Яндекс

Controlled Disaggregation and Multihoming in DCNs

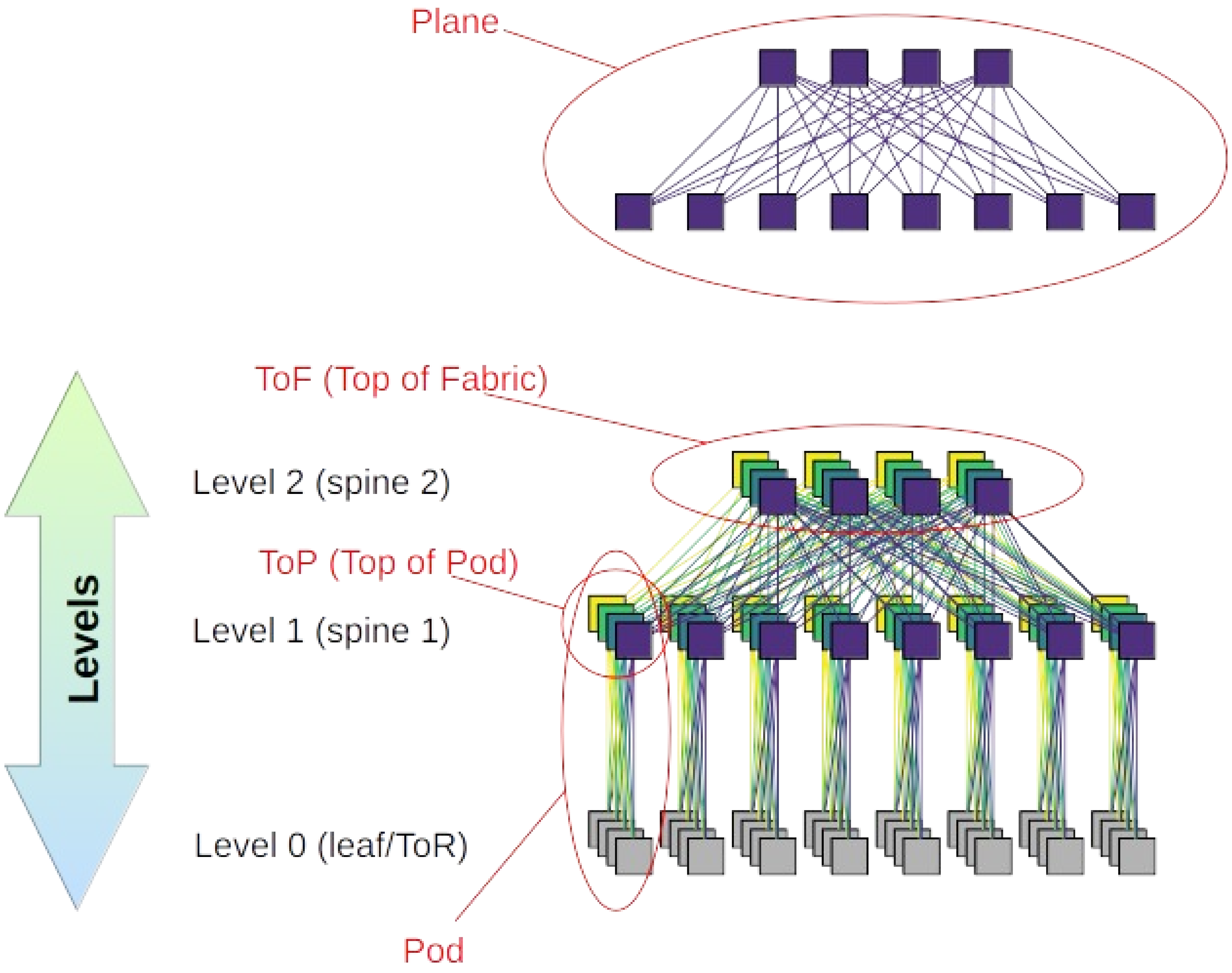
Dmitry Afanasiev

IETF 106: Singapore, Nov 2019

Conditional Disaggregation and Multihoming in DCNs

- Why do it
- Some problems which are difficult to solve with BGP
  - In Fat Tree every level is completely disjoint (no connectivity within level)
  - Some decisions can't be made using only local info
  - But can be elegantly solved with RIFT
- Discussion scope:
  - L3 DCNs only
  - reachability only, not asymmetrical bandwidth
  - no discussion of RIFT internals - available elsewhere

Multi-stage Clos Overview



Plane

ToF (Top of Fabric)

Level 2 (spine 2)

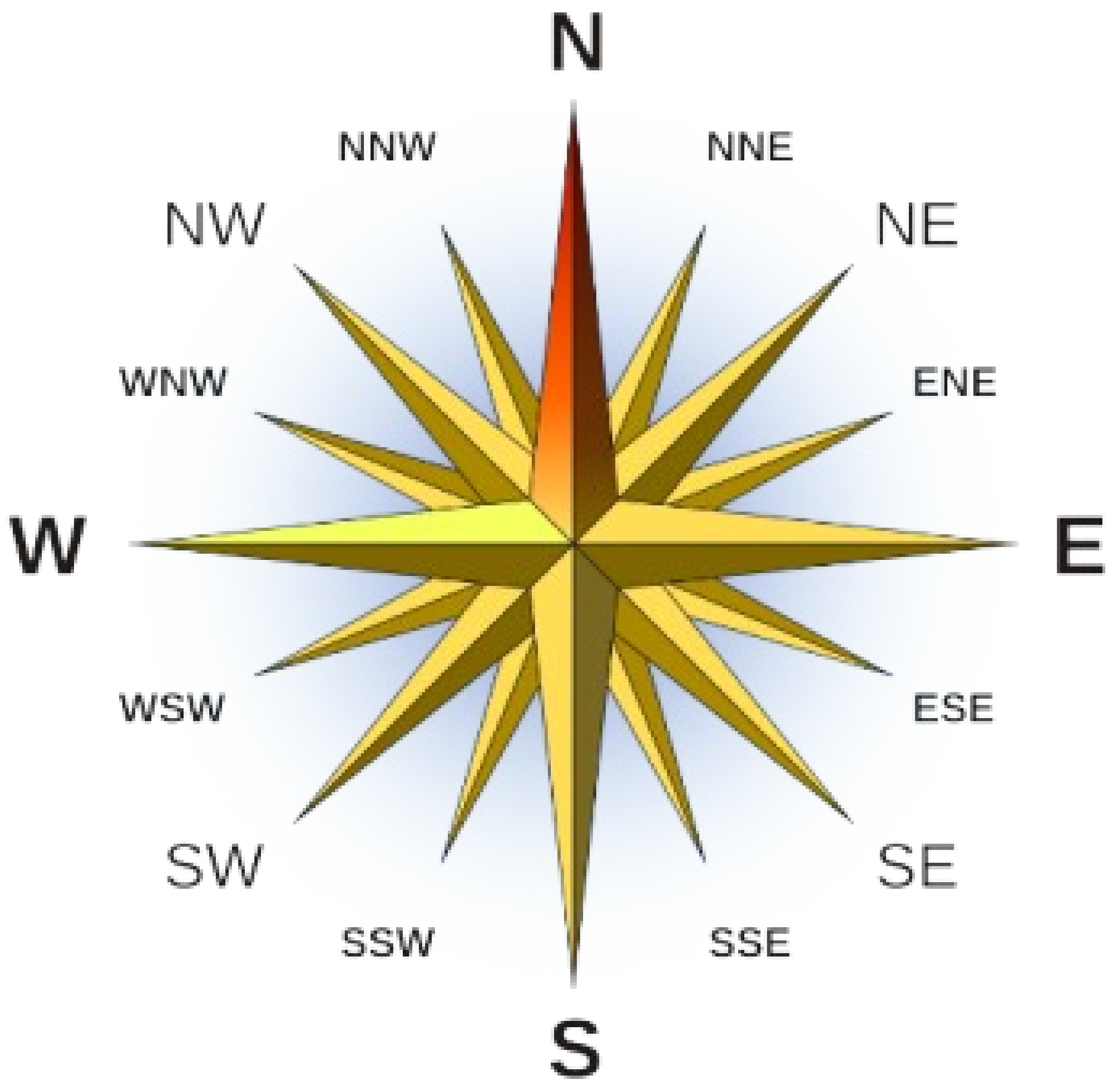
ToP (Top of Pod)

Level 1 (spine 1)

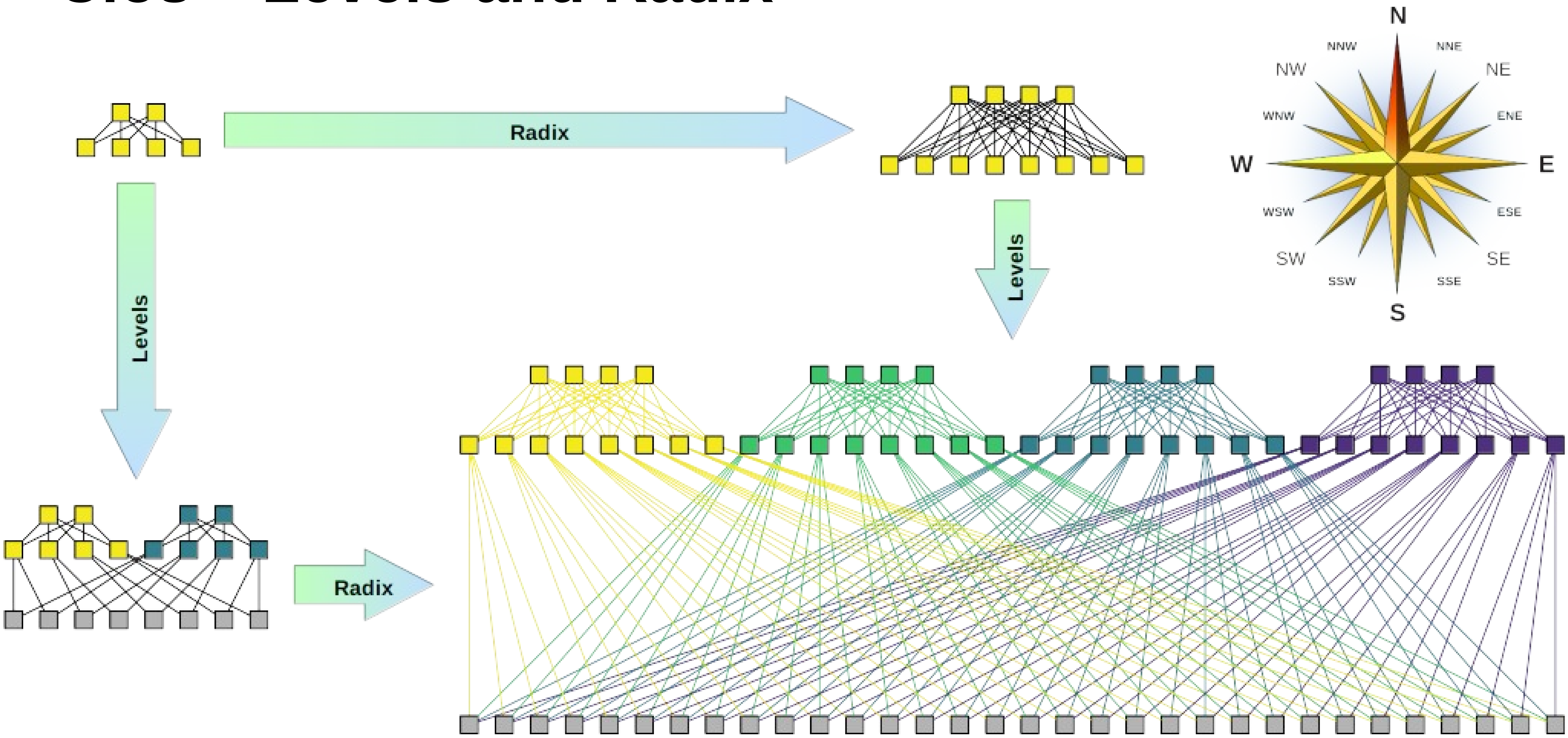
Level 0 (leaf/ToR)

Pod

Levels



Clos – Levels and Radix



Why Aggregate

- Smaller routing and forwarding state
- Potentially smaller blast radius

Aggregate Multipath Routes

- Default or DCN aggregate
    - Points north
    - Propagates southward from the ToF
- Smaller aggregates
    - Appear when we do host multihoming / extra mesh between levels / top of PoD aggregates for redundancy (this also causes valley routing)
    - Direction depends on location in the DCN
    - Propagate north to the ToF, then reflected and propagate south

Why Disaggregate

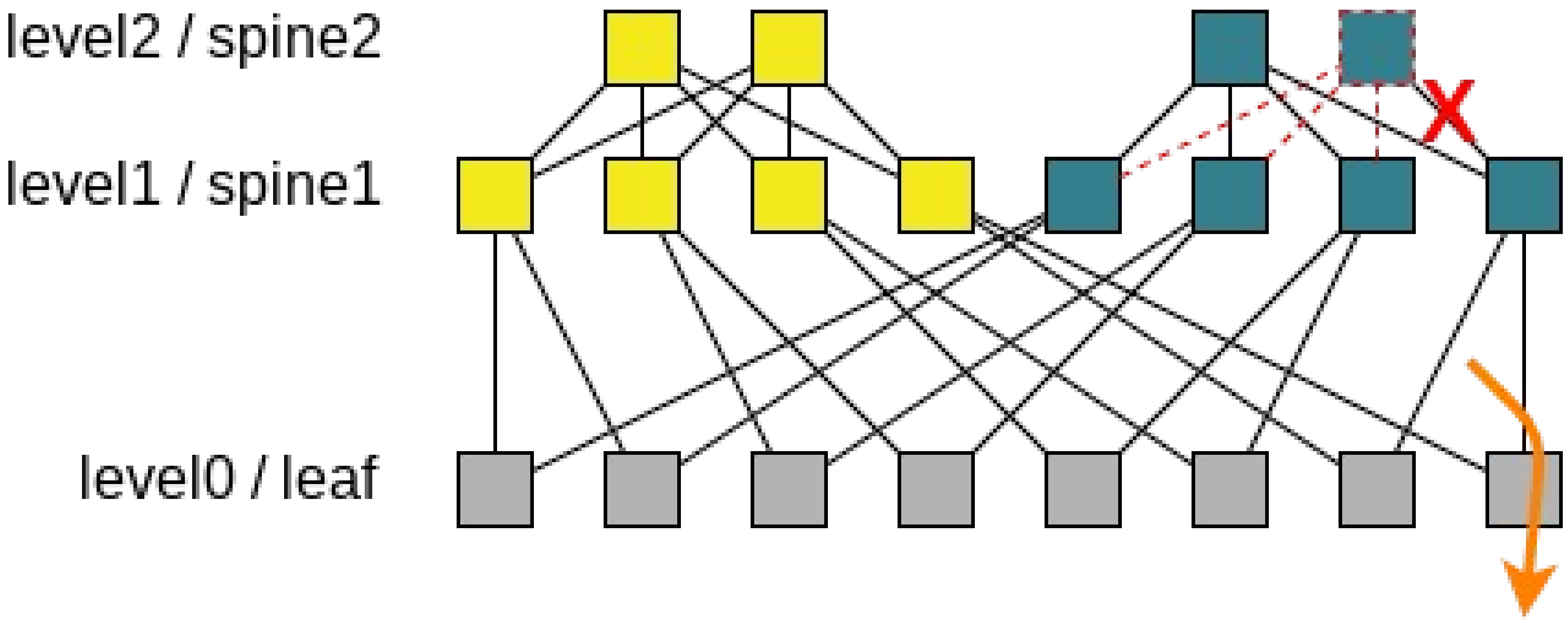
- Multipath + multiple destinations covered by aggregate + [remote] failures
  - some nodes originating (or pointed to) by aggregate may not have reachability to some destinations
  - some direct nexthops become invalid for some destinations
- Clos: single path from ToF node to leaf
  - Any failures along that path make part of topology (ToF or even lower level spine nodes) invalid as nexthops for prefixes behind that leaf
  - can't use northward default anymore
  - in Clos going up/north narrows available part of topology - e.g. once plane selected can't go to another plane
- Note: "just always disaggregate everything" may be an option
  - Makes worst case scenario normal
  - Forwarding state can be a problem

Conditional Disaggregation

- What can't be decided based on local information:
  - Do we need to inject/propagate specifics because some other nodes on the same level don't have routes to some destinations?
  - Do we have max set of reachable destinations?
- All levels are disjoint - node doesn't know what info other nodes on the same level have
- Easy to handle if we start with completely disaggregated
  - Worst case state all the time
- Or full set of destinations is known in advance and distributed to all nodes
  - Usually not feasible

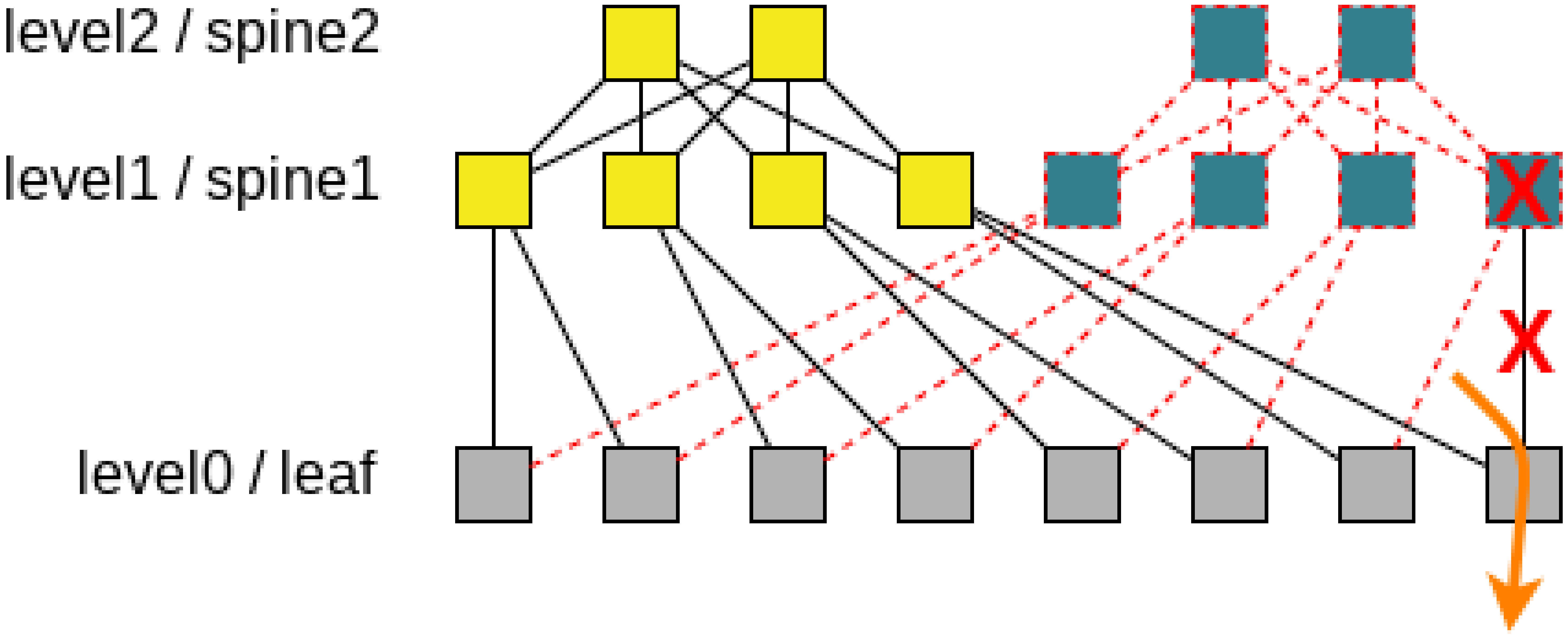
Failure on Level 2

- Can't use default on some spine 1s



Failure on Level 1

- Can't use default on all leaves



More on Failures

- Situation becomes more interesting with more spine levels
  - Need to disaggregate below level where remote failure happened => the lower level where failure happened the larger blast radius
  - When considering failures south links belong to same level as node and north links belong to the next level
- Most deployments don't need > 2 spine levels

Failures and Forwarding State

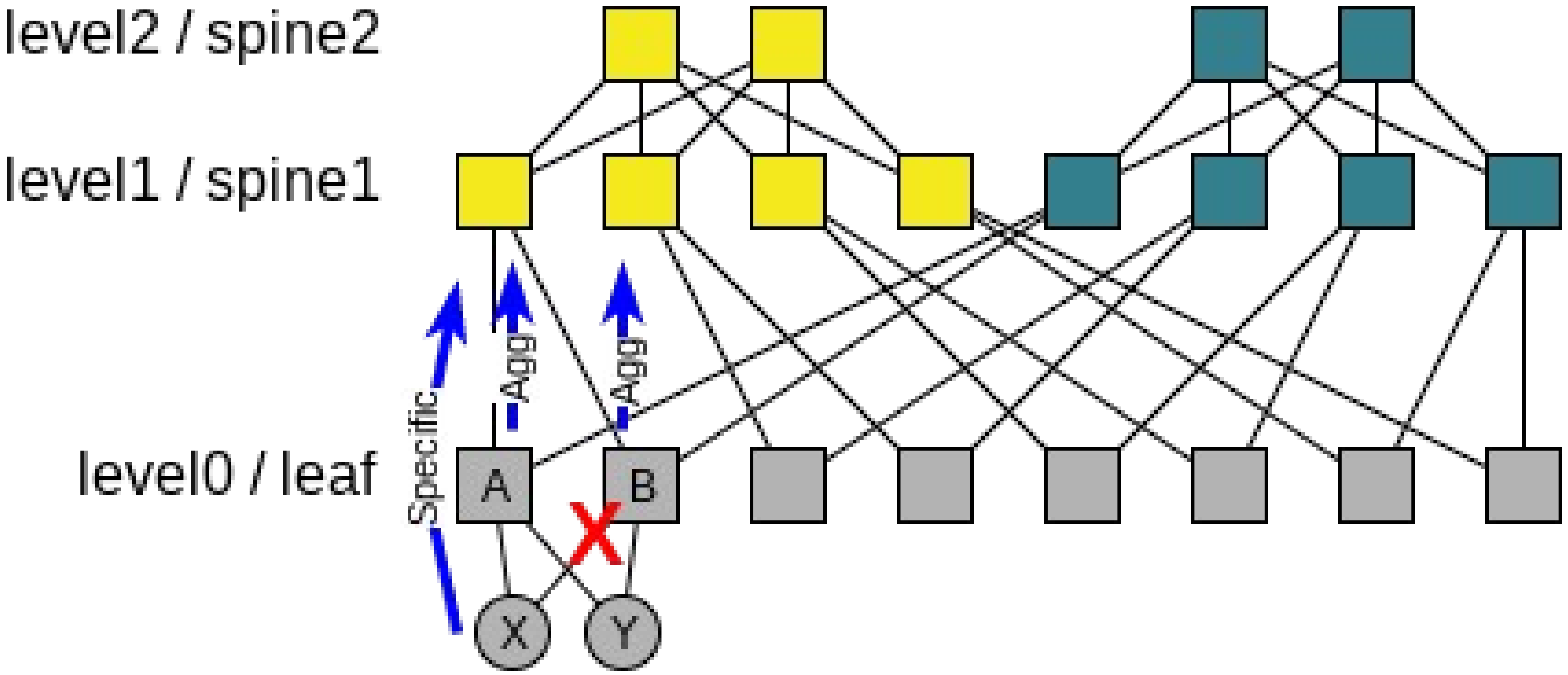
- ECMP forwarding state is often a scarce resource
  - Limited table sizes - ECMP NH groups and ECMP NHs
  - Grows fast - #NH_NHS = #NH_GROUPS * #ECMP_WIDTH
  - North ECMP width 32 to 64 is common.
- ECMP scaling is most problematic on intermediate (non-ToF) spines:
  - have north routes & high north radix
- Leaves/ToRs normally have much smaller north radix
  - with narrow ECMP (normally leaves/ToRs) max #NH_GROUPS is limited by number of NH combinations = 2 ^ NORTH_RADIX
- IP routes with the same set of NHs normally share NH groups
- Each failure potentially introduces new route and new ECMP group

L3 Host Multihoming

- Pair of leaves originate the same aggregate prefix(es)
  - Until something fails
- Leaf doesn't have enough local information and can't figure out:
  - if host is dead or just lost one of the uplinks => host needs to decide
  - if another leaf in pair is alive and injecting the same aggregates
- Assuming valley free routing
  - No traffic reflection via top of Pod - it introduces its own corner cases - can choose between blackholes and loops
- Massive transient specific route injection can be a serious problem
  - e.g. DCN or PoD power up

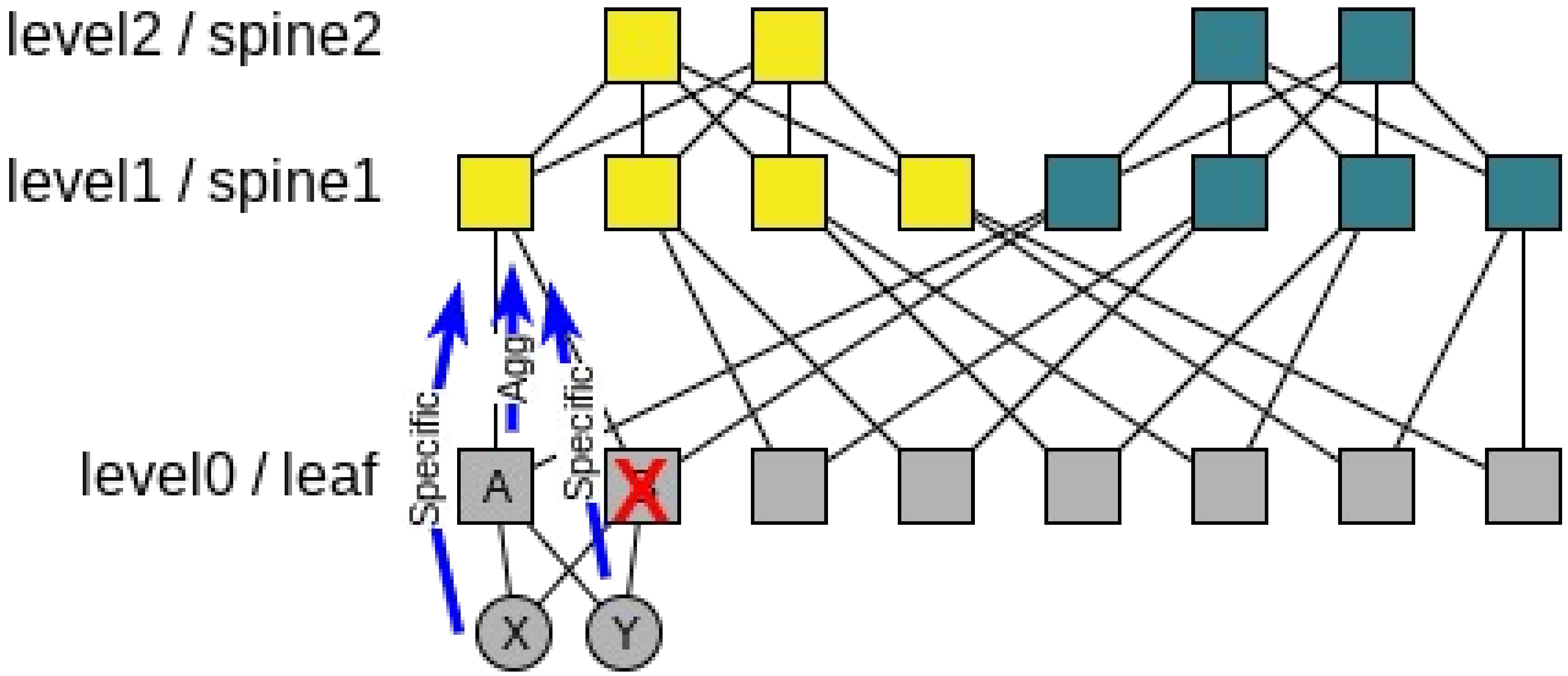
Host uplink failure

- Leaf A doesn't know that host X is unreachable via B
- Host can decide and inject specific route



Leaf failure

- Leaf B is down
- All attached hosts injects specifics
- Leaf A has no way to know that B is down and it's Ok to suppress specifics



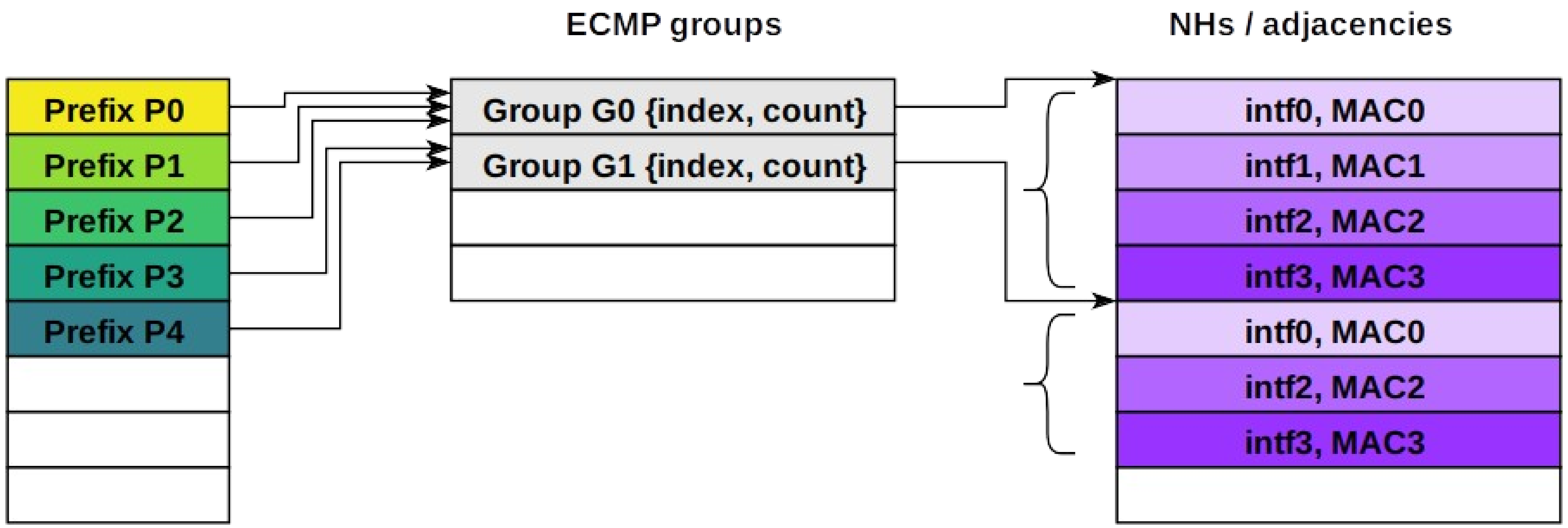
Backup Slides

Dense topologies and ECMP: MPLS vs IP

- IP: can share NH/rewrite entries for different destinations
- MPLS: normally need unique entry per {ingress label, egress interface} tuple
  - but with SR-style global labels optimizations are possible and some chipsets can do that

IP ECMP

LPM lookup with ECMP result



ECMP groups

NHs / adjacencies

Prefix P0

Prefix P1

Prefix P2

Prefix P3

Prefix P4

Group G0 {index, count}

Group G1 {index, count}

intf0, MAC0

intf1, MAC1

intf2, MAC2

intf3, MAC3

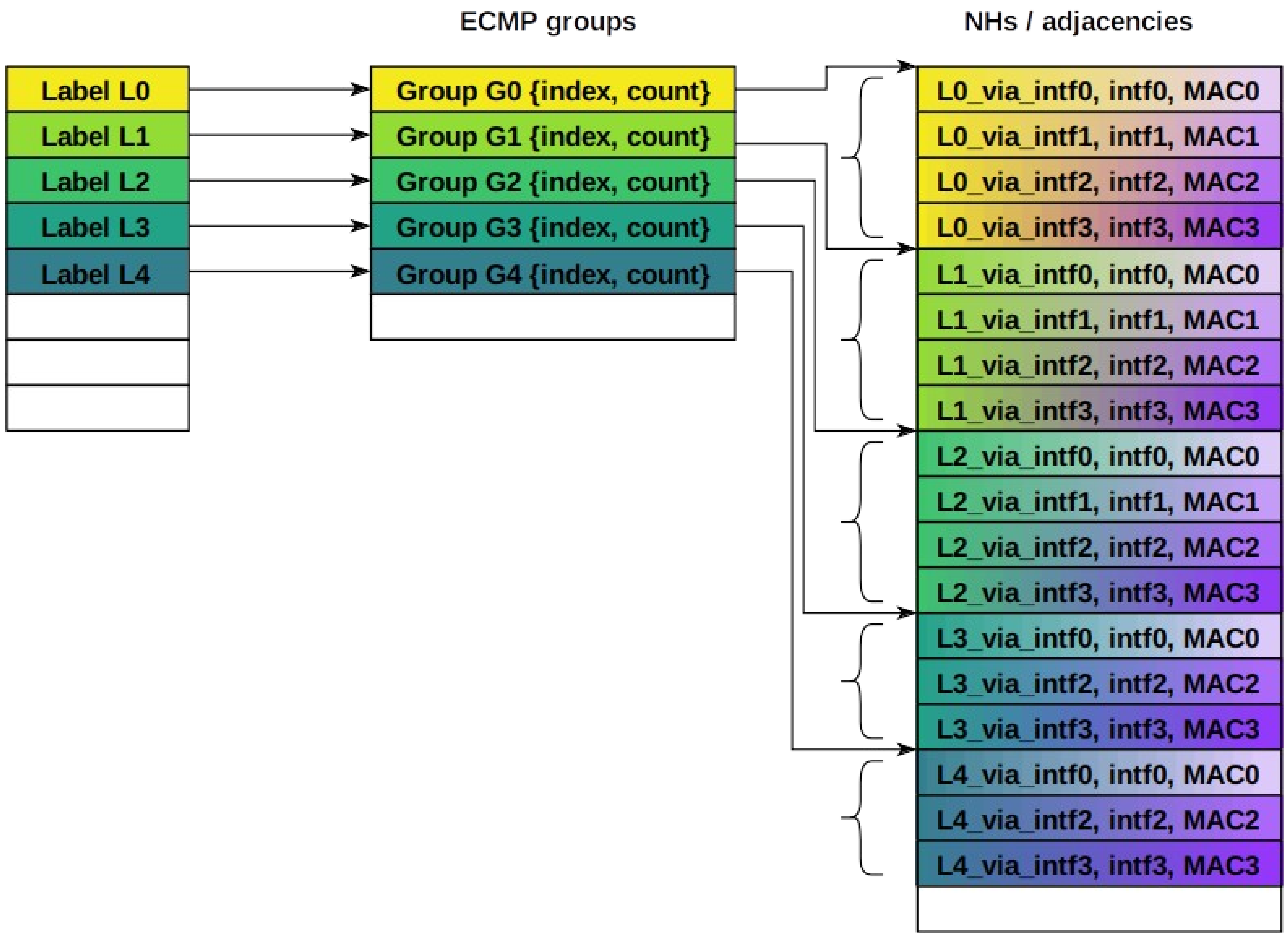
intf0, MAC0

intf2, MAC2

intf3, MAC3

MPLS ECMP

LFIB lookup with ECMP result



ECMP groups

NHs / adjacencies

Label L0

Label L1

Label L2

Label L3

Label L4

Group G0 {index, count}

Group G1 {index, count}

Group G2 {index, count}

Group G3 {index, count}

Group G4 {index, count}

L0_via_intf0, intf0, MAC0

L0_via_intf1, intf1, MAC1

L0_via_intf2, intf2, MAC2

L0_via_intf3, intf3, MAC3

L1_via_intf0, intf0, MAC0

L1_via_intf1, intf1, MAC1

L1_via_intf2, intf2, MAC2

L1_via_intf3, intf3, MAC3

L2_via_intf0, intf0, MAC0

L2_via_intf1, intf1, MAC1

L2_via_intf2, intf2, MAC2

L2_via_intf3, intf3, MAC3

L3_via_intf0, intf0, MAC0

L3_via_intf2, intf2, MAC2

L3_via_intf3, intf3, MAC3

L4_via_intf0, intf0, MAC0

L4_via_intf2, intf2, MAC2

L4_via_intf3, intf3, MAC3