RIFT Multicast

Jeffrey Zhang
Pascal Thubert

IETF104, Prague

Background

- RIFT core design team realized that the flooding reduction mechanism can easily be extended to provide built-in multicast support
  - w/o additional multicast specific signaling
- This turns out to be very similar to PIM-BIDIR
  - Traffic travels north all the way and fork down south along the way
- Further considerations & discussions led to using separate multicast signaling after all
  - Elephant flow; load-balancing
- Current thinking is enhance and extend PIM-BIDIR concept with native RIFT signaling

PIM-BIDIR Background

- (*,G) Joins are sent “upstream” towards a Rendezvous Point Address (RPA)
  - RPA is either on a particular router, or just an address on a LAN not bound to any router
  - The link that the RPA is on is RP Link (RPL - a loopback or a LAN interface)
  - The joins establish sub-trees rooted at the RPL routers (routers on the RPL)
  - The RPL connects the sub-trees into a tree
- Traffic flows along the tree
  - Upstream towards the RPA, eventually arriving at RPL routers
  - Along the way, traffic also forks to downstream routers from which (*,G) joins are received
  - RPL routers flood all traffic to each other
    - They don’t send joins to each other
    - This is fine on a LAN
  - Traffic received on the RPL (from other RPL routers) is sent downstream as needed
- With BGP-MVPN, the provider network can be used as a RPL
  - PEs are RPL routers
  - But they can send joins towards each other, for selectively sending traffic

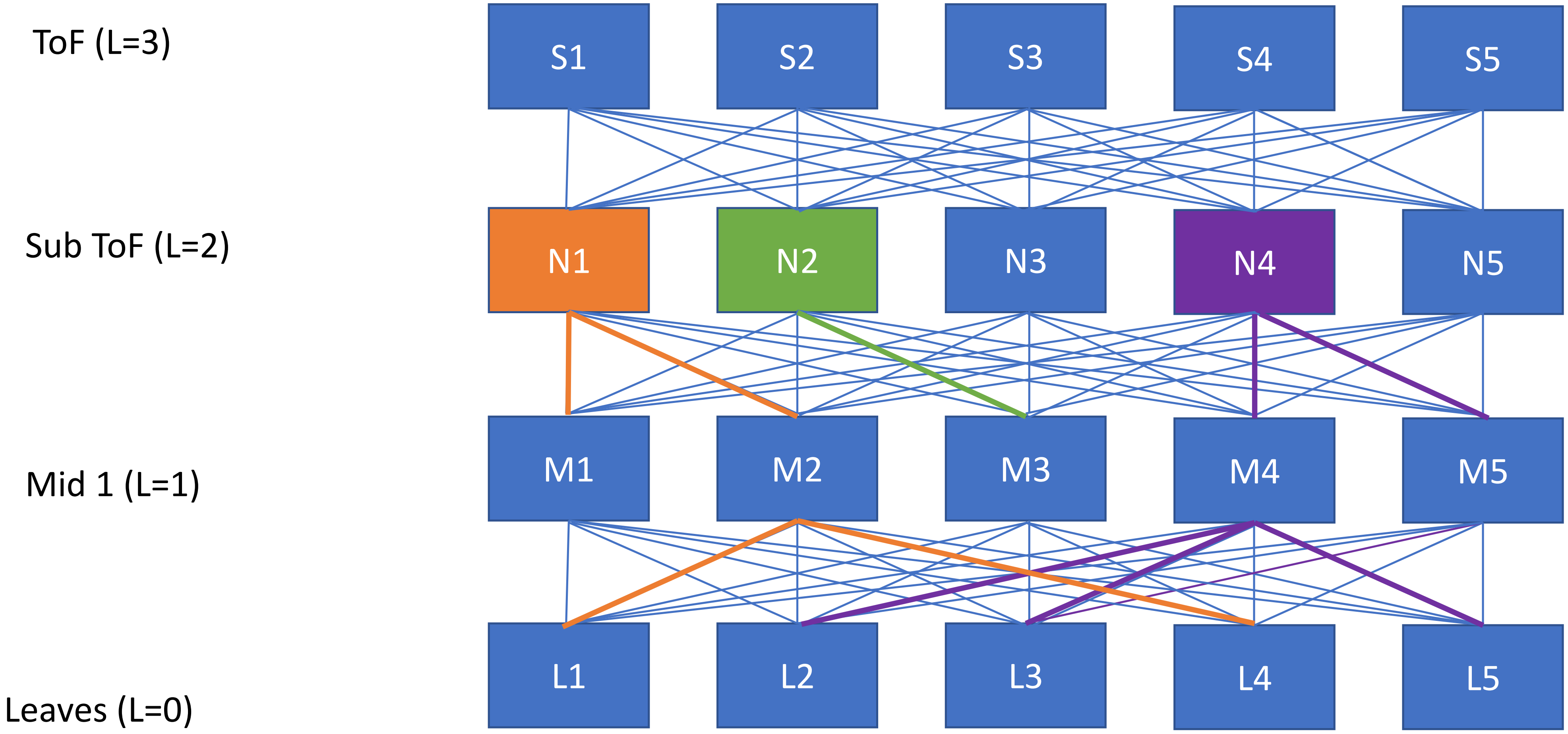
PIM-Bidir Adapted for RIFT 1/2

- No explicit RPA
  - Joins just follow the default route based on control plane hashing
  - *Problem – there is no RPL (the ToFs aren’t connected)*
    - *See later slides*
- Bidirectional (*, G-prefix) trees
  - G-prefix can be ‘*G*’ or ‘***’ to the two extremes, or anything in between
  - (*,*) for “mice” flows - traffic sent everywhere – even if no receivers
  - (*,G) for “elephant” flows – sent only where there are receivers for G
  - (*,G-prefix) for “giraffe” flows
    - sent where there are receivers for any group in the G-prefix

PIM-Bidir Adapted for RIFT 2/2

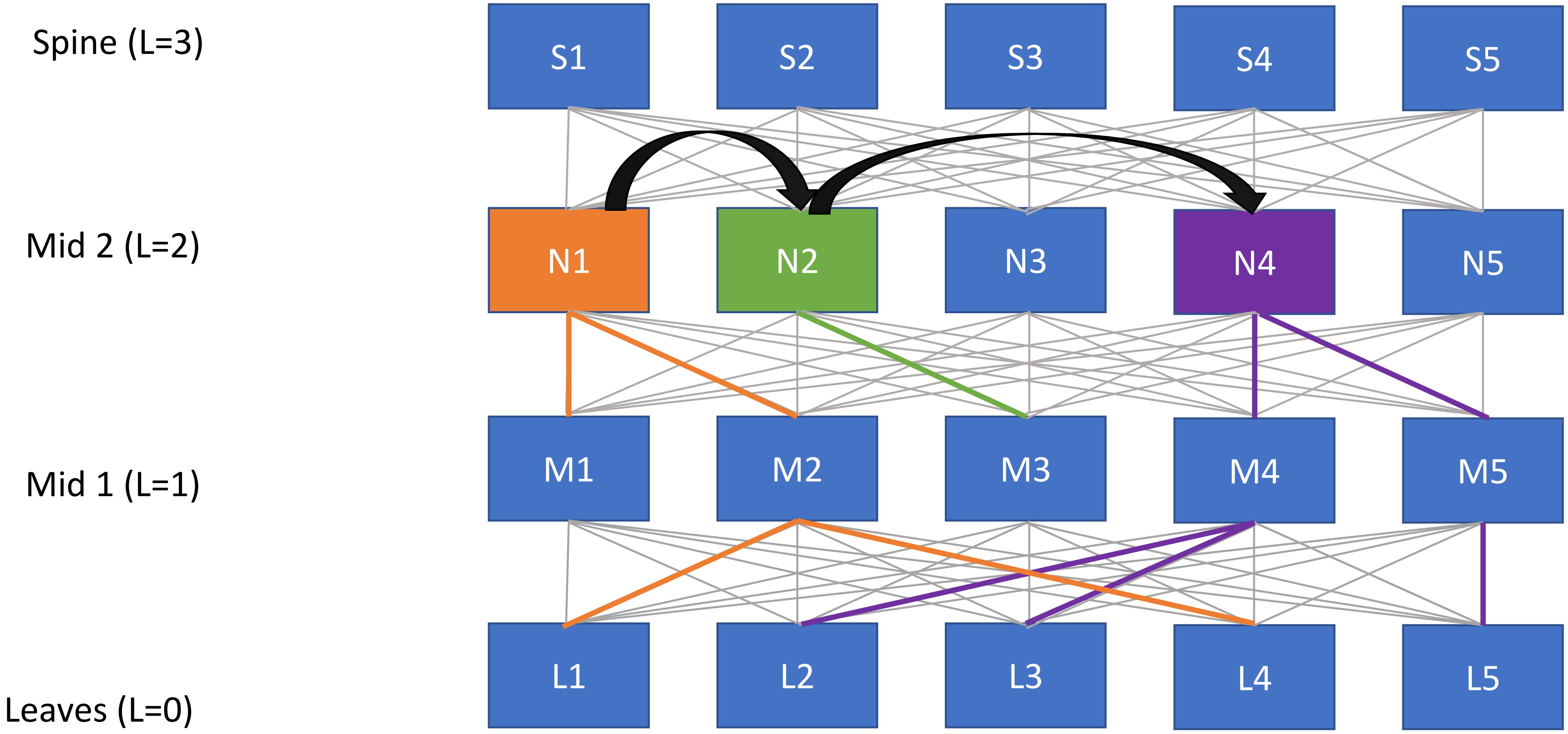
- Joins are done with N-PGPs
  - Consumed, merged and re-originated at every hop
- But only sent to ONE of the north neighbors
  - Chosen by downstream with hashing
    - Load balancing different groups to different upstream neighbors
    - Different downstream nodes will pick the same upstream neighbor for a particular group
      - Even if they somehow pick different upstream it will still work
      - Hash algorithm should prevent too many downstream nodes from picking the same upstream
        - So that the upstream does not have to replicate to too many downstream neighbors
- (*,G)/(*,G-prefix)/(*,*) forwarding state built accordingly
  - Interface list includes hashed northbound interface, and southbound interface on which a join is received
  - Traffic arriving on any of the interfaces forwarded out of others in the list

RPL Problem:
disjoint sub-trees rooted at the Sub ToF



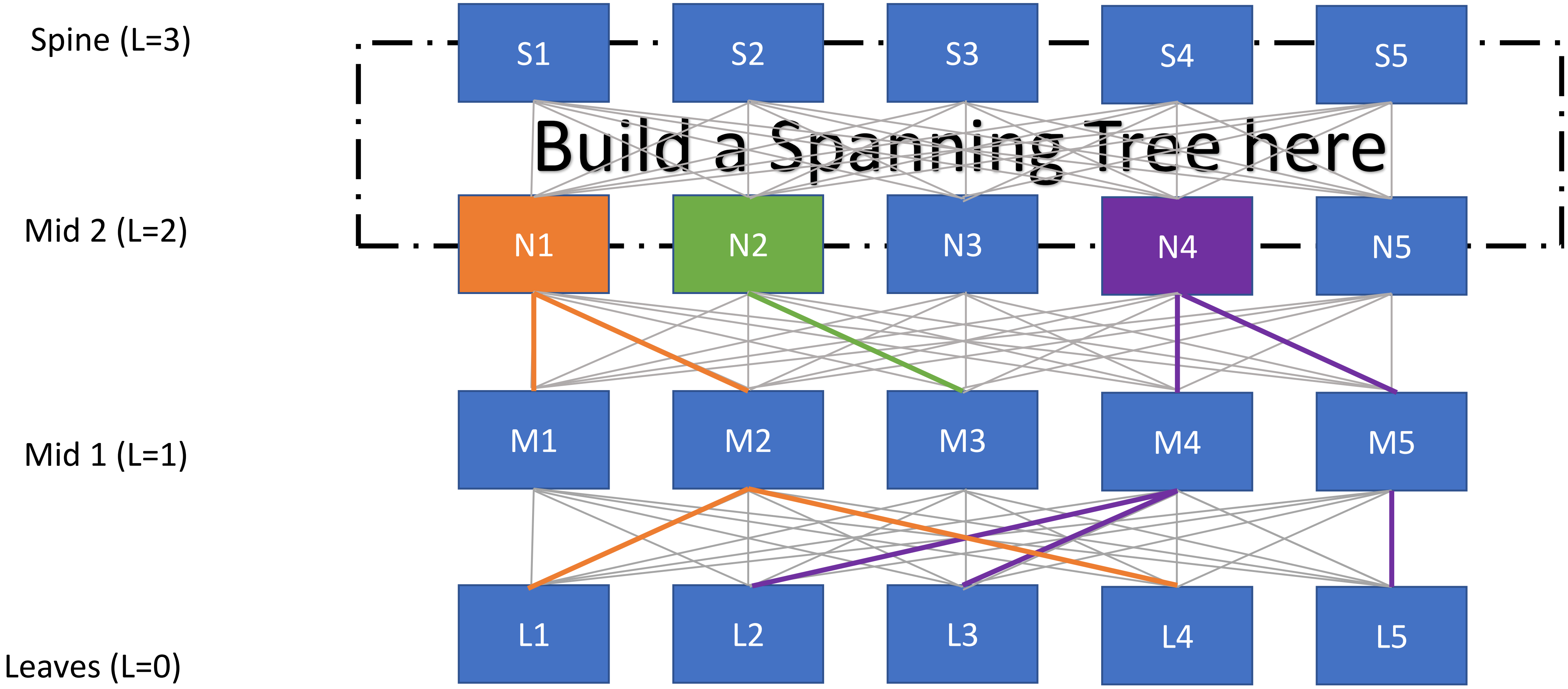
Problem: Build a meta tree (a tree of sub-trees).

Goal: connect the sub-trees



Proposal: Build a loopless a meta-tree (a tree of trees) by joins those trees via the superspine

Approach:
build a spanning tree of ToF and SubToF



The spanning tree must span all subToF and may span some or optionally all ToF nodes

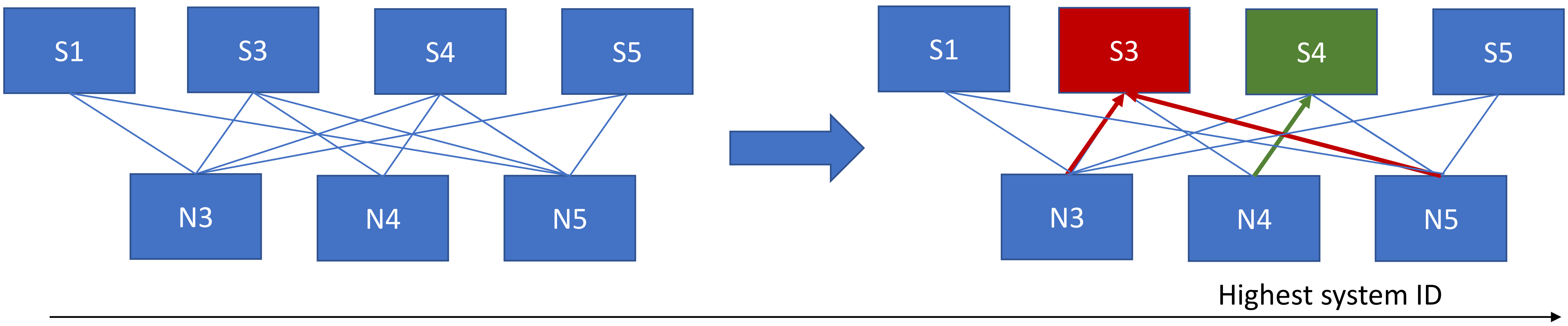
Proposal: Step 1, subToF selects a parent ToF

A hash may determine a subset of ToF nodes

⇒ That subset of ToF nodes now become partial roots

SubToF nodes advertise the ID of their parent to other ToFs

Result: A set of partial trees



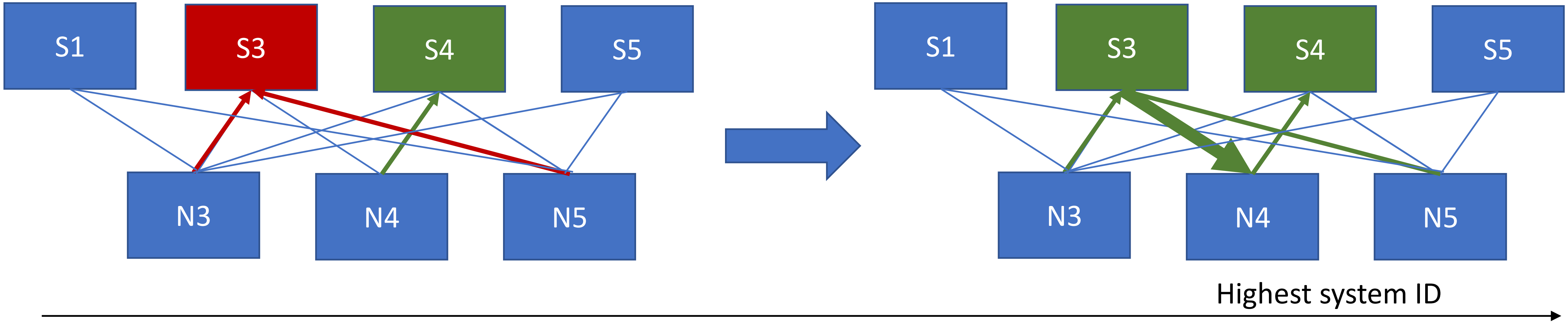
Proposal: Step 2, SubRoot join Main Root tree

Main Root is highest system ID of the Roots (S4 here)

⇒ SubRoot can parent to a subToF in a tree with higher sysID

SubRoots nodes now advertise the higher sysID

Result: A subset of ToF nodes are partial root



Result a spanning structure with subset of ToF