RIFT Auto-FR
draft-head-rift-auto-fr-00
Jordan Head, Tony Przygienda
IETF113
Let’s define the problem.

• We all know that **flat single area** IGPs can come with some pitfalls.
  
  • **Flooding** – every node needs to know.
  
  • **State** – every node needs to remember.
  
  • **Convergence** – every node needs to compute.

• This gets even worse as the network is scaled.

• However, these deployments may be desirable for things like SR.
Let’s visualize the problem.

- Lots of state.
  - Maintain more adjacencies.
  - Maintain a larger LSDB.

- Lots of flooding.
  - Distribute more LSPDUs.

- Slower convergence.
  - More SPF runs and longer runtimes.
  - Higher resource utilization further slows SPF.
What’s the solution?

• IS-IS Flood Reflection!
  • Based on existing LSR work.

• Flood Reflectors are a *bit* like BGP Route Reflectors in that we:
  • Choose a Cluster ID.
  • Designate one or more Flood Reflectors.
  • Designate one or more Flood Reflector Clients.
Let’s visualize the solution.

- Split L2 into multiple flooding domains.
- L1/L2 nodes establish “Flood Reflector” adjacencies in Level 2.
  - Flood Reflectors at T3
  - Flood Reflector Clients at T1
- L1 nodes provide forwarding for Level 2 routes.
  - e.g. Leak L2 routes from T1 into L1.
  - Other methods detailed in the LSR draft.
- L1 and L2 converge independently of one another.
Before and after.

Level 2 Topology

Flood Reflector Topology
Did we improve scale?

- **Example** | Consider a fully meshed topology of 6 L1/L2 IS-IS nodes.

- **Without Flood Reflection**
  - Adjacencies = \( n \times (n - 1) / 2 \)
    - \( n \) = number of L1/L2 nodes.
  - 15 Adjacencies = \( 6 \times (6 - 1) / 2 \)

- **With Flood Reflection**
  - Adjacencies = \( R \times n \)
    - \( n \) = number of L1/L2 nodes.
    - \( R \) = number of Flood Reflectors
  - 8 Adjacencies = \( 2 \times 4 \)
What about the other factors?

• Less links and adjacencies mean less LSPDUs.

• Less LSPDUs means less flooding.

• Less LSPDUs also means less SPF computation.
What’s that got to do with RIFT?

• Flood Reflection, like RIFT is well suited to Clos topologies.

• RIFT builds the “underlay”.

• Auto-FR will use RIFT to build the Flood Reflection topology.
What are the important variables?

• Loopback Address

• ISO System ID

• Network Entity Title

• Flood Reflector Cluster ID
What do they look like?

```rust
pub fn auto_fr_cidsid2isisnet(cid: FloodReflectionClusterIDType, sid: UnsignedSystemID) -> Vec<u8> {
    let mut r = vec![0x49];

    r.extend(&cid.to_ne_bytes());
    r.extend(auto_fr_cidsid2isissid(cid, sid).into_iter());
    r.push(0); // magic end

    assert!(r.len() == 10);

    r
}
```
Auto-FR Analytics

- Provides an overview of the Flood Reflection topology in the fabric from the ToF nodes.

- Auto-FR Clients advertise status via Key-Value TIEs to the ToF.

- Defined via Thrift model (auto_flood_reflection_kv.thrift)
What’s next?

• Co-authorship and comments are welcome.

• Operational considerations and examples
Questions?