# PIM Fast Hello exchange

draft-morin-pim-fast-hello-00

*Thomas Morin – FT Orange*

# Problem statement

- PIM-SM specifications induce a possibly high random delay before neighbors exchange Hellos
  - By default: up to 10s, reduced to 5s if need to send a Join on a link
- Having exchanged Hellos is needed before sending or processing a Join to/from a neighbor
  - ignoring Hellos is doable, but not if you depend on the information put in Hellos to decide how to send a Join
- Multicast traffic blackholing can occur...
  - ...if unicast routing and PIM RPF update happen on a router downstream of a link coming up, before PIM hellos are exchanged
  - Issue discussed in past meetings:
    - draft-morin-mboned-mcast-blackhole-mitigation and draft-asati-pim-multicast-routing-blackhole-avoid

# Proposed improvement

- A simple idea: no delay before exchanging Hellos
- Why the random delay in current PIM-SM specifications ?
  - ➔ Avoid storms of Hellos on LANs
- Proposal:
  - On links that are known to be point-to-point
    - ➔ Set Triggered_hello_delay to zero
    - ➔ A neighbor will “reply” to a Hello instantly
  - On multi-access links / LANs
    - ➔ Recommend to provide a tunable allowing the operator to reduce Triggered_hello_delay, when the number of routers is known to be low enough to not fear Hello storms
    - ➔ Optionally use a modified Hello exchange procedure where a router sends an additional Hello in unicast, when it needs to send a Join to a router for which neighborship is not setup yet. The other router will reply instantly, in unicast too.

# A little bit more detail...

- When messages (Join/Prune/..) needs to be sent on a link toward a neighbor A...
  - 1. a Hello message is sent at once, and a timer T is set to a low value (e.g. 100ms)
  - 2. the messages are queued, and sent on the link when the first of these two events occur:
    - a Hello is received from A
    - T expires
  - 3. if messages were sent because T expired, timers should be set so that these messages will be repeated at once when, eventually, a Hello is received from A
- This allows smooth operations even if the neighbor doesn't implement this spec, and if the neighbor uses relaxed procedures and sends/processes messages even if a Hello was not received/sent before

# How does this help ?

- We can reasonably expect near-instant exchange of Hellos to be faster that RPF update in most cases
  - Should eliminate most blackholing issues due to a link coming up
- Blackholing issues that are left...
  - Misconfiguration
    - → There are other ways to deal with such issues, through adapted operational practices
  - Case where unicast/RPF converges faster than the time required to exchange Hellos
    - → Realistic ?

# Conclusion

- Thanks to Bill Fenner, Mark Handley, Dino Farinacci
- Working group feedback ?
- Adoption ?