

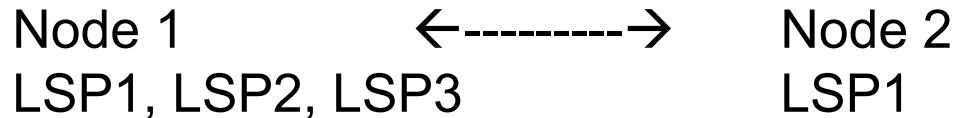
IS-IS Flooding Speed advertisement

draft-decraene-lsr-isis-flooding-speed-03

<u>Bruno Decraene</u>	(Orange)
Chris Bowers	(Juniper)
Jayesh J	(Juniper)
Tony Li	(Arista)
Gunter Van de Velde	(Nokia)

Problem statement

Distributed SPF requires that all nodes have the same LSDB.



Flooding is done between two adjacent nodes.

Need to sync LSDB between those *two neighbors* as fast as possible.

We seemed to have reached consensus on this.

(Previously: some discussion about flooding needed to be done "at the same rate" network wide/on all adjacencies. This is behind us.)

Two main uses cases

Node Failure:

- 2 to 50 LSPs to flood
- In less than 100ms for fast convergence (sub second)
 - Ideally “0” ms for 2 LSPs (fast removal of a PE’s loopback)

Partition repair:

- 1000 - 5000 LSPs to flood
- In a few seconds (1 – 10 seconds)

Lab testing existing behavior

- Idealistic test conditions
 - Major implementation
 - High end router
 - Same implementation on both the receiver and the sender: no interop/interwork/assumptions issues
 - IS-IS only (e.g. no BGP)
 - 0ms RTT link
 - Single IGP adjacency: receiver deals with a single sender

Some outcome of tests

- Default parameters

- Slow LSDB synchronization
 - as per user manual; below 500kbit/s)
- No specific issue

#LSP sent	4024
Duration	150,1919
#LSP/second	26,79239
avg inter-LSP delay (ms)	37,32403
#LSP lost	1

- Tuned parameters

- Faster LSDB synchronization (x10)
- Good but still lower than my 200€ smartphone

#LSP sent	4023
Duration	5,055838
#LSP/second	795,7138
avg LSP inter delay (ms)	1,256733
#LSP retransmitted	0

- Too aggressive parameters

- Receiver is overwhelmed, even in those idealistic conditions
 - Sender needs to send some LSP multiple times
 - Lower goodput, higher load on both nodes
- Three time slower, for a small change in parameters
- Lack of flow control

#LSP sent	6179
Duration (s)	15,11277
#LSP/second	408,8595
avg LSP inter delay (ms)	2,445828
#LSP retransmitted	2156

Transport layer tool box

- Flow control
- Congestion control
- Loss detection & recovery

Flow control

- Prevents the sender from overwhelming the receiver
 - avoid losses & retransmissions
- TCP uses a 'receive window' advertised from the receiver to the sender.
- Draft proposes the same mechanism
 - Unit in number of LSPs (rather than bytes)

Flow control – receive window

- May be static
 - Dynamic flow control achieved by acknowledging the reception of LSP as per today
 - Well known bandwidth delay limitation
 - Higher delay means lower throughput or larger window
 - Benefits in sending xSNP faster.
 - Which value to pick
 - Just like TCP? (use the same value)
 - Platform dependent value?
 - Platform independent value like today (worst case)?

Flow control – receive window

- May be dynamic based on load
 - Advertise a reasonable value at startup
 - Increase/decrease depending on receiver load
 - E.g. waiting for I/O: increase window
 - E.g. can't exhaust incoming queue: decrease window

Flow control – receive window

- May be dynamic based on monitoring of relevant (averaged) hardware resources
 - Buffer space (most likely on the forwarding engine)
 - IS-IS CPU

Congestion control

- Prevents the sender from overwhelming the network
 - P2P high speed link is not the issue
 - Forwarding resources within the router from the ingress link to the control plane
 - platform dependent source of congestion & packet loss

Congestion control

- Does not necessarily require standardization, hence none in current version of the draft.
- Next version could propose one based on existing AIMD algo (used in TCP, SCTP, some DCCP modes).
 - AIMD: Additive Increase/ Multiplicative Decrease
 - start: congestion window := receive window / 2
 - linear increase with proportional control
 - N LSP ack'ed → increase the congestion window by N
 - exponential reduction
 - LSP lost → congestion window divided by 2

LSP loss and retransmission

- Existing mechanism in IS-IS
- Faster loss detection would improve feedback loop delay
 - Currently $>$ minimumLSPTransmissionInterval (5s)
- Draft proposes that receiver advertise a smaller value
 - Hence commit in fast acknowledgement
 - Allowing faster detection of LSP loss

In summary

- TCP like algorithm
 - ‘Receive window’ for flow control
 - Small traffic if blocked
 - ‘Additive Increase/ Multiplicative Decrease’ for congestion control
- Using IS-IS encoding and behaviors
 - Existing ack, loss detection and re-transmission
 - Adding one TLV to advertise parameters
 - Receive Window
 - Amount of “small traffic” if blocked
 - How fast I will ack LSP

Draft changes

- Flooding Parameters TLV may be advertised in both xSNP and Hello
- Encoding:
 - Use of a sub-TLV for each parameter
 - 32 bits values, increased granularity
- New sections:
 - faster acknowledgment of LSPs.
 - faster retransmission of lost LSPs
 - New sub-TLV to signal how fast the receiver will ack the LSPs
- Terminology changes, editorial

Next steps

- Many thanks for the significant constructive discussions and feedbacks.
 - More are welcomed
- Update the draft:
 - Introduction on transmission layer toolbox
 - Congestion control algorithm