Routing in Fat Trees (RIFT) Update draft-rift-rift-02

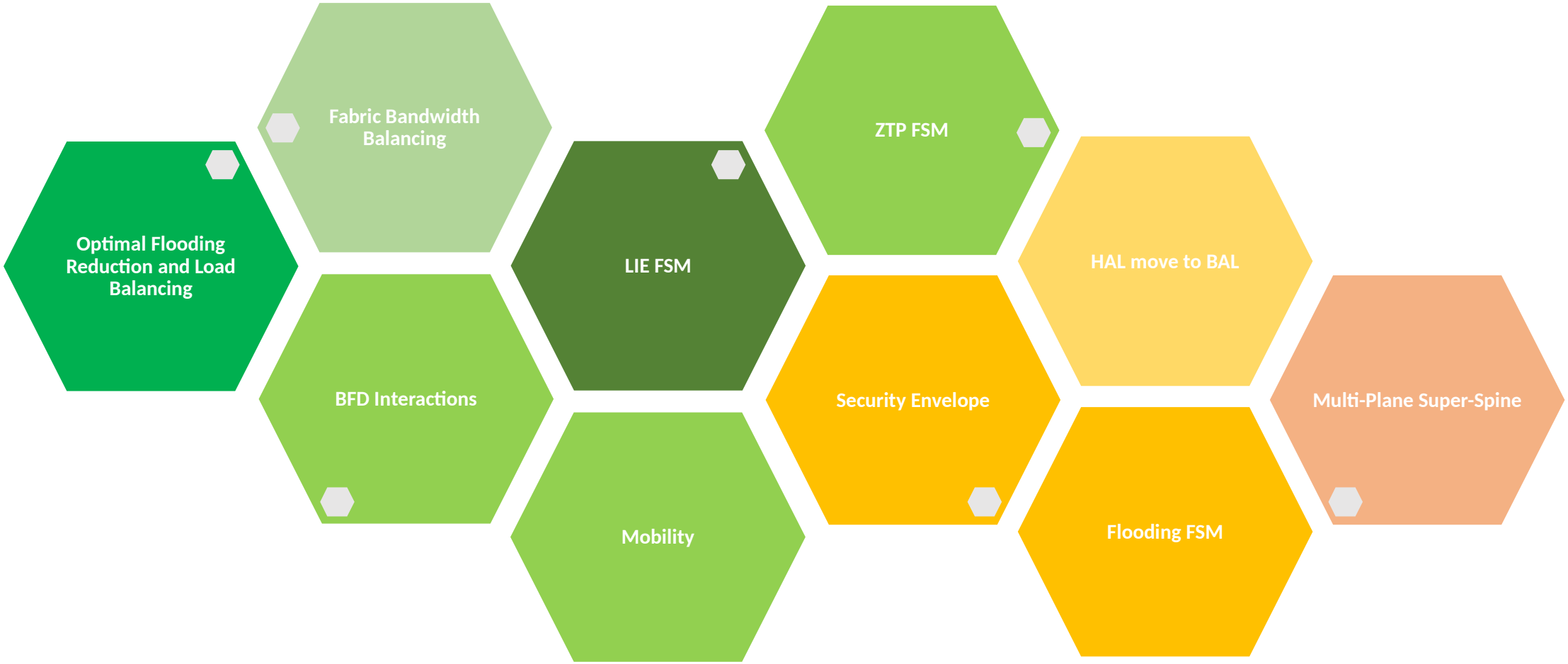
IETF 102, 7/18, Montreal

The RIFT authors' brotherhood

Update from -05, -01 and -02

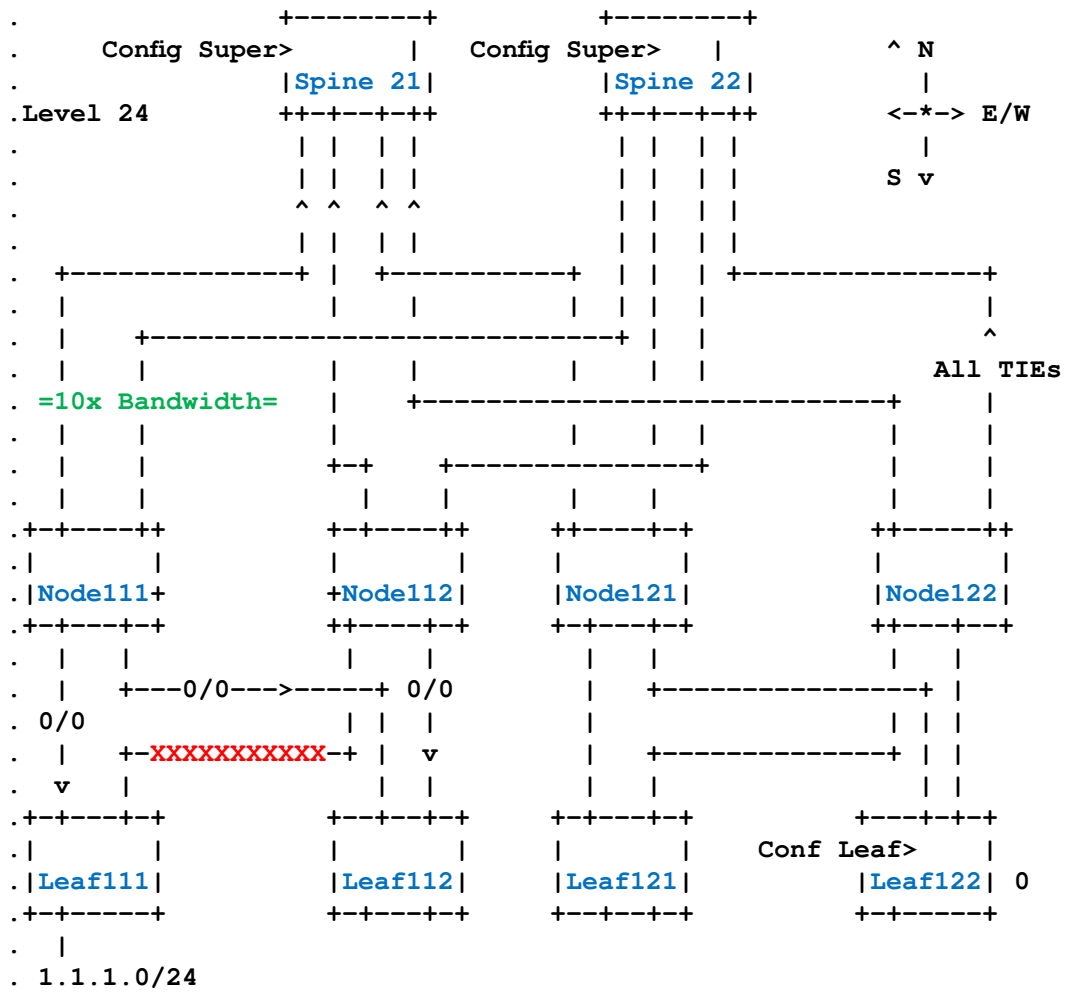
- Last version presented in London –private-04 or so
- We went to -05, rift-00, -01 and -02 in meantime and -03 is under works already with tons hackathon input and last completeness issues
- Interim was held around -00 with some -01 topics
  - This preso includes it if you missed but without the “running code” walk-throughs and detailed drill-downs
  - You can always watch the recording
- This will be a longish preso with tons never-seen-before material, settle down comfortably

Update, Green is Done, Red in -03/-04

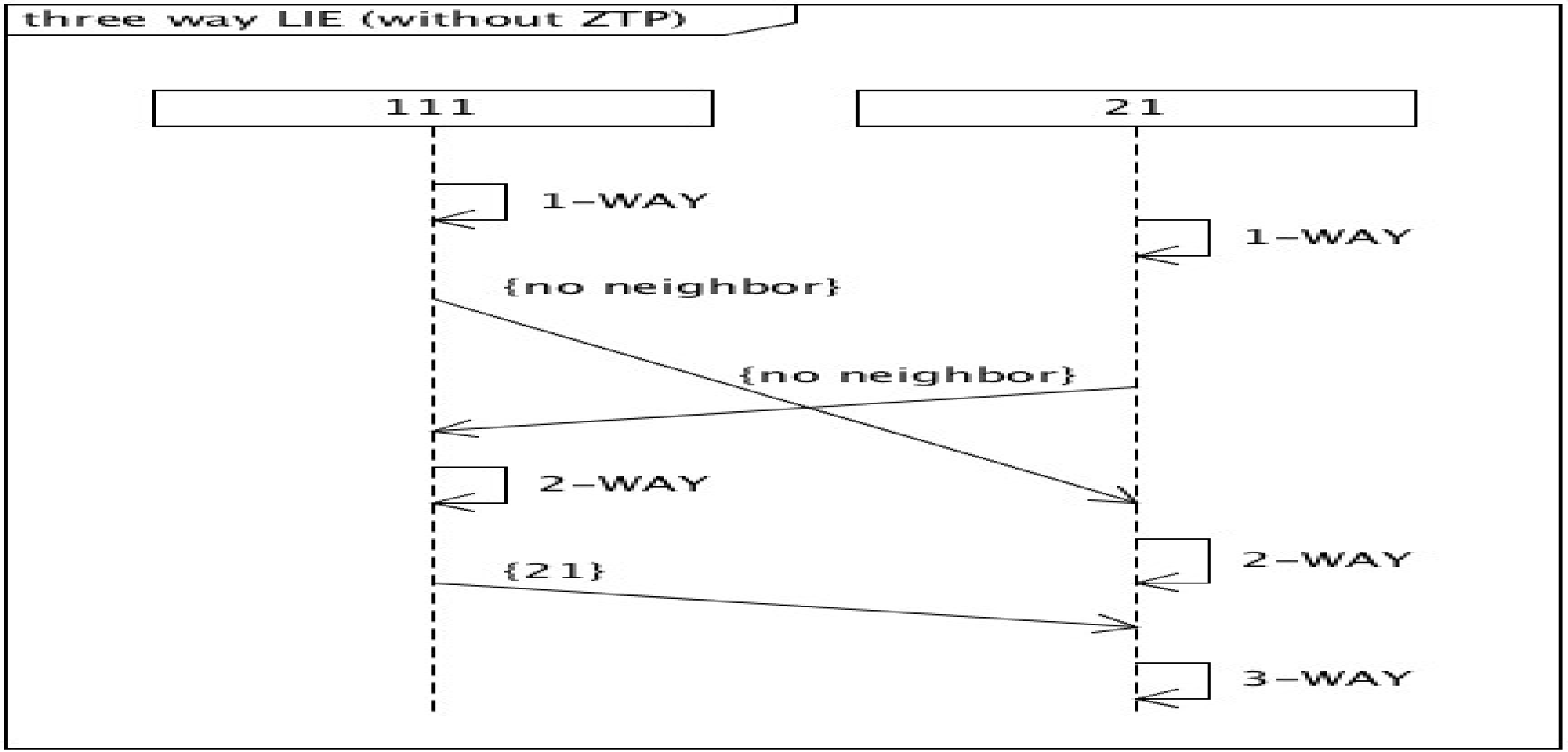


Mini Fabric Used in This Presentation

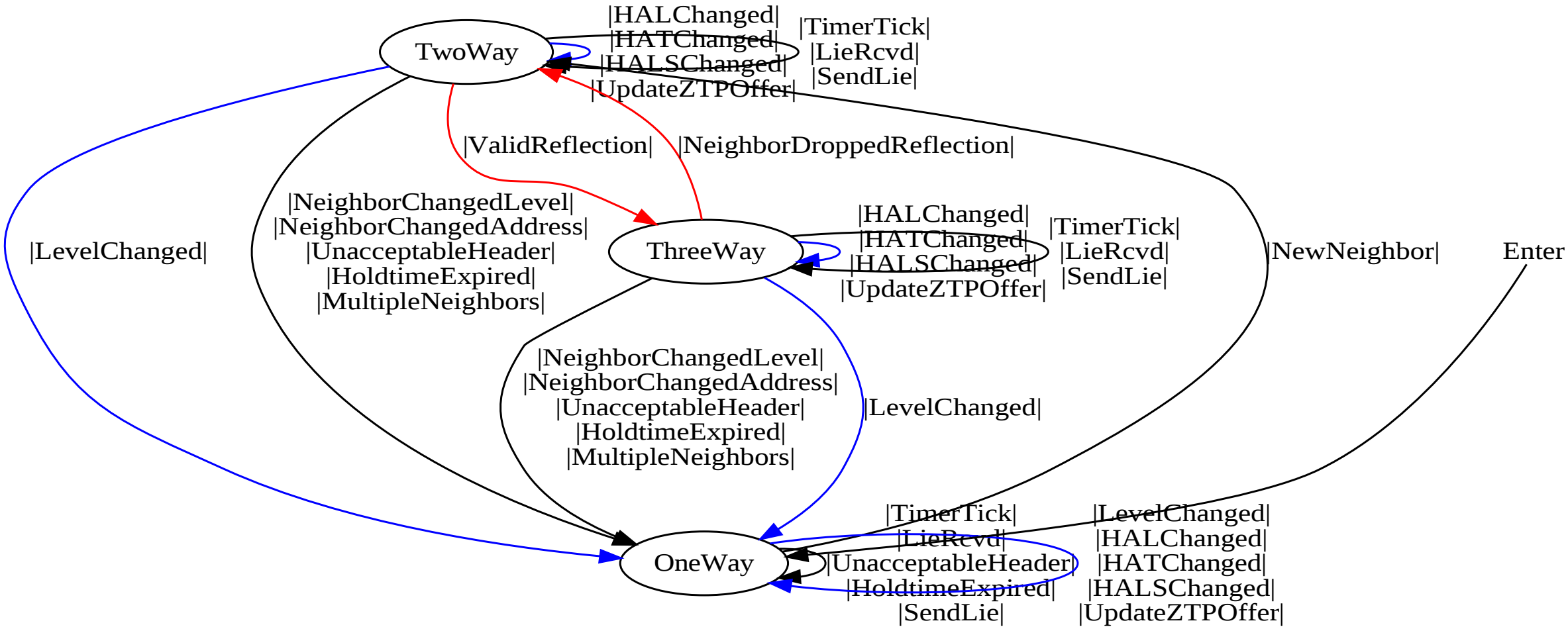
- Configuration
  - Only Superspines MUST be set
  - One Leaf Fixed
  - One Prefix



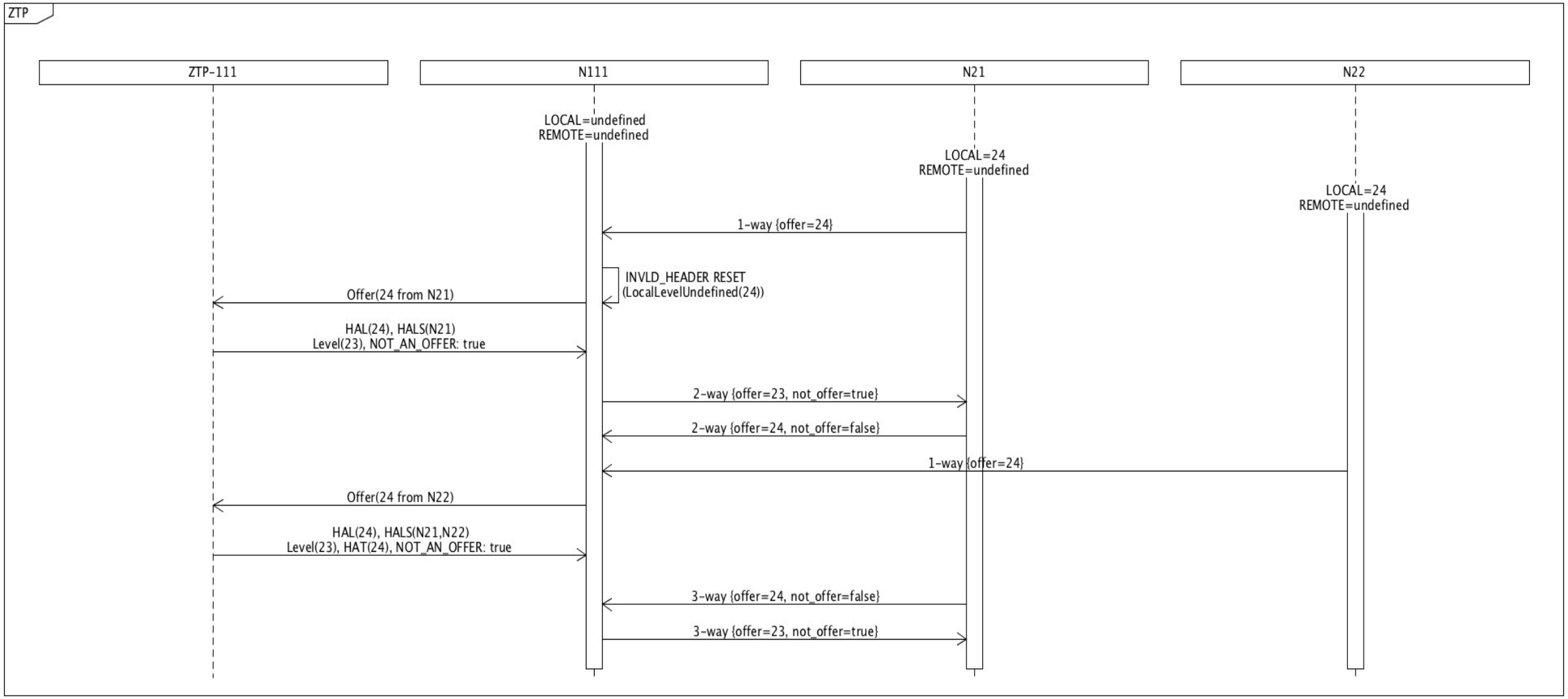
LIE Flow: Warm-Up, Node 111 Spine 21



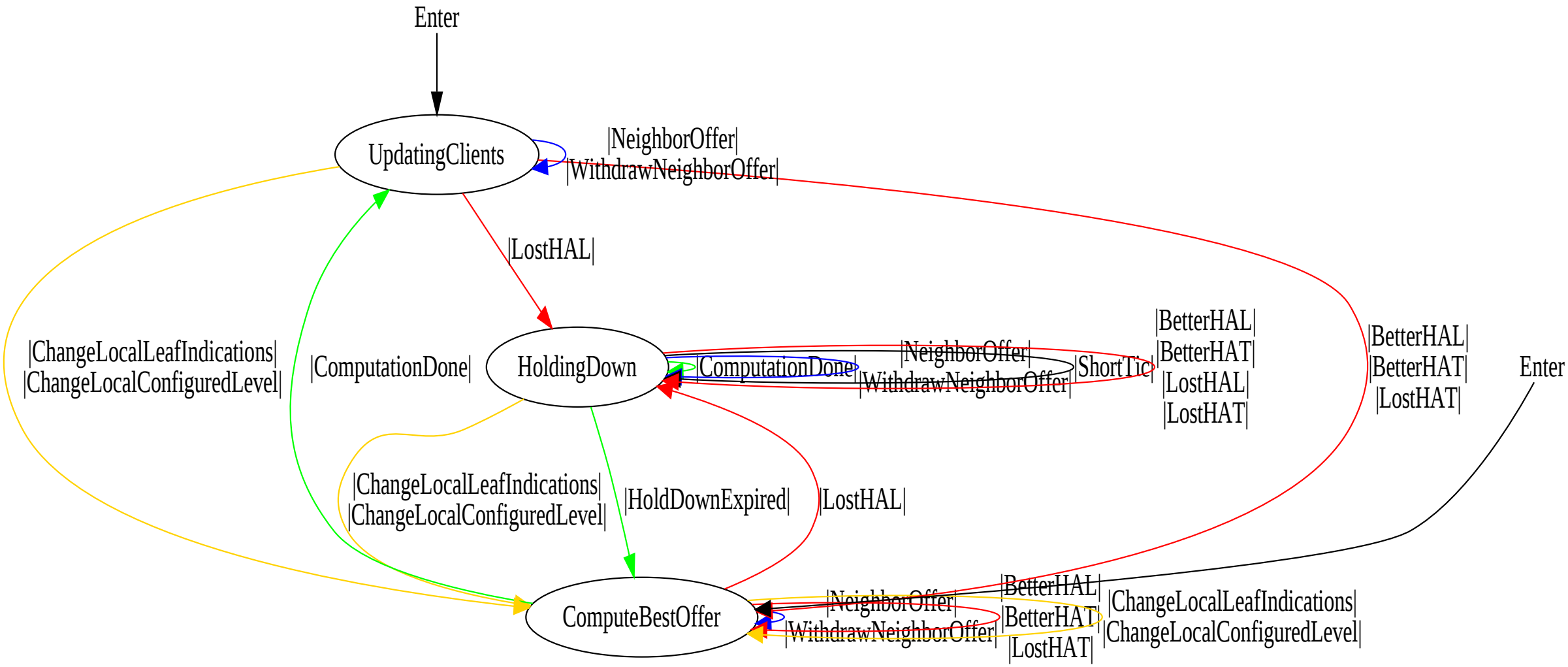
LIE FSM Spec'ed Out in Detail



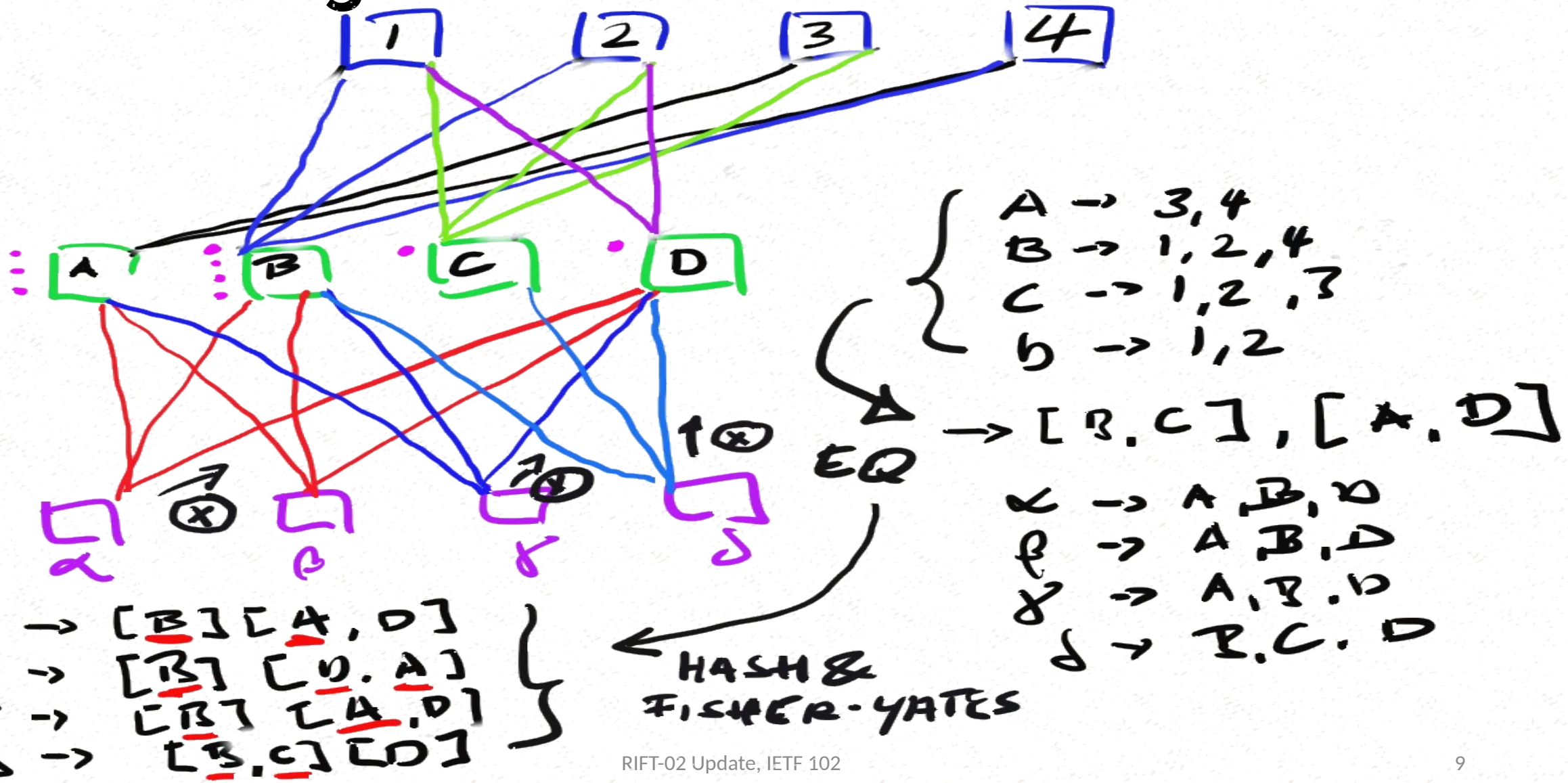
ZTP: Warm-Up


ZTP
ZTP-111
N111
N21
N22
LOCAL=undefined
REMOTE=undefined
LOCAL=24
REMOTE=undefined
LOCAL=24
REMOTE=undefined
1-way {offer=24}
INVLD_HEADER RESET
(LocalLevelUndefined(24))
Offer(24 from N21)
HAL(24), HALS(N21)
Level(23), NOT_AN_OFFER: true
2-way {offer=23, not_offer=true}
2-way {offer=24, not_offer=false}
1-way {offer=24}
Offer(24 from N22)
HAL(24), HALS(N21,N22)
Level(23), HAT(24), NOT_AN_OFFER: true
3-way {offer=24, not_offer=false}
3-way {offer=23, not_offer=true}

ZTP FSM: Spec'ed Out in Detail



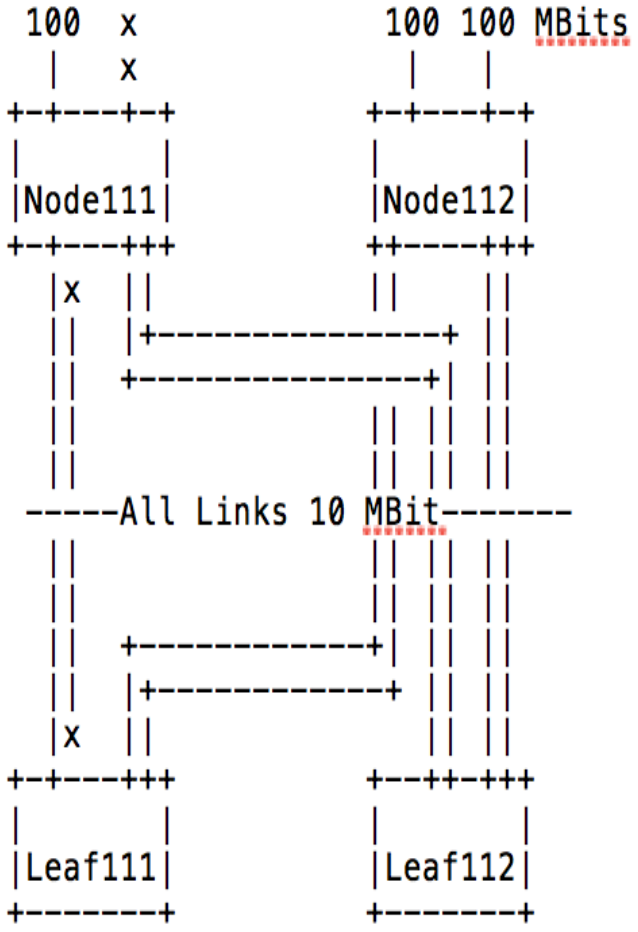
Optimal Flood Reduction **& Load Balancing**



Optimal Flood Reduction **& Load Balancing Continued**

- Vastly superior to previous solution since operational under any cabling failure conditions
- For 5-stage folded this is best solution possible
- For 7-stage (recursive application) this is close to optimal
- Completely stable independently whether nodes implement it or not and whether they use precisely same algorithm or not
- No implementation numbers yet but should be better than suggested, current solution

Fabric Bandwidth Balancing

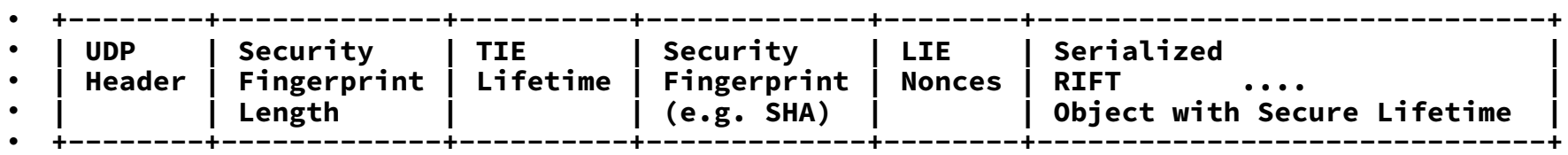


•RIFT calculates the amount of northbound bandwidth available towards a node compared to other nodes at the same level and adjusts the default route distance accordingly to allow for the lower level to have different weights on load balancing.

- **BAD_N:** Bandwidth Adjusted Metric to N
- **L_N_u:** as sum of the bandwidth available from L to N
- **N_u:** as sum of the uplink bandwidth available on N
- **T_N_u:** L_N_u + N_u
- **M_N_u:** log_2(next_power_2(T_N_u))
- **BAD_N:** D * (1 + maximum_of_all(M_N_u) - M_N_u)

Node	N	T_N_u	M_N_u	BAD
Leaf111	Node111	110	7	2
Leaf111	Node112	220	8	1
Leaf112	Node111	120	7	2
Leaf112	Node112	220	8	1

Secure, Optimized RIFT Information Element Envelope Suggestion



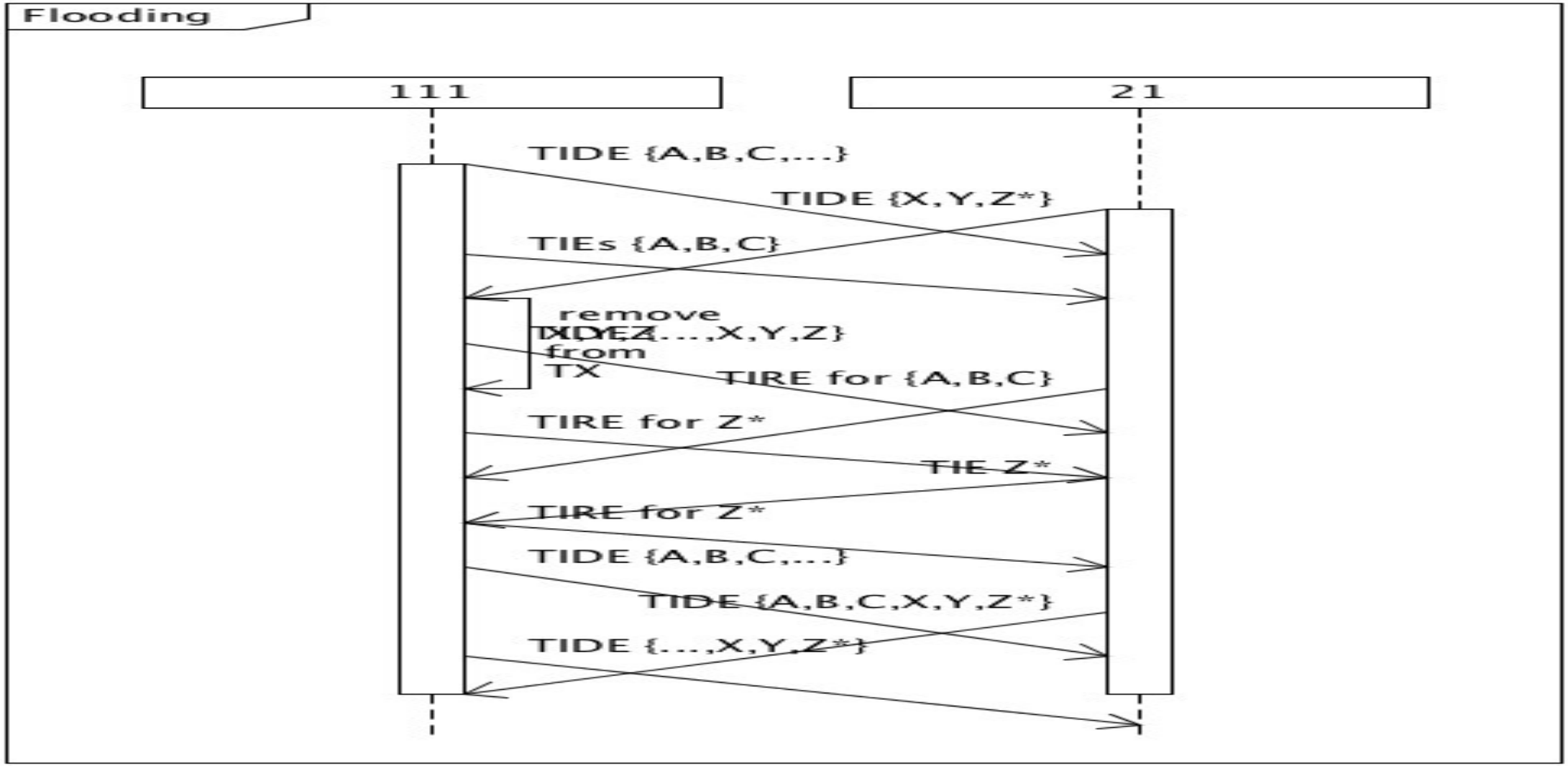
- Avoids Problems we found over years with traditional link-state protocols when securing them
- Maximizes Flooding Speed (No Re-Serialization, No Lifetime protection)
- Security Fingerprint Does Not Get Affected by TIE LifeTime Changes
- Serialized Object Keeps Its Fingerprint and Does Not Need Re-Serialization on LifeTime Field Change by Every Node
- Provides Optimal Security (Lifetime Attacks Are Solved By RFC7987)
  - In object when originated we add optional, secure, absolute timestamp
- Lie Nonces Are Protected by Fingerprint Against Replays, Reflect Neighbors' Nonce, could be kept in the LIE Packet Given They Change All the Time anyway but there maybe other uses
- Only Node with Private Key Can Generate the Fingerprint (Either for LIEs One-Hop or for TIEs Providing Origin Validation and Integrity)

Mobility Support

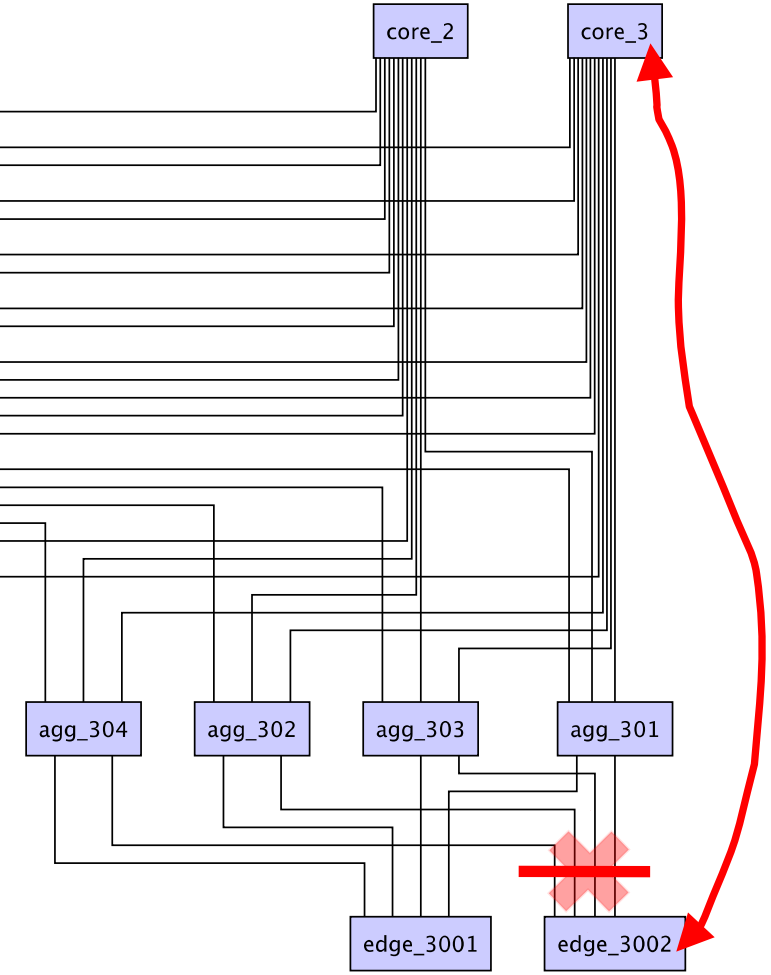
```
struct PrefixAttributes {
…
    /** optional monotonic clock for mobile addresses */
    4: optional PrefixSequenceType    monotonic_clock;
}

struct PrefixSequenceType {
    1: required IEEE802_1ASTimeStampType  timestamp;
    2: optional PrefixTransactionIDType   transactionid;
}
```

- Optional clock attribute on prefix
- If clock not present, anycast
- If present, always better than none
- If both present, RFC5905 or better on fabric assumed
  - If IEEE802_1 less than 200msec diff, transactionid (TID) if present tie-breaks
  - Otherwise timestamp compares
- TIDs are coming from draft-ietf-6lo-rfc6775-update or similar mechanisms

Flooding FSM: Warm-Up Flow



HAL to B(est) Available Level for Leafs



- Current HAL rule will pick the adjacency to core_3 as best and abandon all adjacencies as lower level for edge_3002 (leaf)
- Seems undesirable
- Default rule will be changed to BAL = best link*bandwidth product achieved at a level
  - Bandwidth will be carried on LIEs now as well
- Can be changed/configured per leaf
  - Does not have to be uniform even

Partitioned Superspines and Complex Failures Preview

- Next version will have an extensive write-down on “partitioned superspines”, i.e. something where top of fabric does not see each other fully via southbound reflection
- To deal with all failures in such cases we will need ”negative transitive disaggregation”
  - A prefix ~X/Y will be advertised southbound by a superspine A
  - Node below installs under default X/Y with nexthop ‘everyone except A’
  - When all spines above advertise ~X/Y, the ~X/Y is propagated southbound
- Positive disaggregation at most one level (fast reflex arc)
- Negative transitive disaggregation can heal failures that need decision at leaf to prevent blackholing

THANK YOU FOR YOUR ATTENTION