Agenda

- History of the draft
- Flood reduction algorithm
- Next steps
History of the draft

- Idea of the flood-reduction algorithm
  - Sourced from RFC 5449 RFC 5614
  - Variant defined and implemented in RIFT
  - Well proven and deployed

- New modified algorithm
  - Uses same basic graph theory principles as MANET
  - Does not require protocol extension/link local flooding nor any form of signaling
  - Does not require all nodes in an “area” to support it
  - Completely distributed

- Open fabric version
  - Used RFC 7356 for link-local flooding
  - Implemented and tested on FRR
  - Another implementation by unnamed but obvious vendor done, tested, works well ;-)
Algorithm in One Picture
Algorithm in Few Words

- All metrics = 1, Overload ignored
- Build SPT from O to TN
- Build RNL and THL from TN’s view
- Order RNL by System ID
- Take RNL elements as re-flooders staring from Hash(LSP-ID) until all THL covered
- Most of those results can be cached easily under stable topology
Transient Flooding Partitioning

- Set a small timer
- On timer expiry send bundled PSNP for all LSPs that have not been reflooded to the neighbors that were not flooded to unless PSNP/CSNP have been received from the neighbors
  - If neighbors up-to-date they’ll send same PSNP
- Process received PSNPs as per standard protocol procedure
Next steps

- Request review and comments
- OSPF version fairly obvious
- WG adoption
Thank you
New modified algorithm

• Stage 1
  • Find set of IS that will all flood to same set of neighbors as local AS
  • Deterministically decide which set of nodes should reflood received LSPs

• Stage 2
  • Find neighbors on the shortest path towards originator
  • Do not flood towards these neighbors
Algorithm

Reflooder

Normal flood
Step 1: Build the Two-Hop List (THL) and Remote Neighbor's List (RNL) by:

- Set all link metrics to 1
- Calculate an SPT truncated to 2 hops from the perspective of TN
- For each IS that is two hops (has a metric of two in the truncated SPT) from TN:
  * If the IS is on the shortest path towards the originator of the modified LSP, skip
  * If the IS is the neighbor of the LSP originator, skip
  * If the IS is not on the shortest path towards the originator of the modified LSP, add it to THL
- Add each IS that is one hop away from TN to the RNL
Algorithm

Step 2: Sort RNL by system IDs, from the least to the greatest.

Step 3: Calculate a number, N, by adding each byte in LSP-ID (without the fragment ID) and fragment ID MOD 2 (allowing for some balancing of LSPs coming from same system ID without introducing excessive amount of state in an implementation) and then taking MOD on the number of neighbors. N MUST be less than the number of members of RNL.

Step 4: Starting with the Nth member of RNL:

- If THL is empty, exit

- If this member of RNL is the local calculating IS, this IS MUST reflood the modified LSP; exit

- Remove all members of THL connected to (adjacent to) this member of RNL

- Move to the next member of RNL, wrapping to the beginning of RNL if necessary