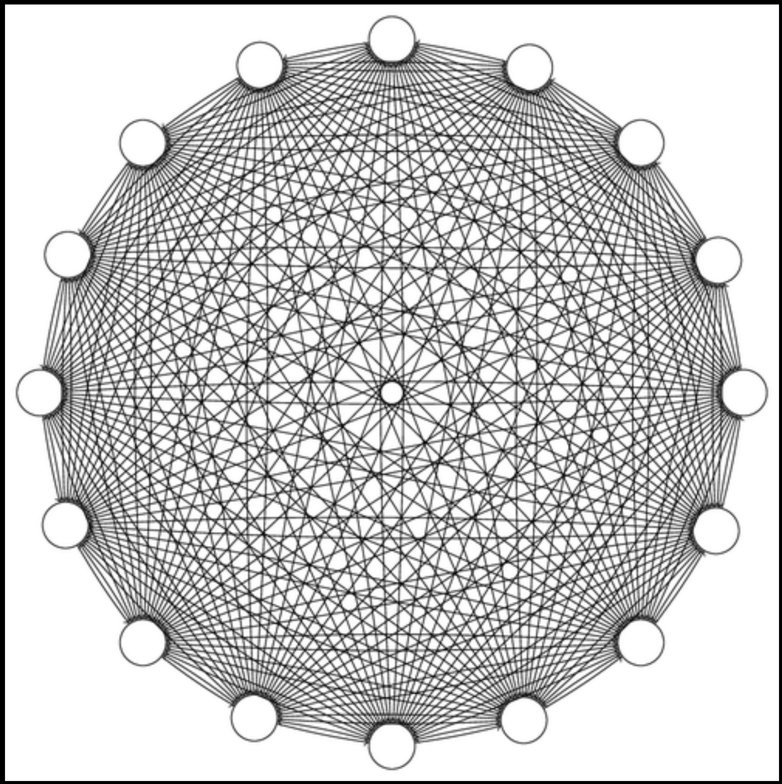
# Dynamic Flooding

Tony Li
Arista

# The Dense Topology Problem

- Link-state IGP's flood updates
  - Works great in sparse networks
  - Fails badly in dense topologies
    - Full-mesh (e.g., Frame-Relay, ATM meshes)
    - DC routing (spine-leaf, fat-tree, Clos)

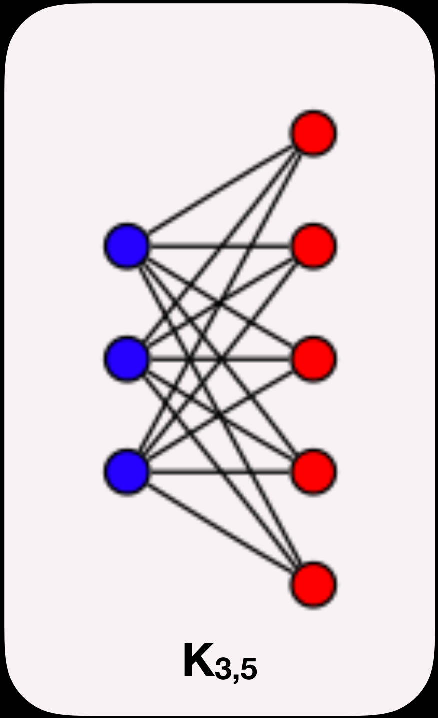


# Don’t flood on all links!

- Previous hack: mesh groups
    - Manual configuration; doesn’t scale
- Better idea: compute a subset of the topology just for flooding
- Which system does the computation? Elect a leader.
- Advertise the flooding topology.

# Example

- Topology: $K_{20,200}$ (20 spines, 200 leaves)
- Links: 4,000
- Flooding topology: cycle using round-robin selection
  - Links: 400 (90% reduction)
  - Cycle protects against single failures in the flooding topology
  - Spines: flood on 20 links



# Mechanism: Leader Election

- Need one (and only one) system to compute the flooding topology.
- Like DR/DIS election, but across the LSDB.
- Proposal: Add a TLV to indicate eligibility and priority.
- Usual tie breakers.

# Mechanism: System List

- Distribute the flooding topology as an adjancency matrix. Assign indices in this matrix by a list of system ID's.
- Proposal: Add a TLV to carry this list.

| **Index (implied)** | **System ID** |
|---|---|
| 0 | 01:02:03:04:05:06 |
| 1 | 01:02:03:07:08:09 |
| 2 | 01:02:03:0a:0b:0c |
| 3 | 01:02:03:0d:0e:0f |

# Mechanism: Adjacency Matrix

- The adjacency matrix itself is just a list of bits.
- Left to right, top to bottom.
- Proposal: Add a TLV to carry these bits.

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | | 0 | 1 | 1 |
| 1 | | | 1 | 1 |
| 2 | | | | 0 |
| 3 | | | | |

# Mechanism: Flooding Path

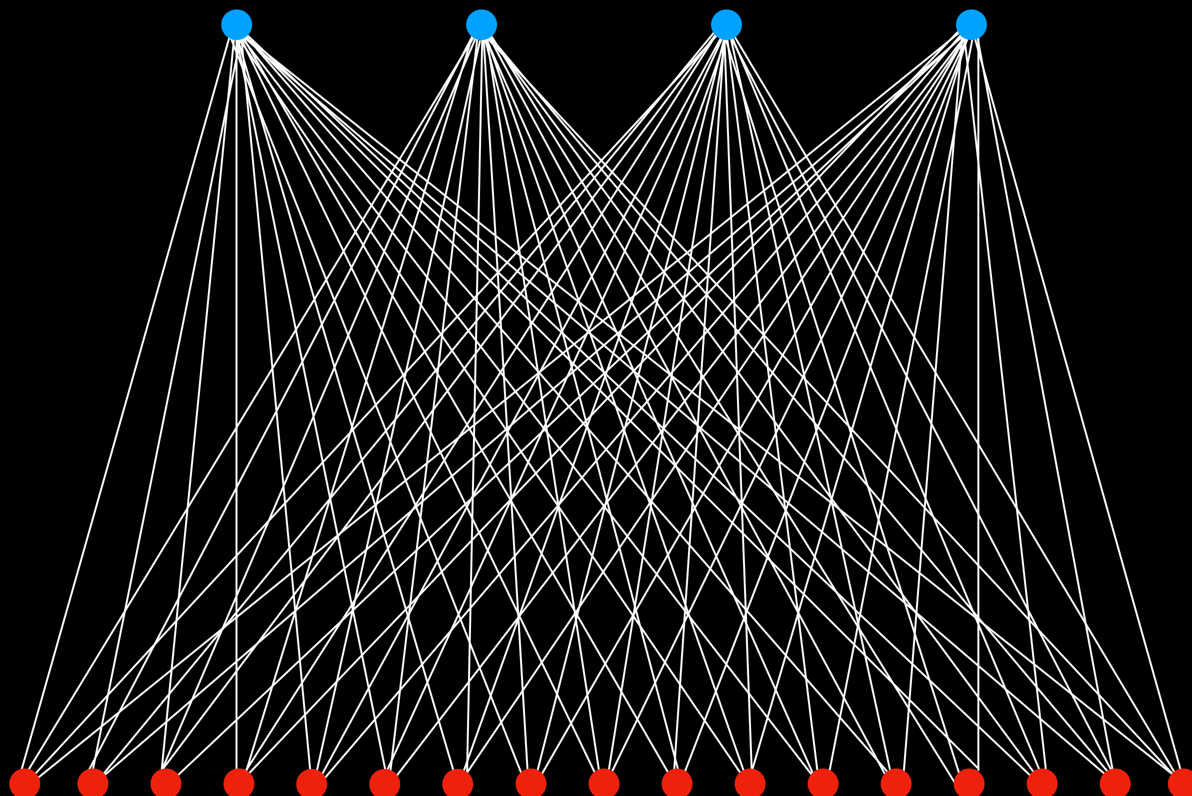
- Encode topology as a set of paths.
- Each path is a sequence of system indices.
- More efficient at scale for sparse topologies.

- Path: 0, 2, 1, 3, 0

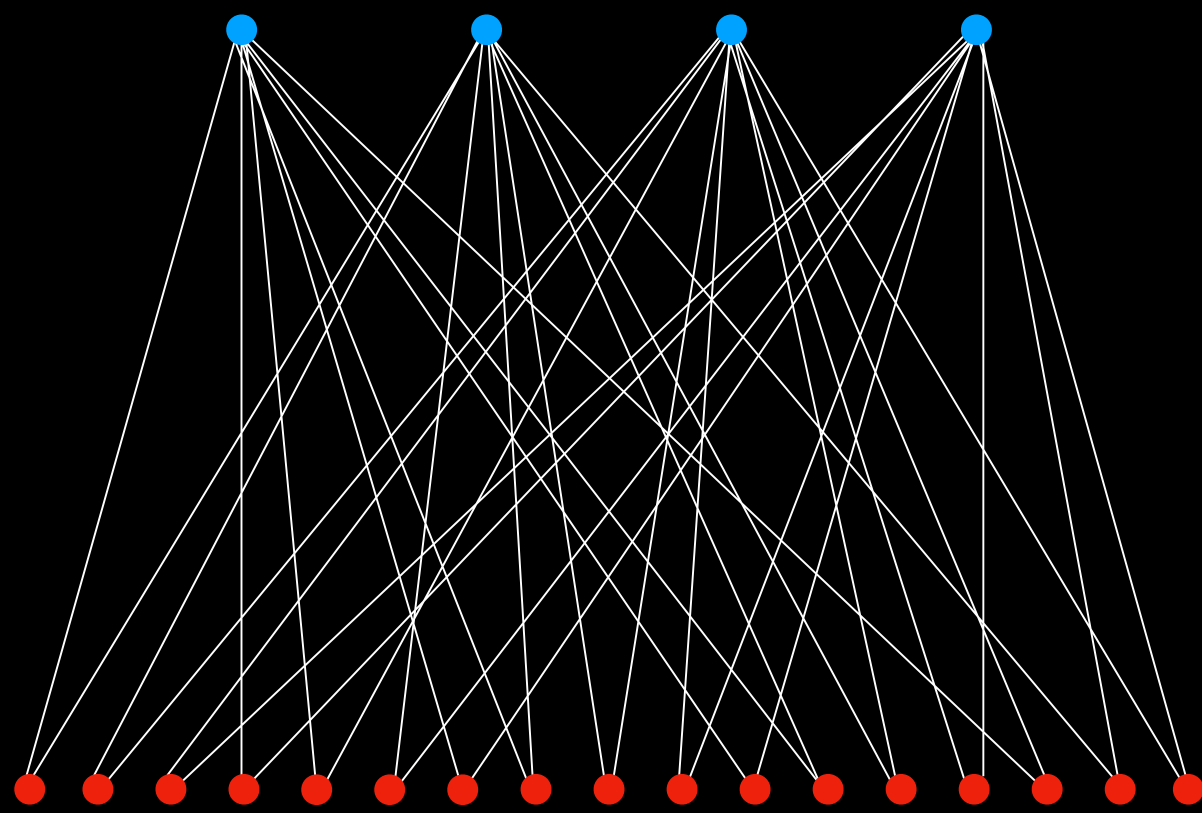
# Computing the Flooding Topology

- Required: All nodes
- Required: Bi-connected
  - Optional: Higher connectivity
- Desired: Minimize node degree
- Desired: Minimize diameter
- Topology is a local computation — need not be in an RFC
- What’s optimal? For further study…

$K_{4,17}$ (68 links)



# Minimal Flooding Topology (34 links)



# Theorem

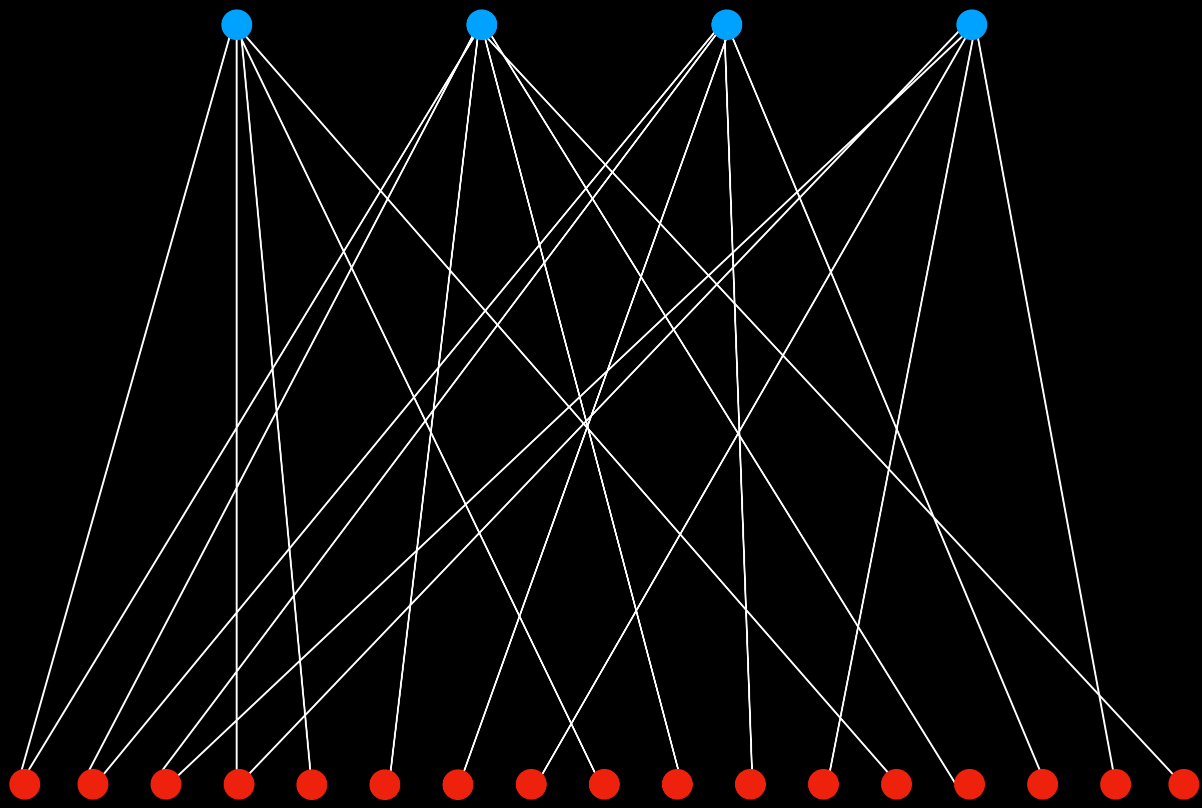
- A **minimal flooding topology** on a spine-leaf architecture is bi-connected with d(leaf)=2.
- For $K_{n,m}$ where m >= n(n/2-1), there is a minimal flooding topology with diameter 4.
- If you have a large enough topology, dynamic flooding performs very well.

# Xia Topologies

- Create a cycle of spines interspersed with leaves
- Any unconnected leaves connect with a single link

- Some leaves are a single point of failure. Work around by reacting on those failures.
- For $K_{n,m}$, diameter is m+2
- All nodes receive updates no more than twice
- Spines send m/n updates

## Xia Topology (21 links)



# Benefits

- When attacking scalability problems, use ALL available tools!
- Dynamic flooding applies to ALL dense topologies, not just spine-leaf
  - Lateral links are not an issue
  - Connectivity outside of DC is not an issue
- Can be combined with other proposals (e.g. draft-shen-isis-spine-leaf-ext) that have topology restrictions giving even better scaling