sending rate hazard

Increased sending rate means more stress, thus potentially losses in IO paths.

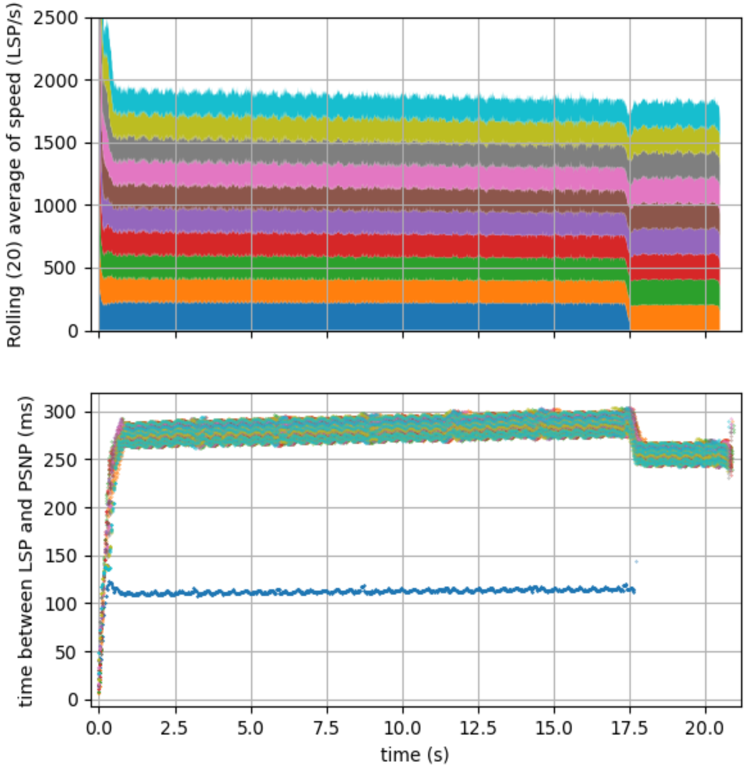


Experimental setup – Nvs1

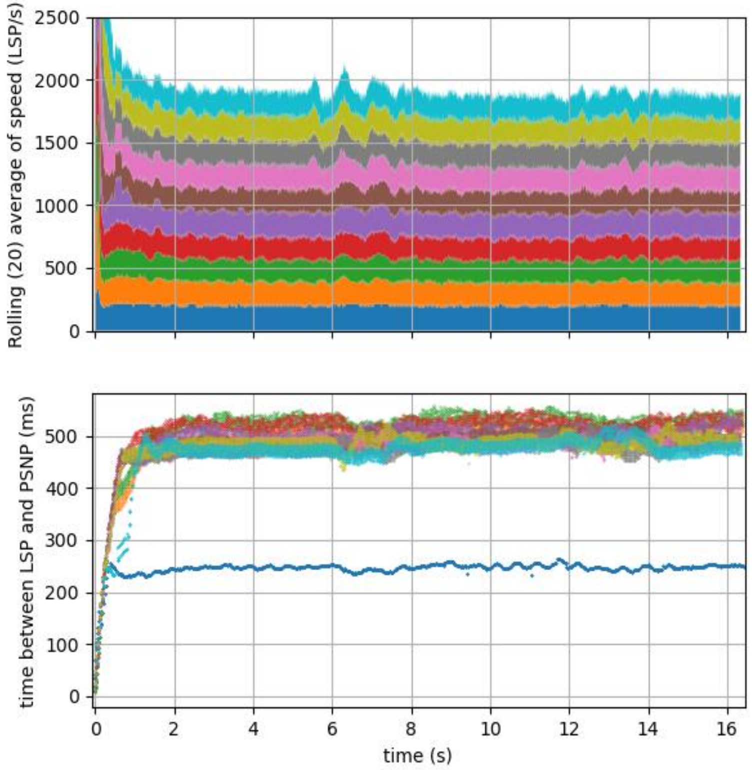


Flow Control – Results 10vs1– LPP = 1 – RTT = 1ms

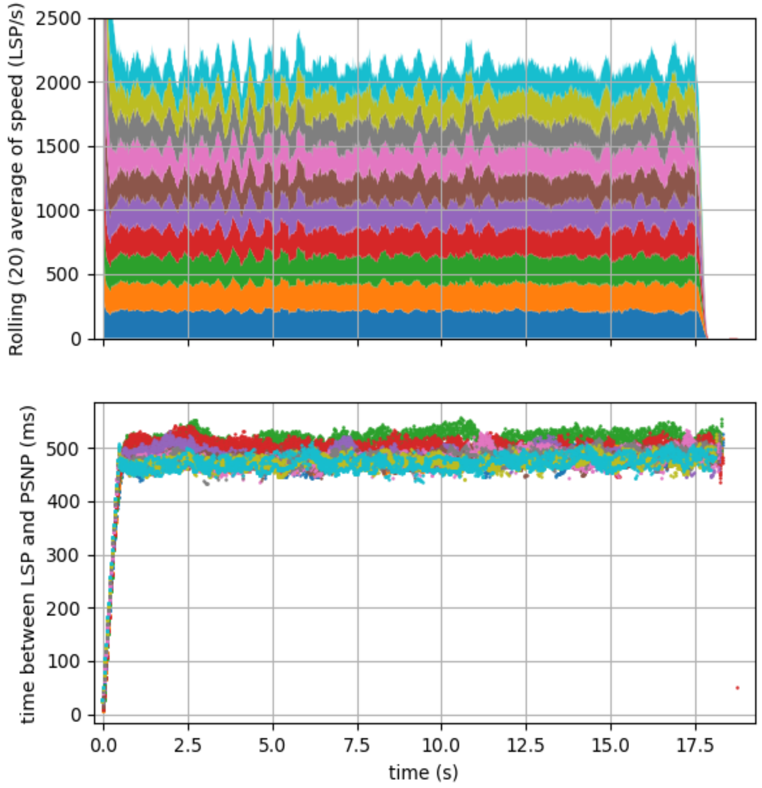
RWin=25



RWin=50



RWin=100

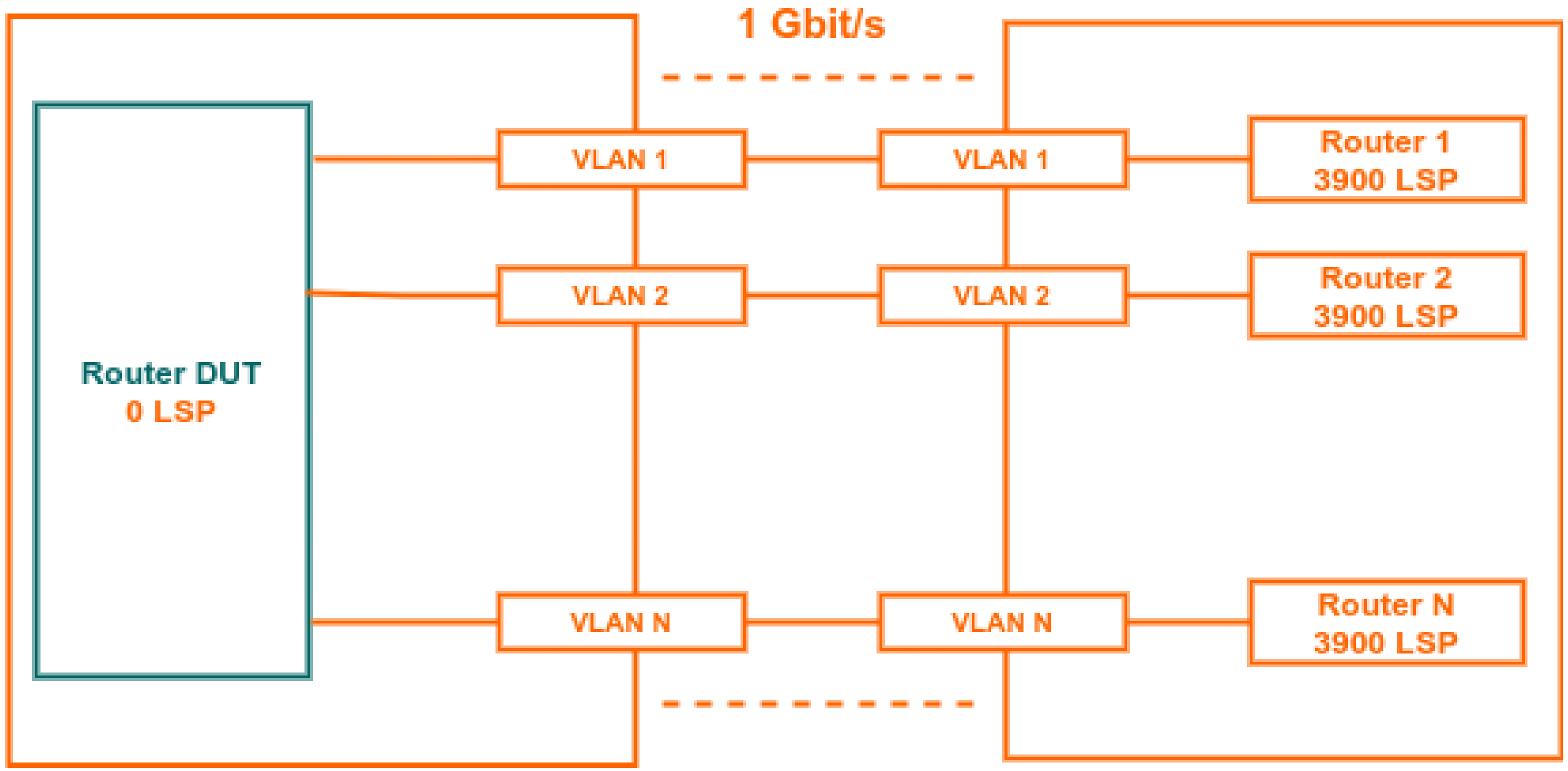


- Good point : no LSP loss, senders pace well $\frac{200 \text{ LSP/s}}{30 \text{ LSP/s}}$ = 6.7 speedup compared to default rate
- CPU Bound. Increasing Rwin only increases latency.

$$\max\_rate = \frac{\text{RWin}}{RTT}$$

2. RWin with IO bottleneck

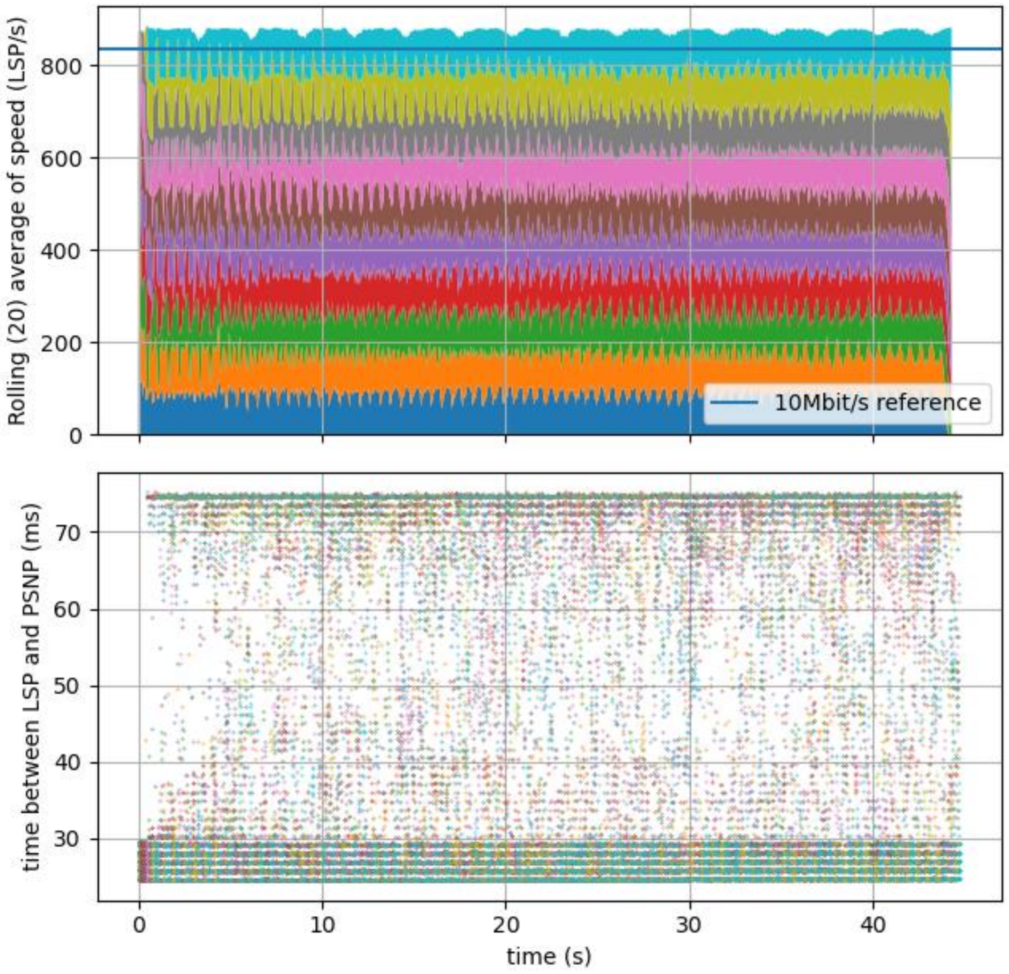
Experimental setup – Nvs1



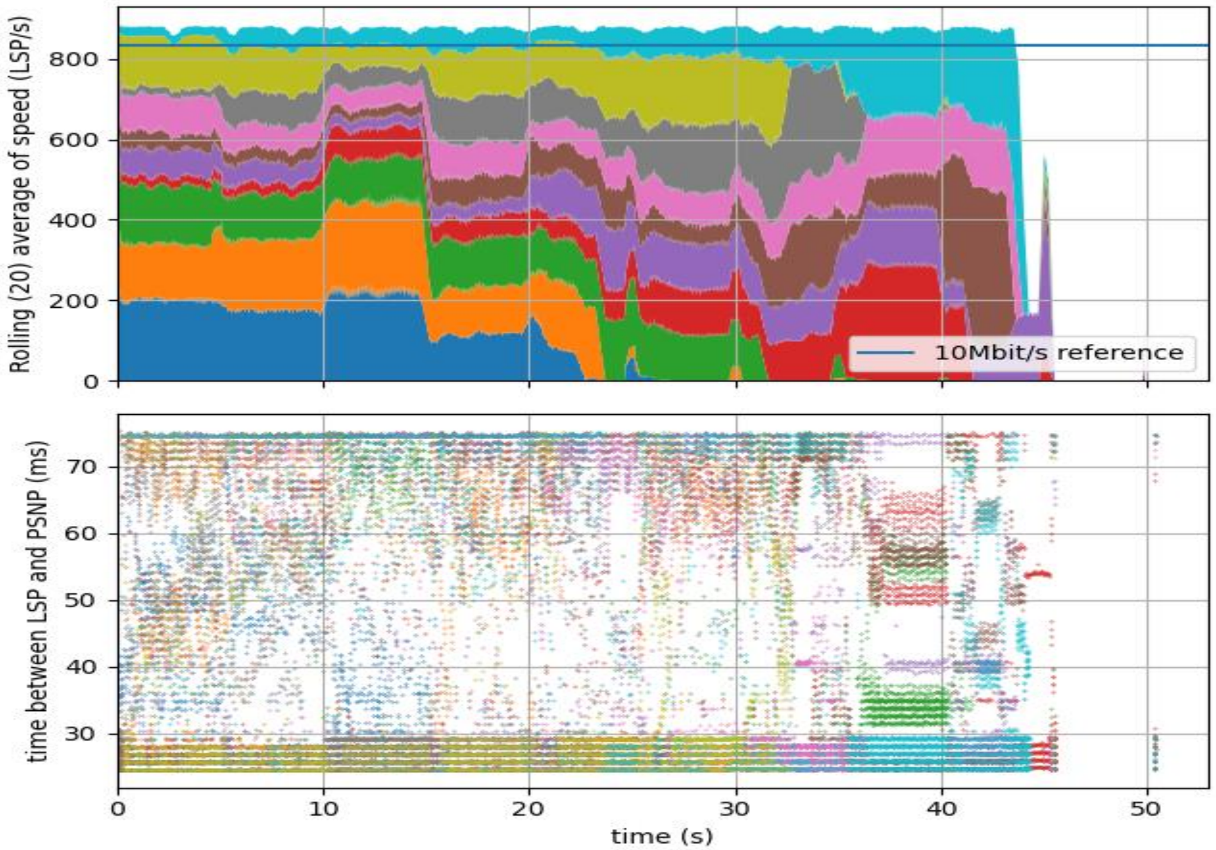
Flow Control – Results 10vs1 – LPP = 5 – RTT = 25ms
Bottleneck = 10Mbit/s, buffer = 2600 packets

- Good point : no LSP loss because :

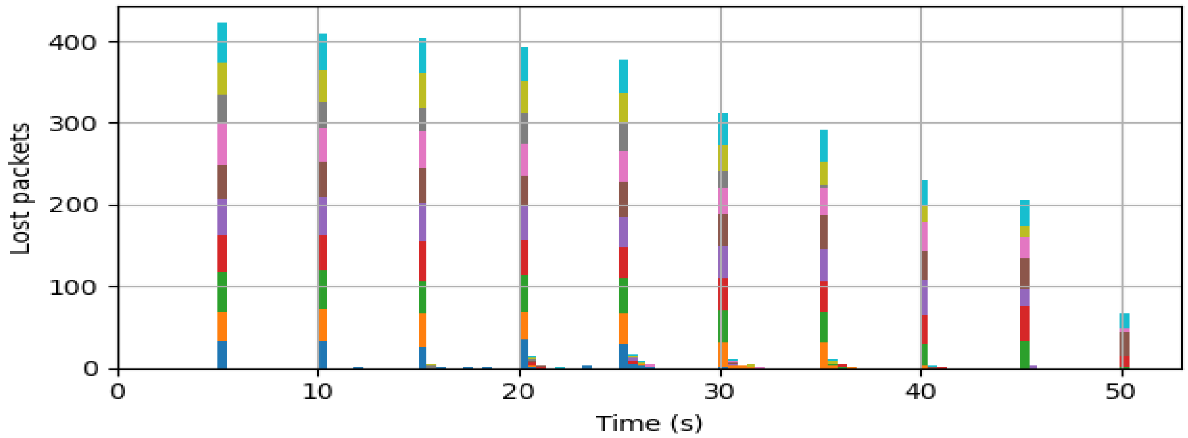
$$nb\_senders * RWin = 10 \; * 50$$
$$< 2600 = \; bottleneck\_buffer$$



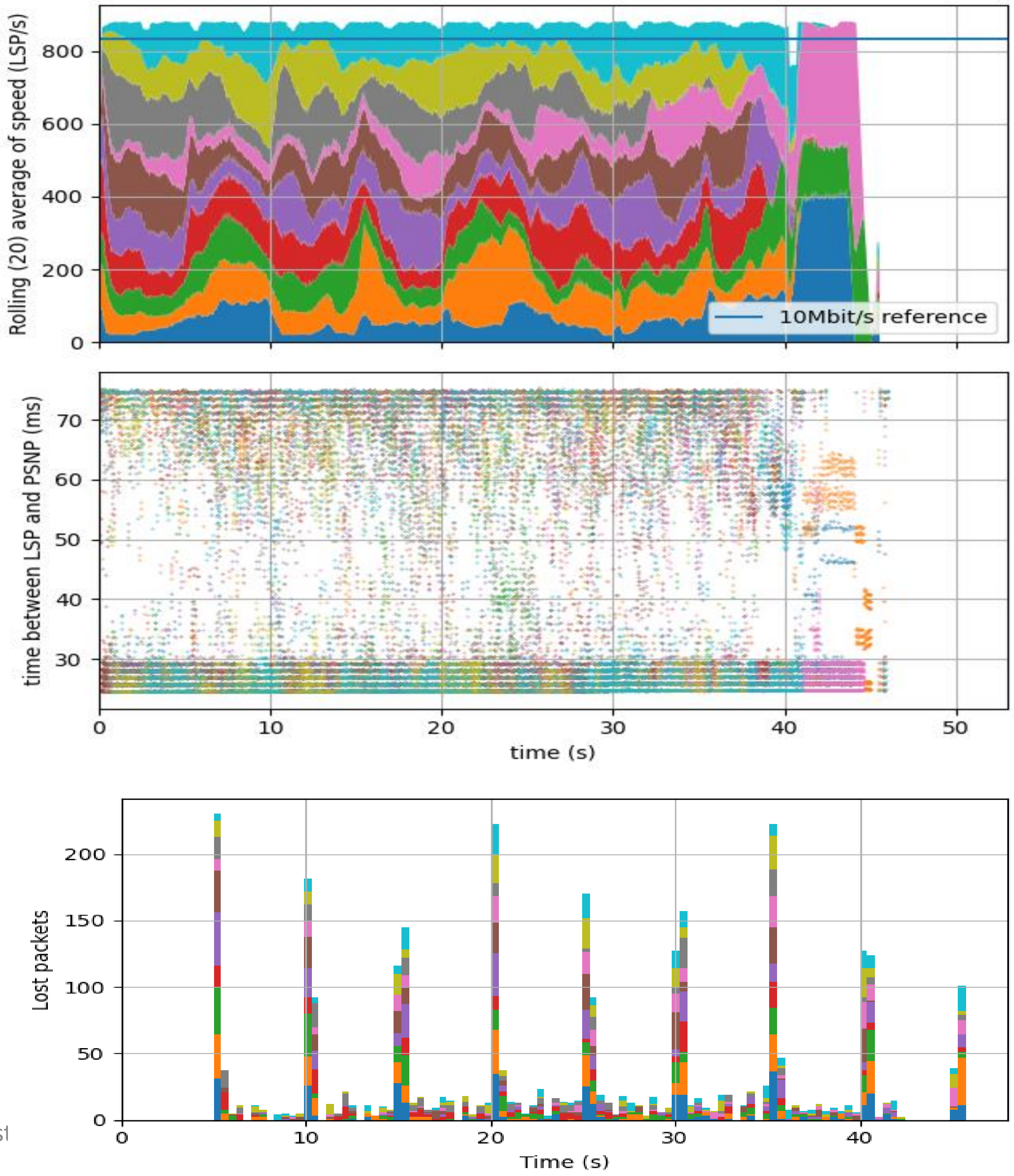
# Flow Control – Results 10vs1 – LPP = 5 – RTT = 25ms
Bottleneck = 10Mbit/s
buffer = 64 packets



- 3218 losses (8.2% of 39000 LSPs)
- Losses are bursty : corresponds to the LSP timeout
- At every burst, approx. 500 – 64 = 473 LSPs lost.



Flow Control – Results 10vs1 Multiple senders – LPP = 5 – RTT = 25ms Bottleneck = 10Mbit/s, buffer = 64 packets, slow-start



- 3023 losses (7.7% of 39000 LSPs)
- Slow start helps slightly mitigating the burstiness

RWin algorithm behavior

- 3 sending rates possible :
  - RWin limit (RWin /RTT)
  - CPU limit
  - IO limit

- Effective rate = min(RWin/RTT, CPU, IO)
- First two cause no LSP loss – and we think CPU will be the bottleneck in most (if not every) implementations today
- Only IO bottleneck is addressed by congestion control

Flow Control – Results 10vs1 Multiple senders

Recap :

- Cost:
  - New TLV
  - Socket buffer
  - More PSNPs (with LPP=15, 6 times more PSNPs)
- Gain:
  - No LSP loss due to socket buffer exhaustion
  - Speed paced by receiver ACKs → “CPU congestion” is dealt with
  - Dropped LSPs artificially fills RWin → “Internal congestion” causes speed to drop –which is good-.

3. Congestion Control with CWin

Congestion Control algorithms

Extensively studied in the case of TCP

3 steps : slow-start, end of slow-start, congestion avoidance

Various approaches : losses, delay, bandwidth

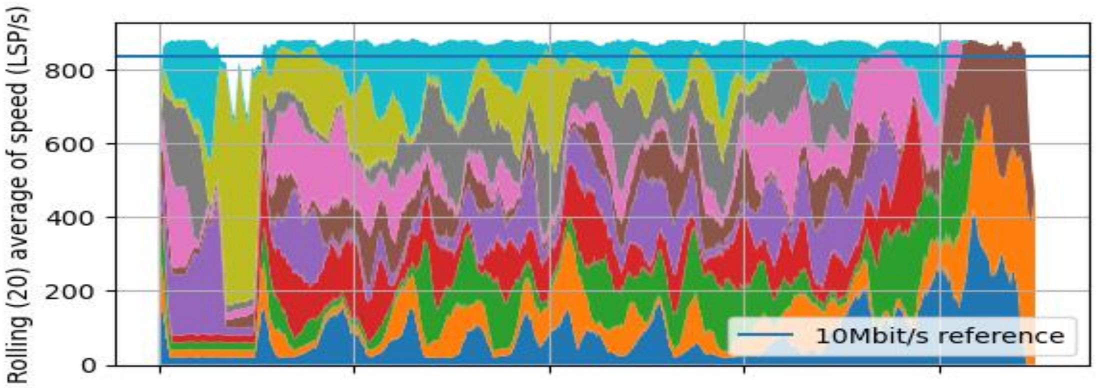
Losses : for TCP, packet reordering (not available here) & timeouts
Delay : Try to detect queuing delay, not necessarily good here because IS-IS processing time will be the bottleneck in many cases → not a general solution
Bandwidth : interesting but needs enough data to stress the bottleneck. Unfavorable case for IS-IS as it depends on the neighbors.

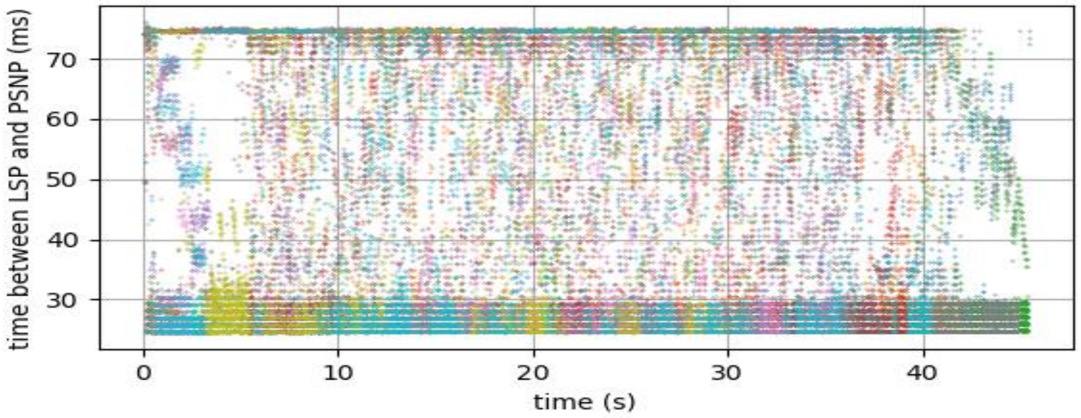
Overall, the more reactive, the more cycles/memory is needed

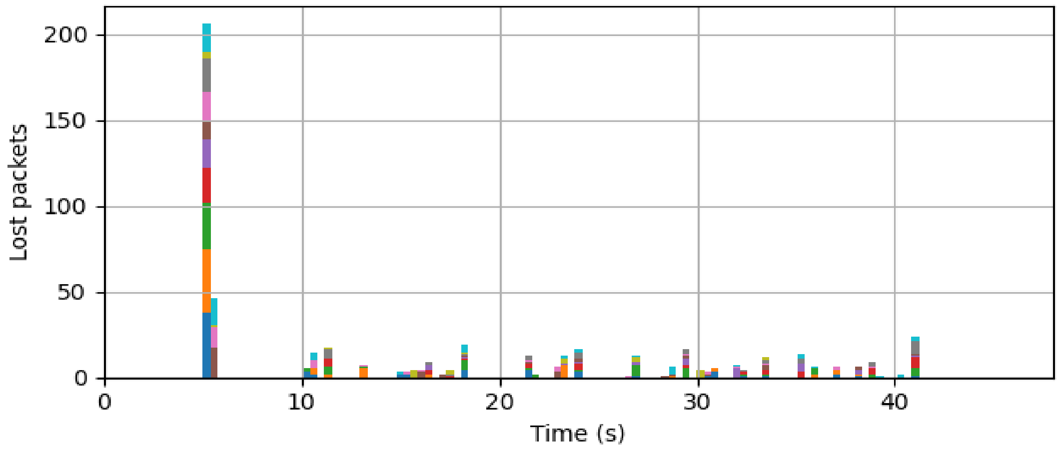
The algorithm tries to deal with buffers. But for internal IO-path, might be very small buffers → hard to deal with in any case.

Flow Control – Results 10vs1 Multiple senders – LPP = 5 – RTT = 25ms Bottleneck = 10Mbit/s, buffer = 64 packets, congestion avoidance

- 524 lost LSPs (1.3% of 39000 LSPs)
- Congestion control helps a lot in avoiding losses
- Large overshoot at the end of the slow-start (252 losses on first round only)
- Slow-start could be removed but helps scaling to larger links (otherwise rate of growth is slow)







Why not only CWin/Congestion Control ?

- Implementations are likely to be CPU bound : RWin is perfect for this case.
- CPU can be busy doing something else than processing LSPs : RWin will naturally pause the sending, thus avoiding losses that a CC algorithm alone will take some time to detect.
  - e.g BGP, SPF + TI-LFA + µ-loop, C-SPF TE
- Congestion control will have to loose packet to detect CPU slowness. It will detect CPU busyness as congestion while it really **is not**.
- They can work together !

Recap

Scenario	RTT	RWin	LPP	IO rate (LSP/s)	IO buffer	Bottleneck	Achieved rate (LSP/s)	% lost LSPs
1v1	20ms	50	5	80k	Inf	RWin	~2500	0
1v1	200ms	50	5	80k	Inf	RWin	~250	0
10v1	1ms	25	1	80k	Inf	CPU	2000	0
10v1	1ms	50	1	80k	Inf	CPU	2000	0
10v1	1ms	100	1	80k	Inf	CPU	2000	0
10v1 – RWin	25ms	50	5	833	2600	IO	860	0
10v1 – RWin	25ms	50	5	833	64	IO	860	8.2%
10v1 RWin + Slow-start	25ms	50	5	833	64	IO	860	7.7%
10v1 CWin	25ms	50	5	833	64	IO	860	1.3%

Recap

- ✓Faster flooding when the receiver has free cycles
- ✓Slower flooding when the receiver is busy/congested
- ✓Avoiding/minimizing the parameters the network operator has to tune
- ✓Avoiding/minimizing the loss of LSPs
- ✓Robust to a wide variety of conditions (Good & bad ones)
- ✓Simplicity of implementation

Recap

	RWin only	**CWin only**
CPU congestion on Receiver CP	Ok : RWin bounded and lower than socket size	Partial : CPU availability can change very fast
IO Bottleneck	Partial : losses bounded by sum of advertised RWins; lost packets trash RWin, inducing speed decrease	Ok (with hypothesis)
CPU resources	Low cost	Increases with algorithm complexity
Memory resources	Known buffer size for RWin	Increases with needed state

Thank you

Appendix

Default optimized Fast-flood parameters

Default Optimize Values for IS-IS

The following table summarizes the configuration impacted by default optimize:

IS-IS command	Parameters	Default optimize disabled	Default optimize enabled
fast-flood			
	# of lsps flooded back-back	Disabled	10

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/seg_routing/configuration/xe-16-8/segrt-xe-16-8-book/sr-fast-convergence-default-optimize.pdf

Flow Control – Results 10vs1 Multiple senders

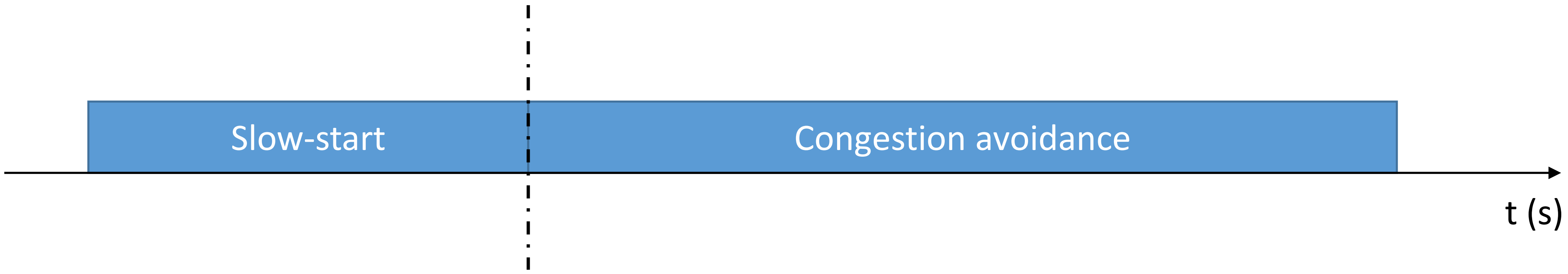
- LSP retransmitted
- per VLAN :

RWin	Retransmitted LSP Sender -> Receiver	Retransmitted LSP Receiver -> Sender
25	0	0
50	0	141
100	30	2343

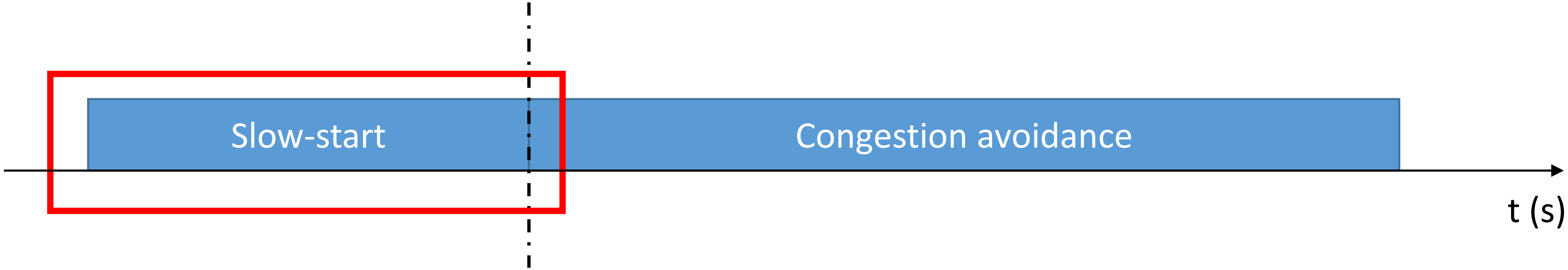
- Socket size : 212992 bytes
- Overhead per packet : 576 bytes (sk_buff, skb_shared_info)
- PDU size ~ MTU = 1500 bytes
- LSPs/socket : $\frac{212992}{1500+576} = 102 \text{ LSP}$ -> not much room for PSNP and Hello !

- → Important to advertise a correct RWin to avoid overflooding the socket buffer

Congestion Control – What is it ?

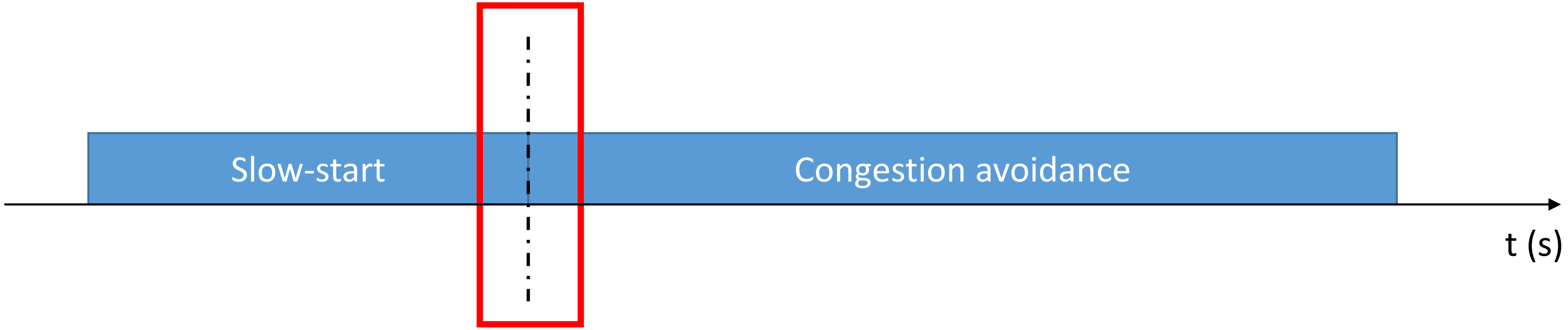


Congestion Control – What is it ?



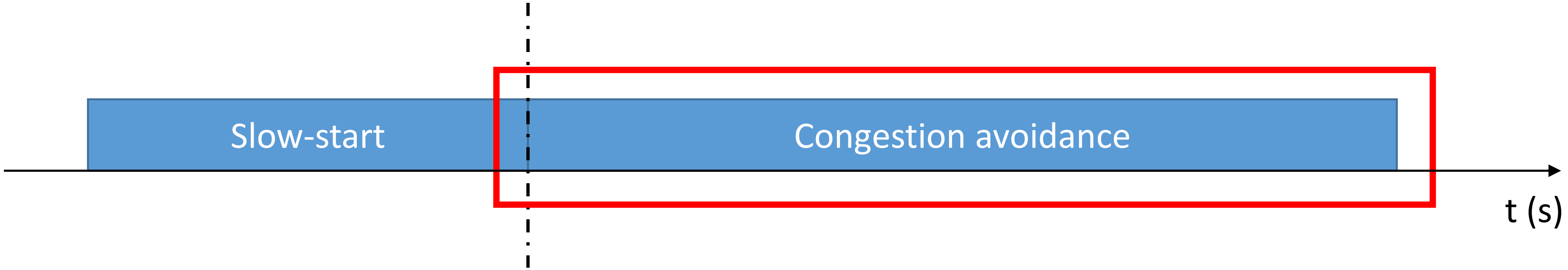
Exponential growth

Congestion Control – What is it ?



Packet loss
RTT measurements
Bandwidth

Congestion Control – What is it ?



Additive increase multiplicative decrease (AIMD)
Bandwidth target & control loop