# Discussion of Optimized Flooding Leader/Leaderless and other issues

Les Ginsberg, Cisco
Peter Psenak , Cisco

# Agenda

Historical Perspective

Leader/Leaderless Comparison

“Blast Radius” Discussion

---Pause for discussion--

Algorithm Versioning

Algorithm Support Advertisements

Periodic CSNPs

Backup Slides(Mesh Groups, Algorithm interaction w enablement)

# Relevant Documents

draft-prz-lsr-interop-flood-reduction-architecture

draft-ietf-lsr-distoptflood

RFC9667 - Dynamic Flooding on Dense Graphs (distributed mode only)

This presentation does not object to any of these documents.

Does raise concerns about Leaderless

Discusses some encoding/operational issues.

# Perspective

Update process is the “crown jewel” of link state IGPs – 100% reliability

- Not purging on checksum errors (circa 1993)
- POI to address inappropriate purging (RFC6232 – 2011)
- Minimum lifetime (RFC7987 – 2016)

Optimizing update process has been discussed for at least 25 years – always with caution

Modest improvements (mesh groups, suppress flooding on parallel links)

Not until RFC9667 (2024) was a formal infrastructure defined

- Leader based
- One active optimized algorithm

Only recently was the idea of running optimized multiple algorithms simultaneously introduced

# Perspective(2)

Why is Leaderless Needed??

What has been stated in WG discussion:

- Easier to deploy than Leader based - it is not
- Blast Radius – very misunderstood

Justifiable motivations?

- Need to run different algos in different sub-topologies – is this a real requirement?
- Easier transitions from one algorithm to another – but this comes with risk

# Leader/Leaderless

| Leader Based | Leaderless |
|---|---|
| Per node enablement | Per node enablement |
| Simple Leader Election – same as Flex-Algo | No election |
| No more than one optimized algorithm enabled at a time | Multiple algorithms can be enabled at the same time (intentionally or not) |
| No need to know behavior of neighbors | Need to know what algorithm neighbor has enabled |
| Transitions via base flooding or flag day | Transitions use multiple algorithms |
| Test matrix is well defined | Test matrix is unbounded |

**Just as with base flooding, subtle issues for a new algorithm may take years to discover**
**Add to that interaction issues associated with running multiple algorithms at the same time**

**Only justified if there is an actual need to deploy multiple algorithms in a single area.**

# “Blast Radius” Comparison

Intelligent implementations of Leader based require:

1. Leader selection/advertisement of an algo
2. Local enablement of an algo

in order for a non-default algorithm to be in use on a node.

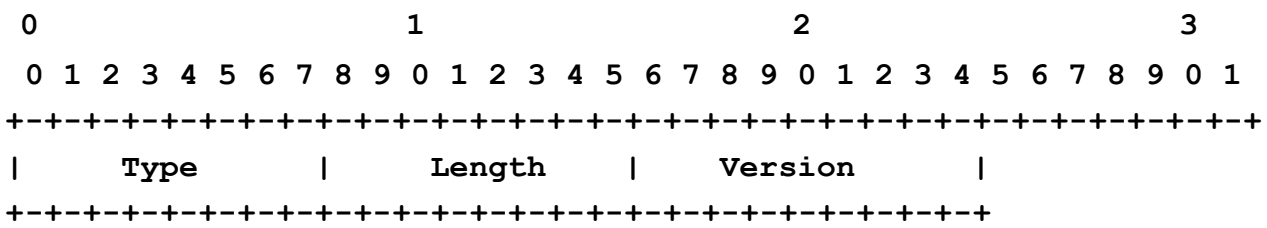
“Blast Radius” for Leader Based/Leaderless are the same (one node at a time)

| Action | Leader Based | Leaderless |
|---|---|---|
| Leader Advertises an Algorithm | Only nodes who have the algorithm locally enabled change their behavior.<br>Operator can limit “blast radius” to 1 by proper ordering. | N/A |
| Local enablement of an algorithm | Neighbors are not affected | Neighbors running a different algorithm must learn of neighbor config change and possibly update flooding to that neighbor |
| Remote enablement of an algorithm | Neighbors are not affected | Neighbors running a different algorithm must learn of neighbor config change and possibly update flooding to that neighbor |

---Pause for discussion--

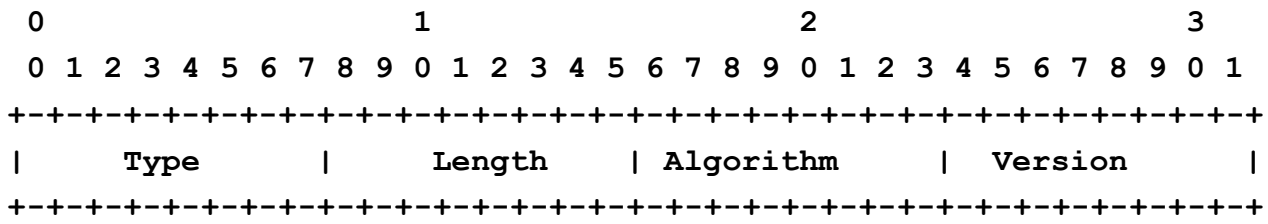
# Version vs Algorithm

draft-ietf-lsr-distoptflood-07 defines a sub-TLV dedicated to distoptflood:




This model means we would need a different code point for every algorithm.

Conflicts with draft-prz-lsr-interop-flood-reduction-architecture



# Upgrades and Versioning

Case 1: Upgrade to a new version of an algorithm

- Upgrade is backwards compatible – no need for versioning
- Upgrade is NOT backwards compatible: identical to deploying a new algorithm. No benefit to using a version

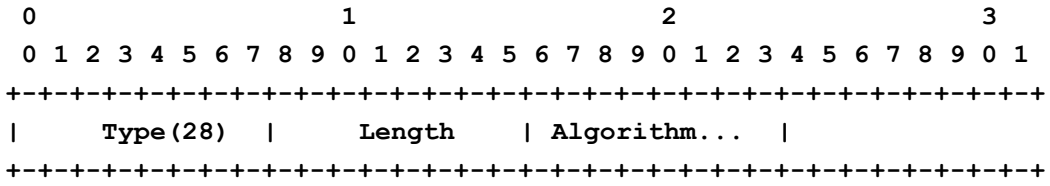
Case 2: Fix a Bug

- Bug in specification: See Case 1
- Bug in an implementation: No way to usefully define vendor specific version for interoperability

Conclusion: Version is not needed

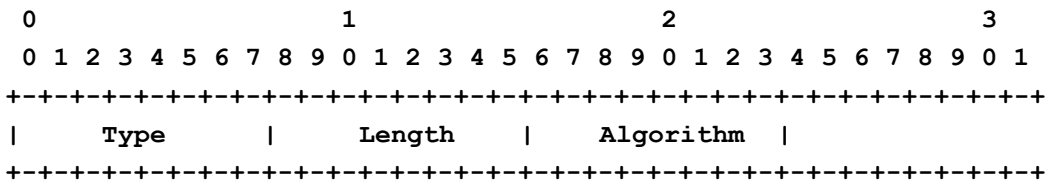
# Advertising Algorithm

## Dynamic Flooding sub-TLV defined by RFC 9667:




1. Indicate that node supports dynamic flooding. This is indicated by the advertisement of this sub-TLV.
2. Indicate the set of algorithms that it supports – NOT what is currently enabled.

## Leaderless nodes need to advertise what is actually enabled:




1. Indicate that it supports dynamic flooding. This is indicated by the advertisement of this sub-TLV.
2. Indicate the algorithm that is currently enabled (independent of Leader/Leaderless)

This advertisement can be sent by both Leader and Leaderless nodes.

Used by Leaderless nodes to determine what algo is enabled.

Options: New sub-TLV or revision of RFC 9677 sub-TLV (TBD)

# Algorithm Registry

Existing registry defined by RFC 9667 is sufficient:

IGP Algorithm Type For Computing Flooding Topology

**A numeric identifier in the range 0-255 that identifies the algorithm used to calculate the flooding topology. The following values are defined:**

**0:Centralized computation by the Area Leader.**

**1-127:Standardized distributed algorithms.**

**128-254:Private distributed algorithms. Individual values are to be assigned according to the "Private Use" policy defined in Section 4.1 of [RFC8126] (see Section 7.3).**

**255:Reserved**

# Use of Periodic CSNPs

Longstanding recommendation when using flooding optimizations (e.g., mesh groups)

Aid to reliability/Required on LANs

Works well at modest scale

At high scale – should be used with caution.

93 LSP Descriptions/CSNP (assumes 1492 MTU)

- 11 CSNPs/1000 LSPs
- 108 CSNPs/10000 LSPs

Number of neighbors has an impact

Targeted PSNPs (recommended by distoptflood) may be better – but timing needs to be considered so as not to undermine flooding optimization

# Backup Slides

# Interaction with Mesh Groups

Mesh groups are a “static flooding algorithm”.

It may be useful to announce that a node uses mesh groups – perhaps by reserving an algorithm value for it (#1 ???)

Beyond the announcement, not useful to try to provide further details about what mesh groups do.

Operators who use mesh groups in combination with optimized flooding algorithms do so at their own risk.

# Implementation Note: Algorithms and Enablement

Flooding algorithms are independent of the enablement method

- The task of an algorithm is to determine whether an adjacency should be used for flooding
- The logic is independent of the method of enablement
- Disablement of flooding optimization based on state of neighbor does not need to be considered by the algorithm and/or can be signaled by an API which is algorithm independent

# Algorithm/Update Process Interaction

Algorithm output => Update process indicates whether flooding should/should not be enabled for each neighbor

Model A:

- Algorithm does not know what algorithm other nodes support – assumes they all support the local algorithm
- Update process overrides flooding disablement if neighbor does not support local algorithm

Model B:

- Algorithm is told (Boolean) whether node supports local algorithm or not.
- Update process follows flooding state output from algorithm