Routing Protocol in Industrial IoT

Pascal Thubert

Rennes

October 25th, 2023

Part 1: RPL Design

Scaling to Pervasive IoT

1000*scale => No leak in Internet

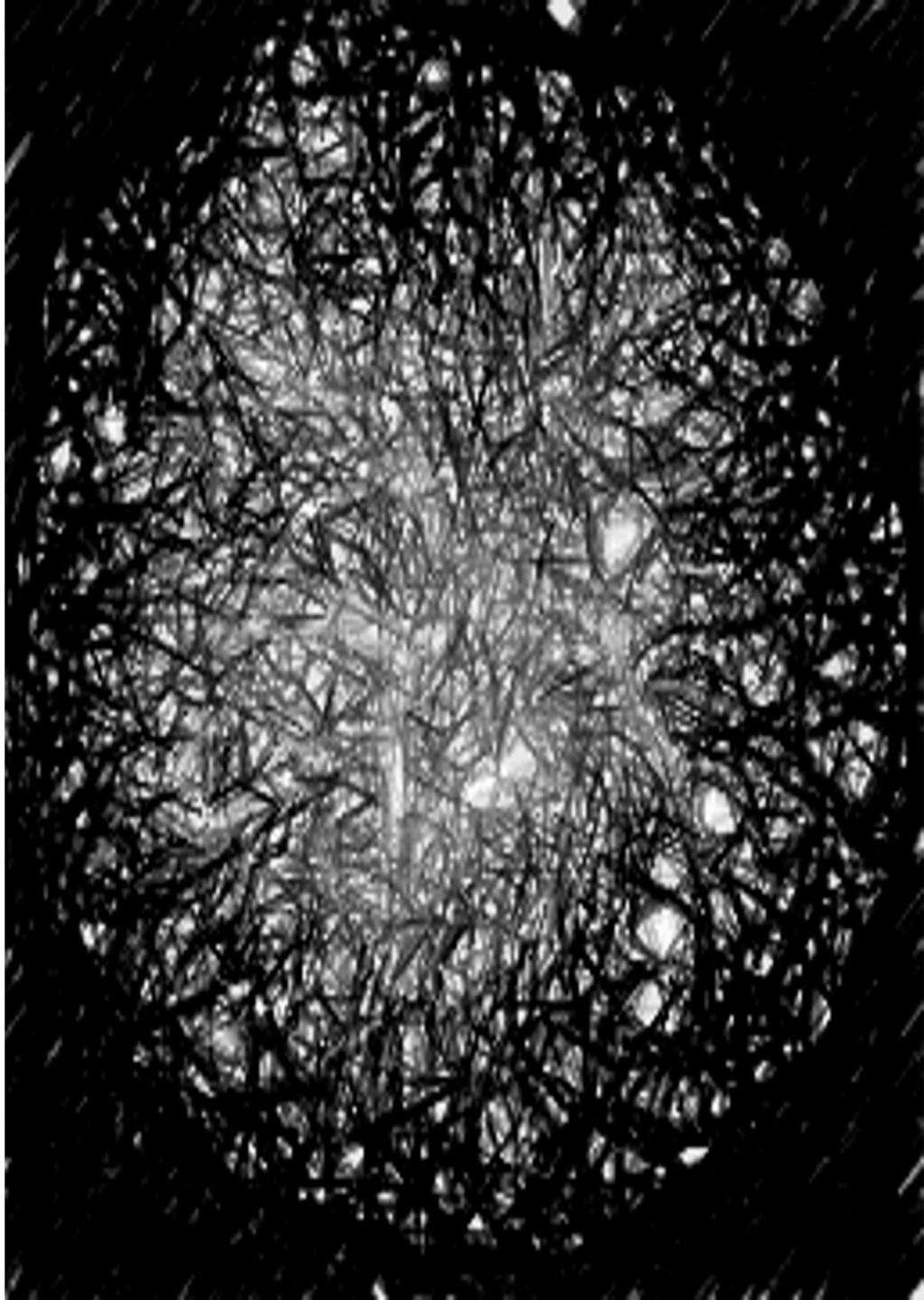
=> Opaque operations

Reachability => Radio (IEEE)

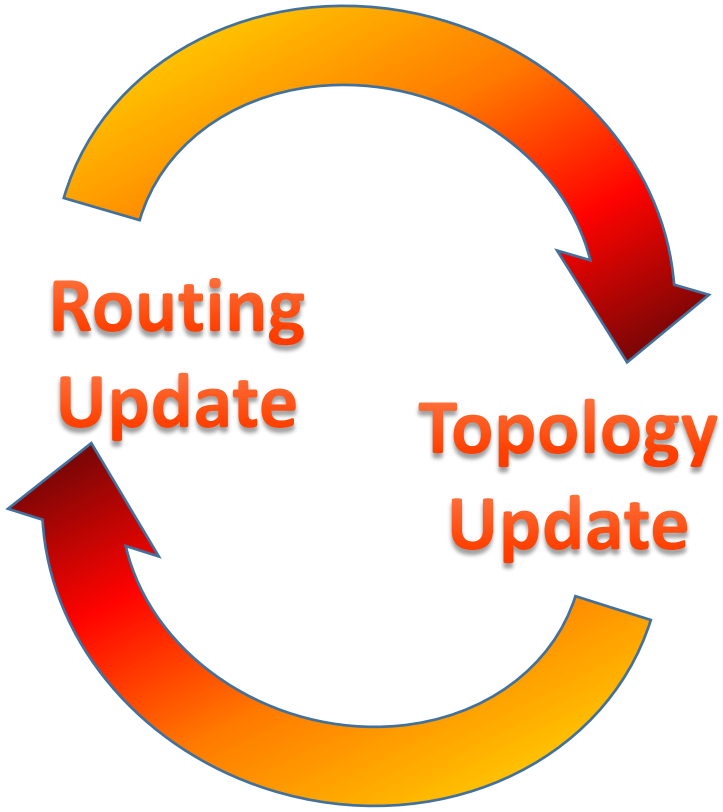
Addressing => IPv6 (IETF)

Density => spatial reuse

=> Routing



Dynamic topologies



No preexisting physical topology

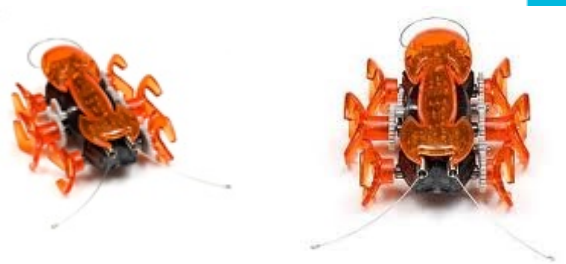
- Can be computed by a mesh under protocol, but…
- Else Routing must infer its topology

Movement

- natural and unescapable
- Yet difficult to predict or detect

quick and knowledgeable => expensive

Peer selection



Potentially **Large** Peer Set

Highly **Variable** Capabilities

Selection Per Objective

{

Metrics (e.g. RSSI, ETX…)

L3 Reachability (::/0, …)

Constraints (Power …)

Constrained Objects

Smart object are usually

- Small & Numerous
- « sensor Dust »

Battery is critical

- Deep Sleep
- Limited memory
- Small CPU

Savings are REQUIRED

⇒ Control plane (minimize state)

⇒ Data plane (minimize control messages)

Fuzzy (non-deterministic) links

Neither transit nor P2P

More like a changing NBMA

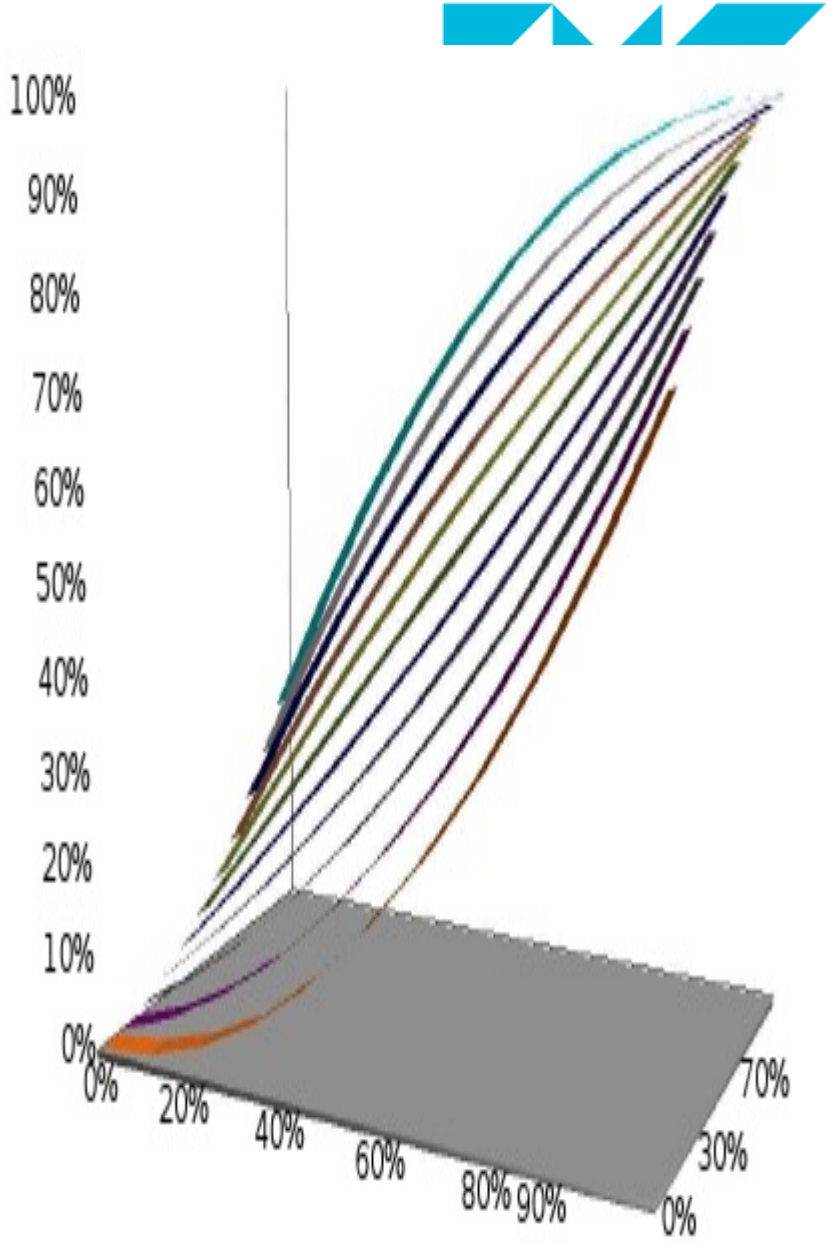
- a new paradigm for routing

Changing metrics

- (tons of them!)
- (but no classical cost!)

Inefficient flooding

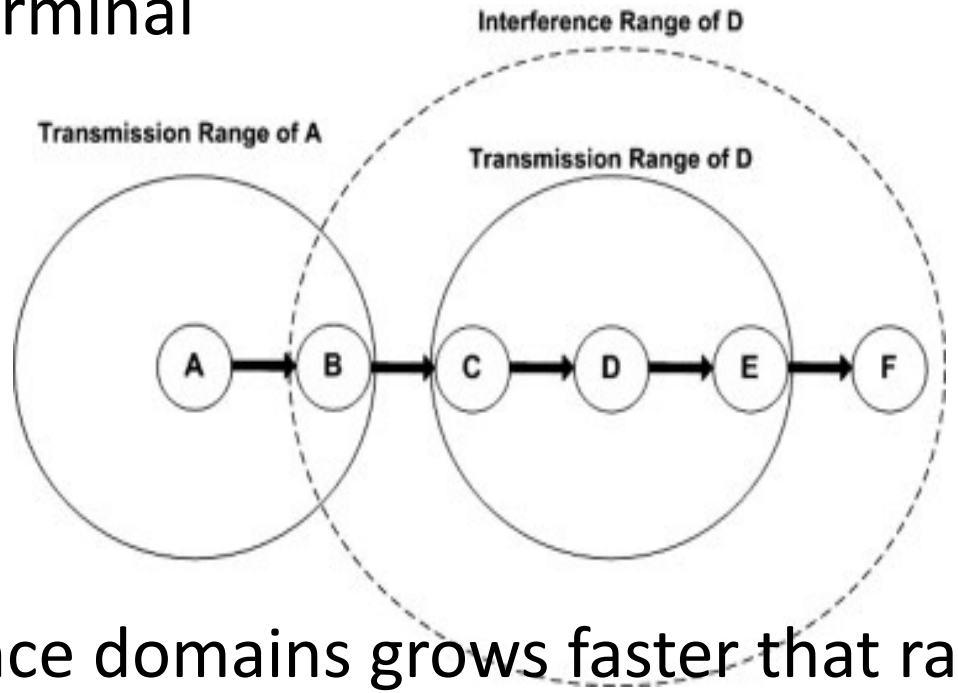
- Self interfering



Routing Concepts

Routing for Spatial Reuse

- Hidden terminal



- Interference domains grows faster that range
- Density => low power => multihop => routing

Centralized vs. Distributed Routing
(RPL now has both with the new DAO Projection)

Centralized: A controller sets up the routes

Pros

God's view, enables global optimization

Elastic resource management / NFV

Traffic Engineering, may compute per flow paths

Non Equal Cost Multipath, Replication & Elimination

Cons

Delays learning about breakage

Delays fixing breakage

NP complete optimization => limits scalability

Distributed: A routing protocol sets up the routes

Pros

Since ARPANET, enables high resilience

Different IGPs for different needs

Installed base

Cons

Microloops

Tends to build crowded avenues, wastes bandwidth

Same costs for all, NECM difficult, manual TE

Need additions for reroute / fast reroute

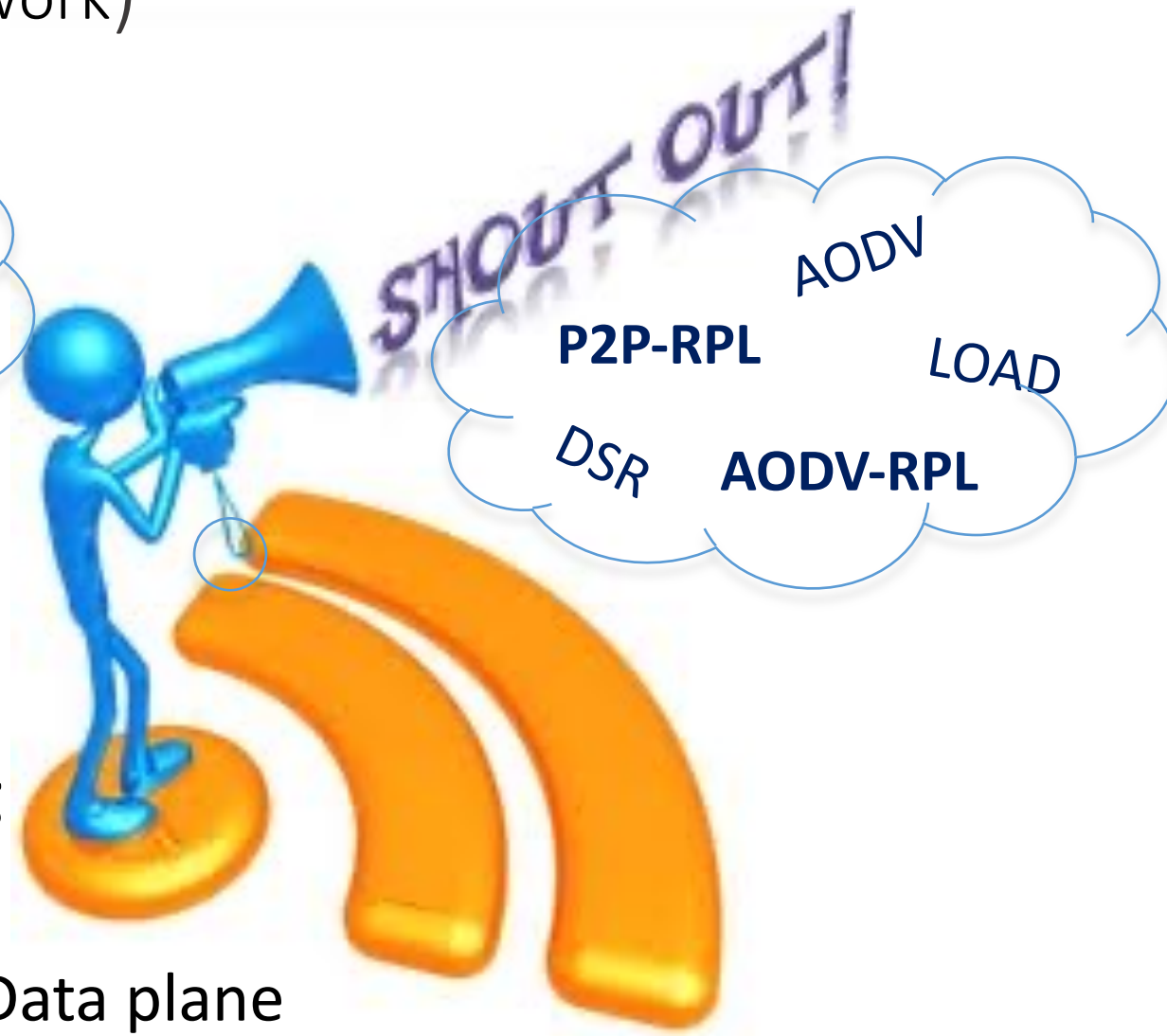
Proactive vs. Reactive AKA on-demand
(RPL now has both with the new AODV work)

Aka

OSPF
OLSR
RPL
RIP
BABEL
EIGRP
IS-IS

Stateful vs.

On-demand routing



Note: on-demand breaks control vs. Data plane separation

Stateful vs. Source-Routing
(RPL has both modes)

Stateful: routing decision at every hop

Pros

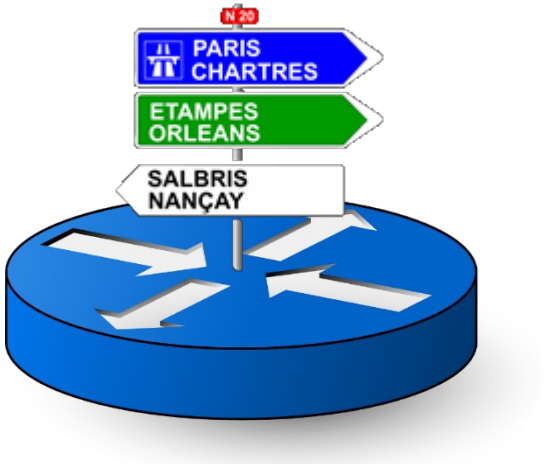
Resilient (DARPA), autonomic

Cons

Requires routing state in every router

Tends to concentrate flows

Routing Loops



Source Routing: The path is in the packet

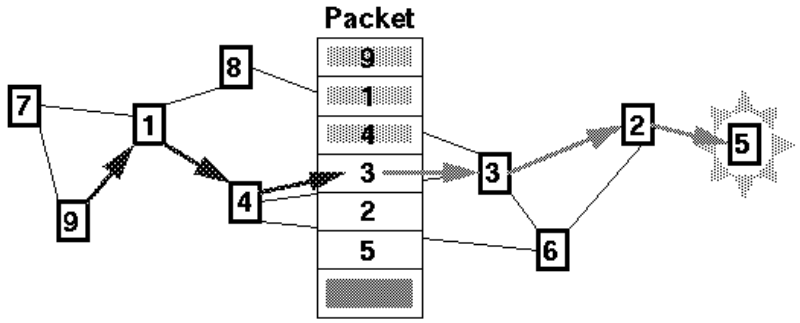
Pros

Saves state in routers

Enables per flow routing (segment routing)

Cons

Larger packets / Source Route Header (SRH)



Link State aka Dijkstra aka SPF vs. Distance Vector aka Bellman-Ford (RPL is DV)

LS requires full state and convergence

- Every node draws a same graph
- Then run the shortest path first algorithm to get to all other nodes
- Then associates node with reachable destinations

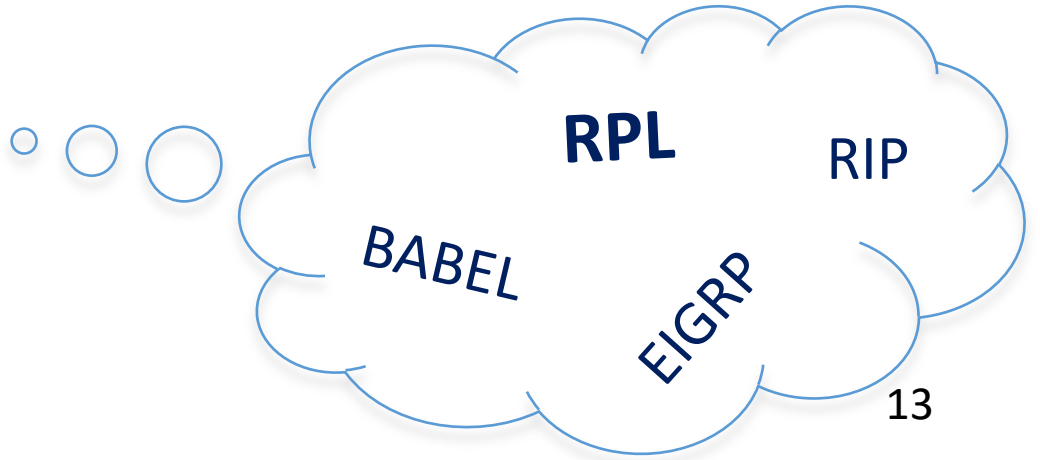
LS can be very quiet on stable topologies

DV learns costs to destination asynchronously per destination

- Nodes advertise their distance to a destination
- Recursively other nodes learn their best successor
- and compute their own cost

DV hides topological complexities and changes

DV enables multiple NECM feasible successors



Route stretch vs Shortest Path
(RPL stretches to save)

Most protocols are Optimized Routing Approach (ORA) and span advertisements for any change to maintain a shortest path to all

Routing overhead can be reduced if stretch is allowed: this is the Least Overhead Routing Approach (LORA) approach

=> (Nothing to do with the LoRa LPWAN protocol!)

The base RPL is an **anisotropic** form of LORA: it optimizes routes from and to a Root and stretches P2P

Routing Trees
(RPL has a preferred parent tree)

Most classical routing structure

Typically what Internet Gateway Protocols (IGPs) Build or each reachable destination

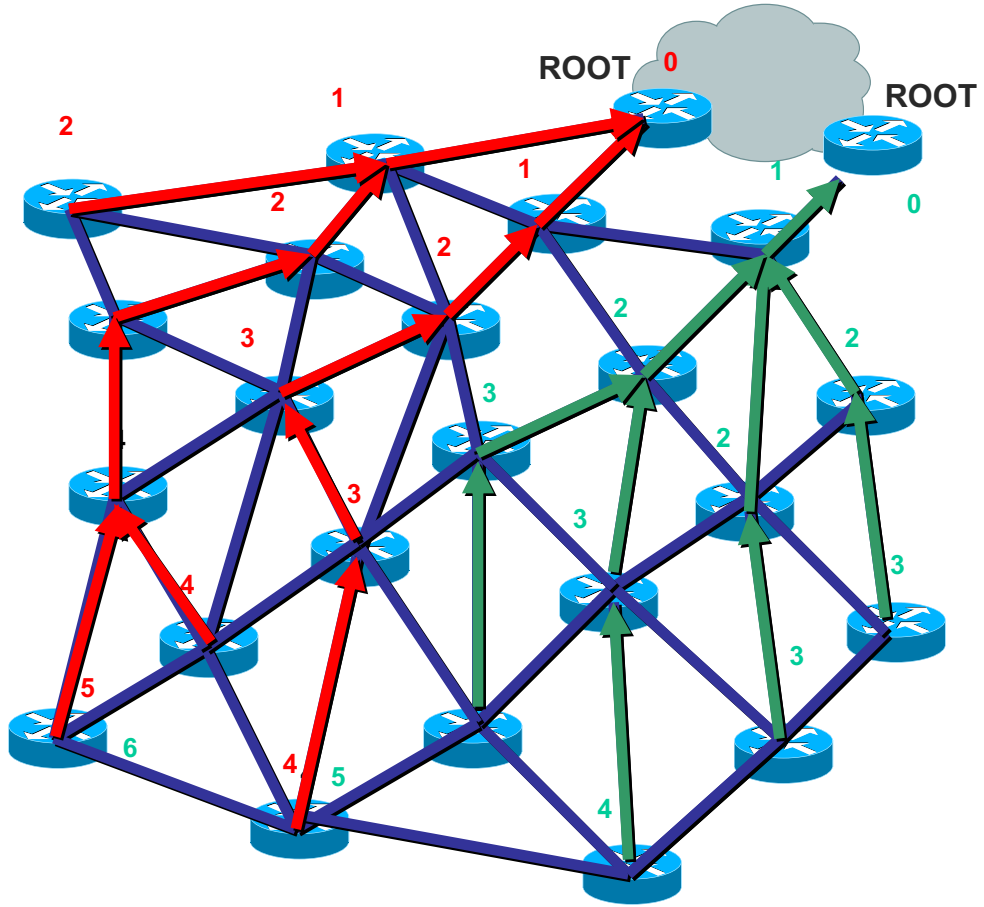
Only thing Link State builds, though Distance Vector has feasible successors

Must be reconstructed upon connectivity loss, service is interrupted

FRR techniques must be added (MRT, LFA with SR …) on top

No inner load balancing capabilities

Walkable structure (e.g. depth first)



DAGs and DODAGs
(RPL builds DODAGs)

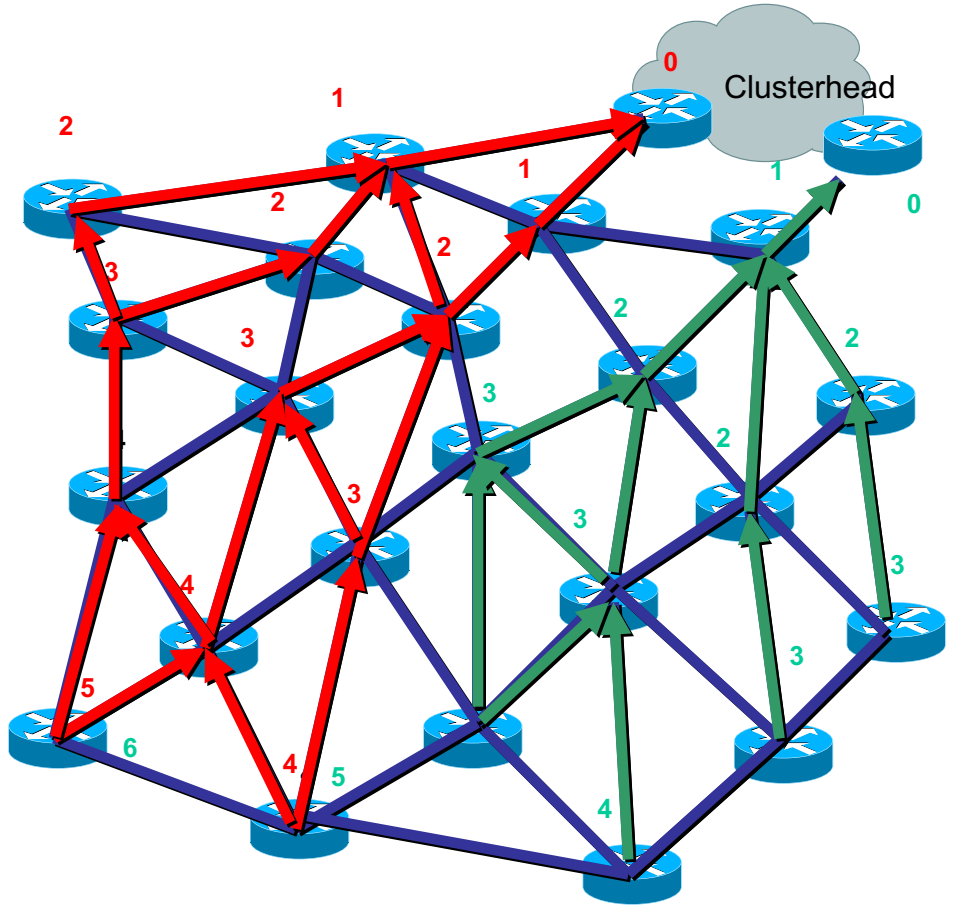
In the context of routing, a Directed Acyclic Graph (DAG) is formed by a collection of vertices (nodes) and edges (links).

Each edge connecting one node to another (directed) in such a way that it is not possible to start at Node X and follow a directed path that cycles back to Node X (acyclic).

Greedy => Not all nodes have 2 solutions even in biconnected networks

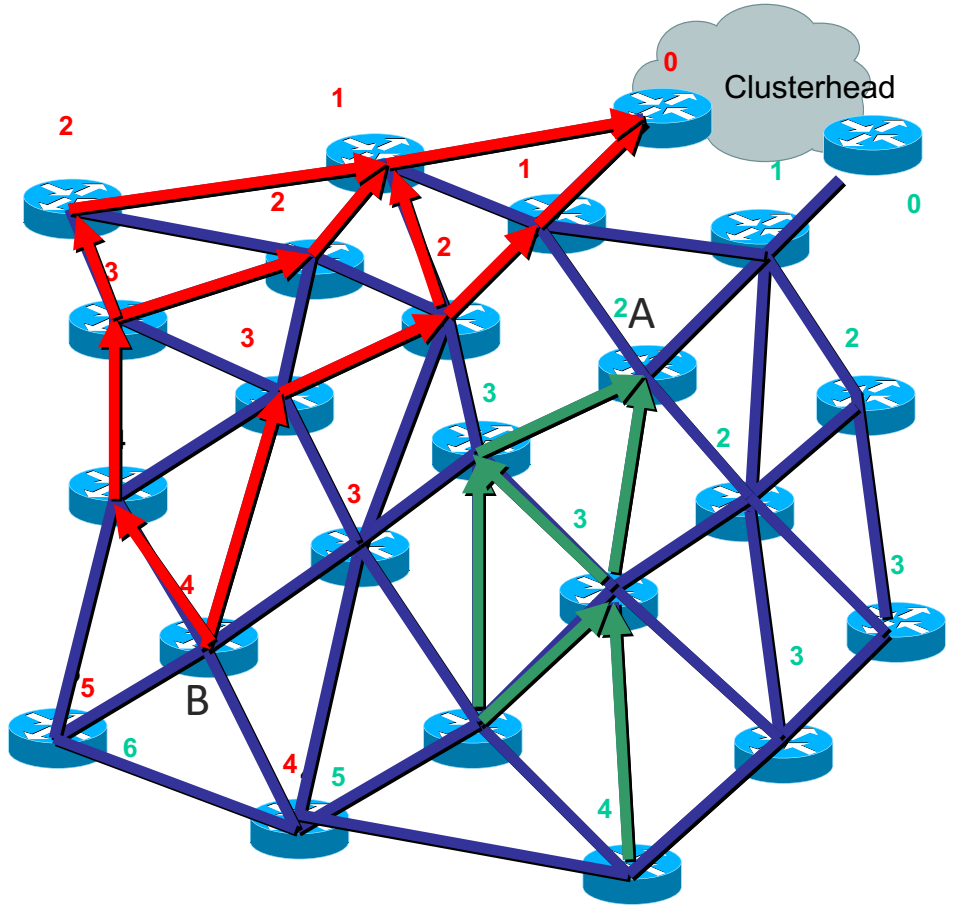
A Destination Oriented DAG (DODAG) is a DAG that comprises a single root node.

Here a DAG that is partitioned in 2 DODAG



SubDAG, and Fanout DAG

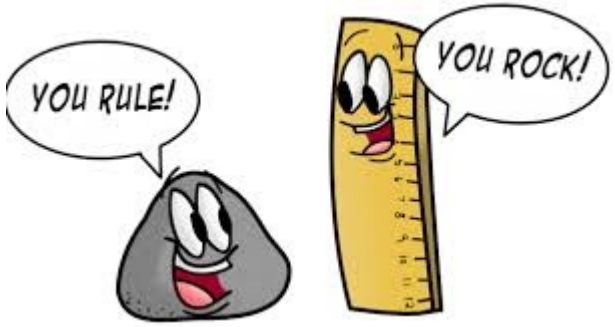
- In Green: A's subDAG.
  - Impacted if A's connectivity is broken
  - Domain for routing recovery
- In Red: B's fanout DAG
  - (or reverse subDAG)
  - Potential SPAN on B's DAO
  - Thus potential return paths
  - Fanout must be controlled to limit intermediate states



Instances and Objectives

RPL is objective driven

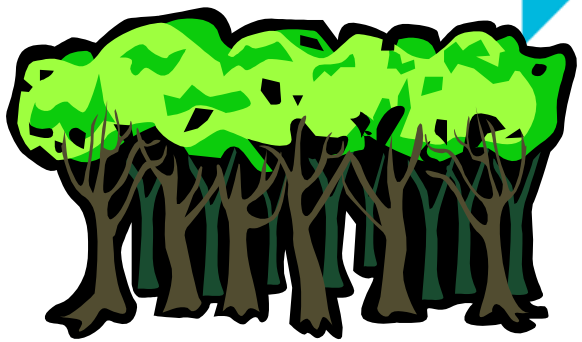
- e.g. reach a certain gateway, concept of a goal
- Instances are built to reach certain goals
- Instance ID tagged in packets



RPL is generic and extensible

- RFC 6550 is a core distance vector functionality
- Objective functions plug in to adapt to use case / need
- Objective functions enforce policies

Objective Function (OF)



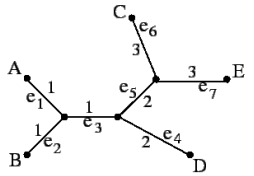
Extend the generic behavior

- For a specific need / use case

Used in parent selection

- Contraints
- Policies
- Metrics

} Position in the DAG



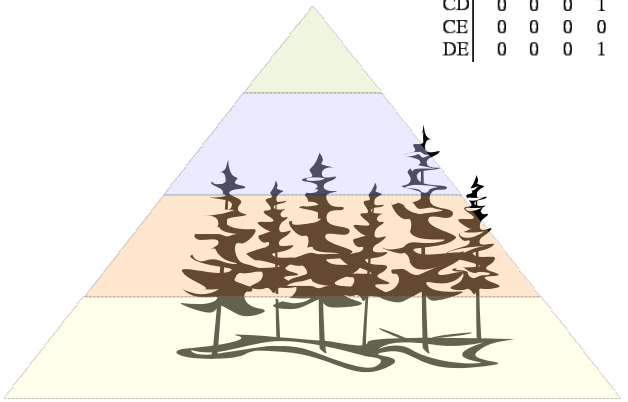
path-edge incidence matrix A x edge lengths **e** = path lengths **d**

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
AB	1	1	0	0	0	0	0
AC	1	0	1	0	1	1	0
AD	1	0	1	1	0	0	0
AE	1	0	1	0	1	0	1
BC	0	1	1	0	1	1	0
BD	0	1	1	1	0	0	0
BE	0	1	1	0	1	0	1
CD	0	0	0	1	1	1	0
CE	0	0	0	0	0	1	1
DE	0	0	0	1	1	0	1

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 7 \\ 4 \\ 7 \\ 7 \\ 4 \\ 7 \\ 7 \\ 6 \\ 7 \end{pmatrix}$$

Computes the Rank increment

- Based on hop metrics
- Do NOT use OF0 for adhoc radios!
- (OF 0 uses traditional weighted hop count)



Trickle: An Optimized Diffusion

Suppression of redundant copies

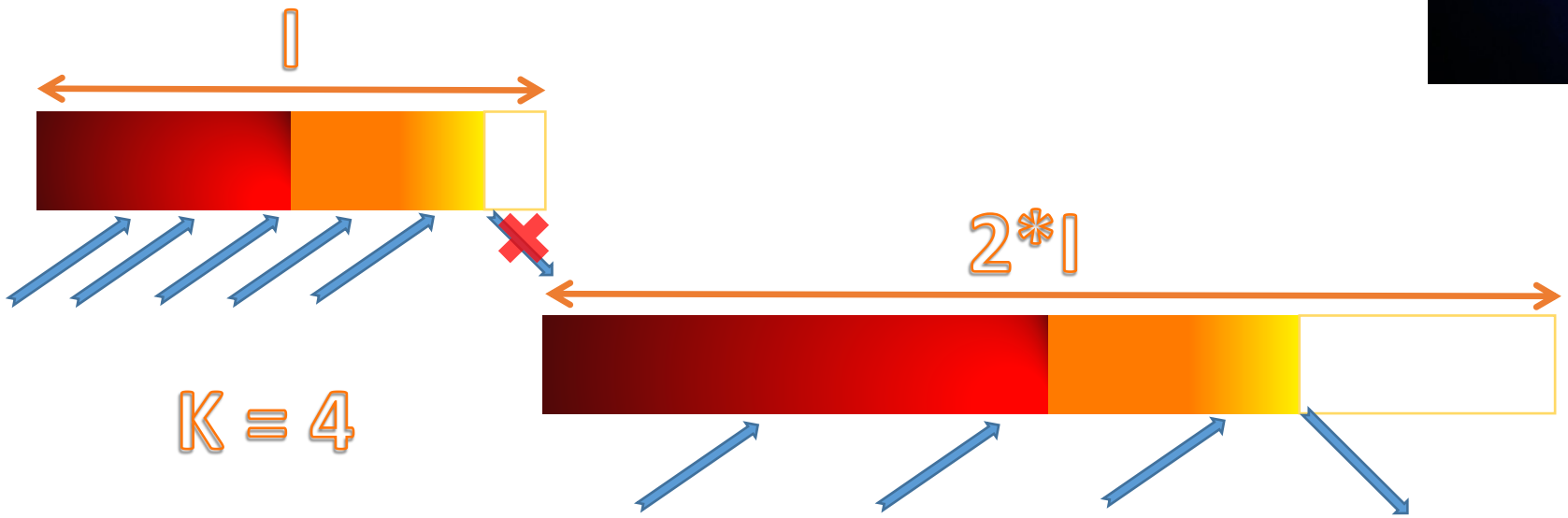
- Do not send copy if K copies received

Jitter for Collision Avoidance

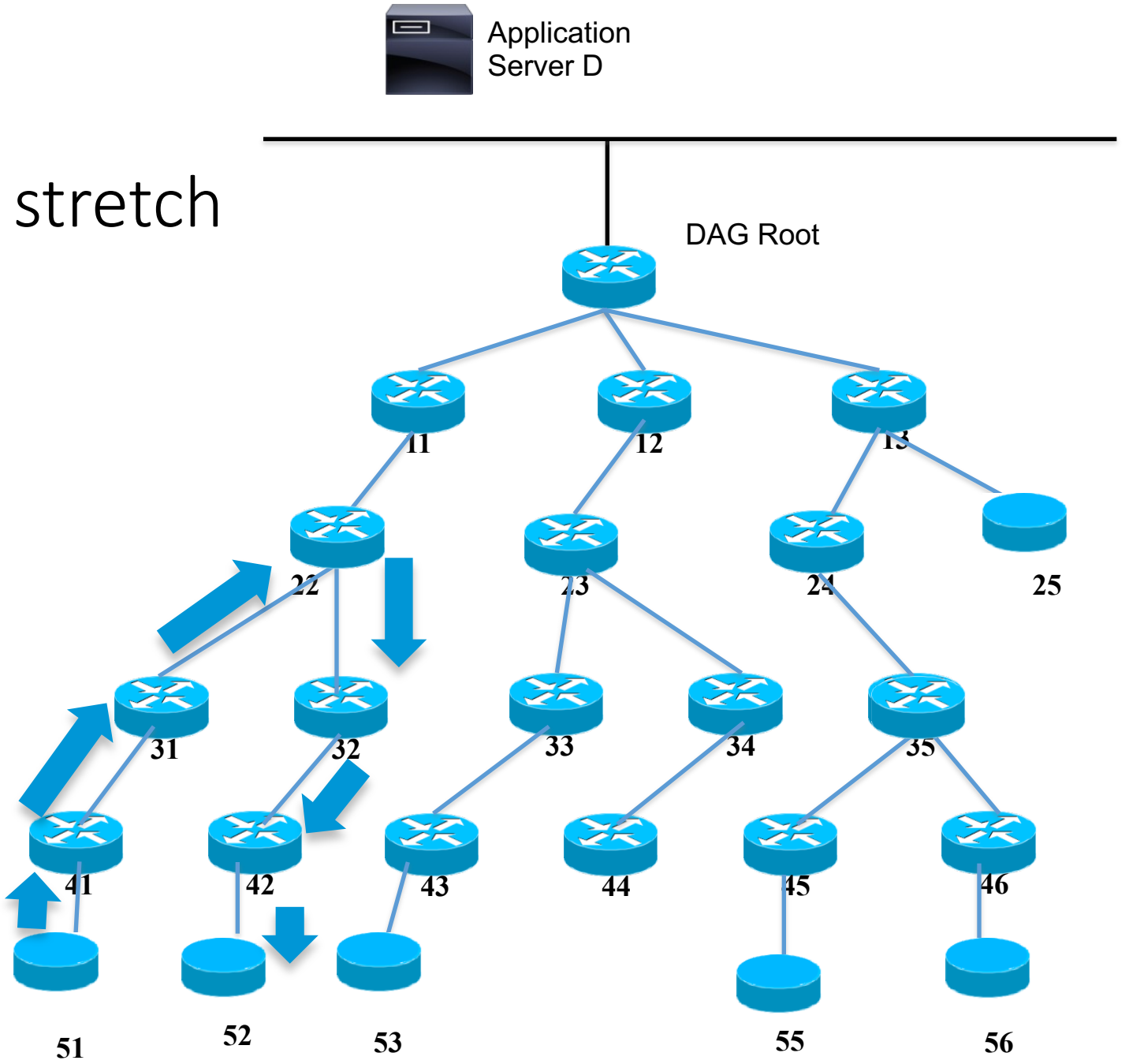
- First half is mute, second half is jittered

Exponential backoff

- Double I after period I, Reset I on inconsistency



Storing Mode Routing stretch



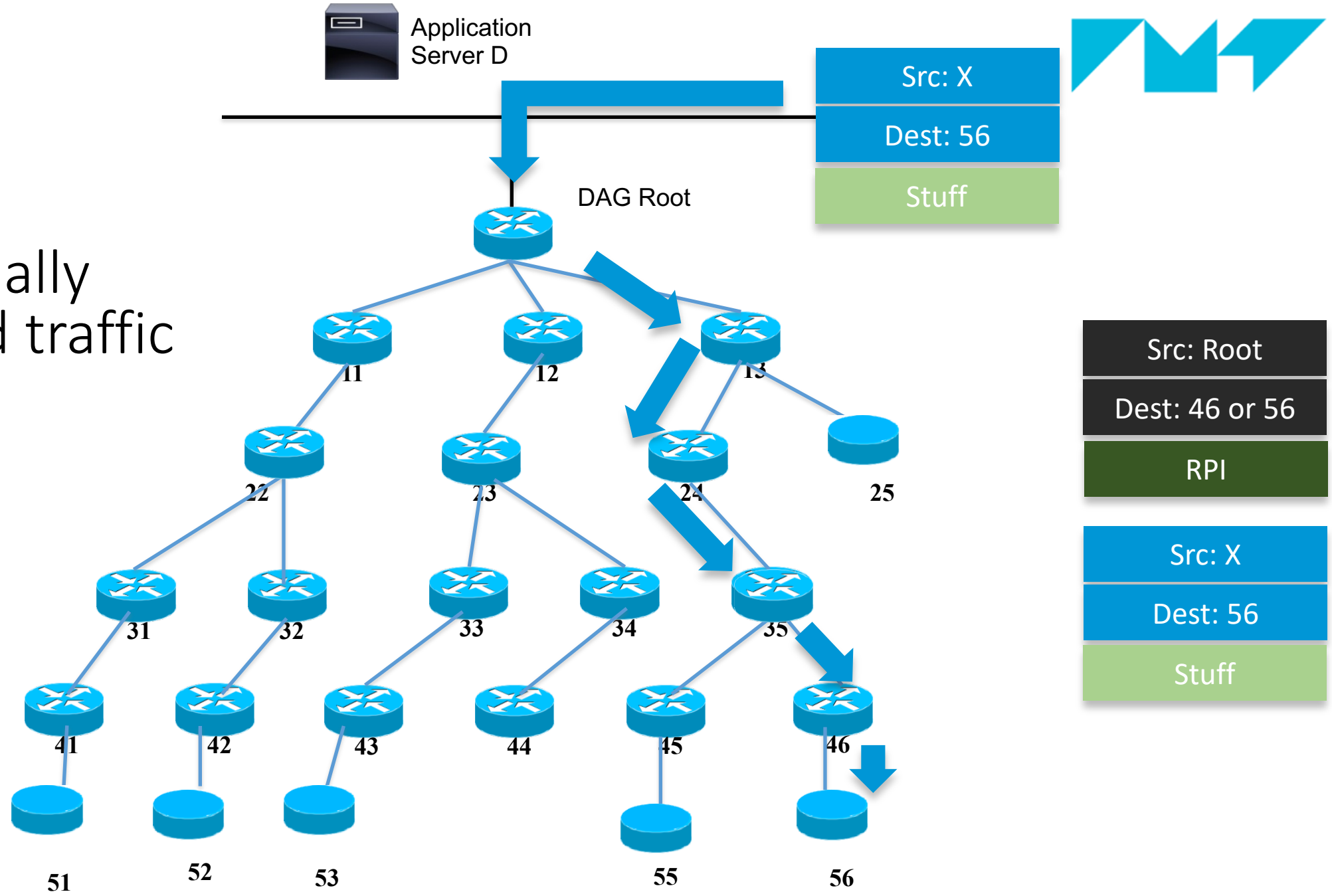
Src: 51

Dest: 52

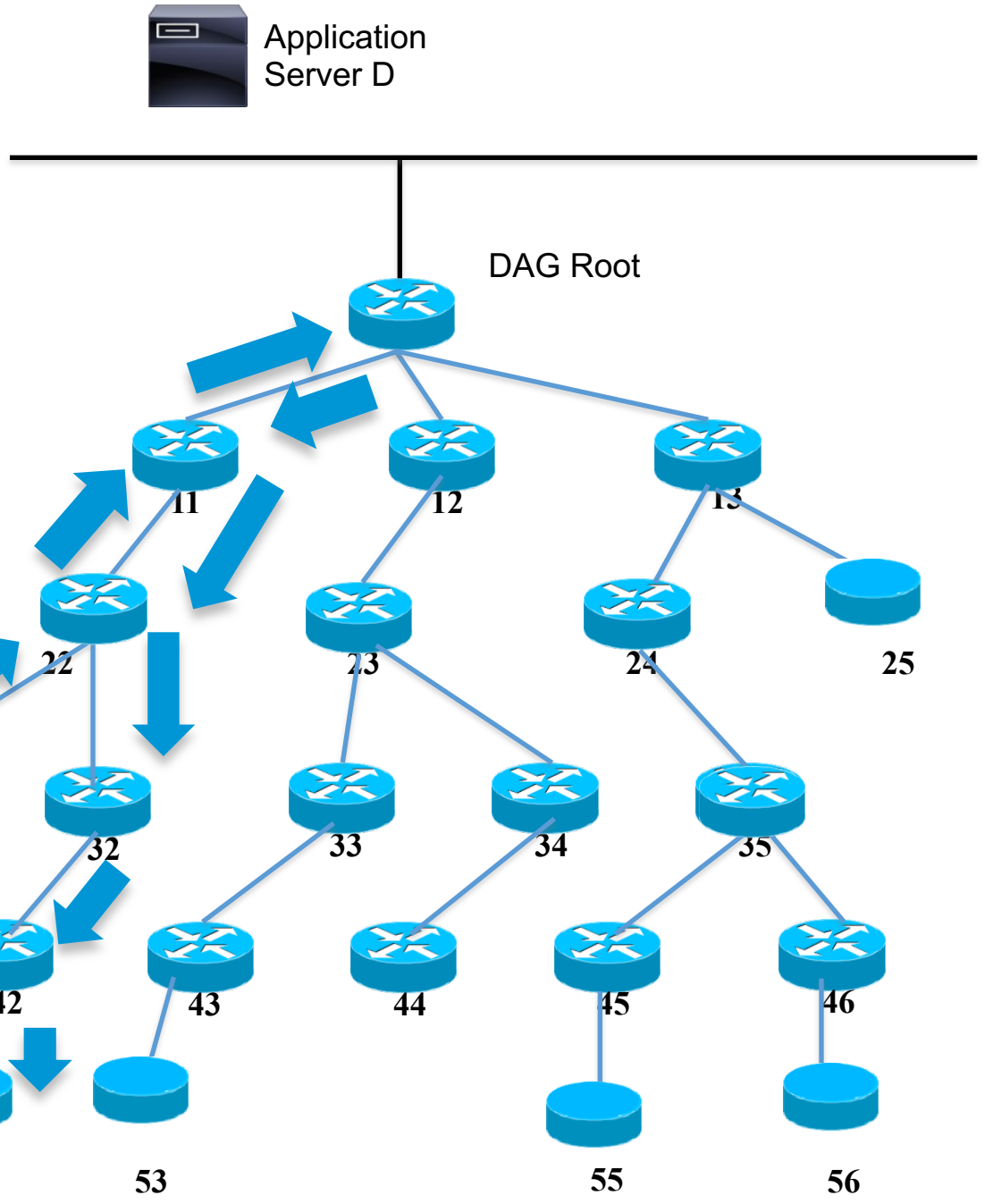
RPI

Stuff

Storing Mode Overhead For Externally Originated traffic



Non-Storing Routing stretch



Src: 51
Dest: Root
RPI
Src: 51
Dest: 52
Stuff

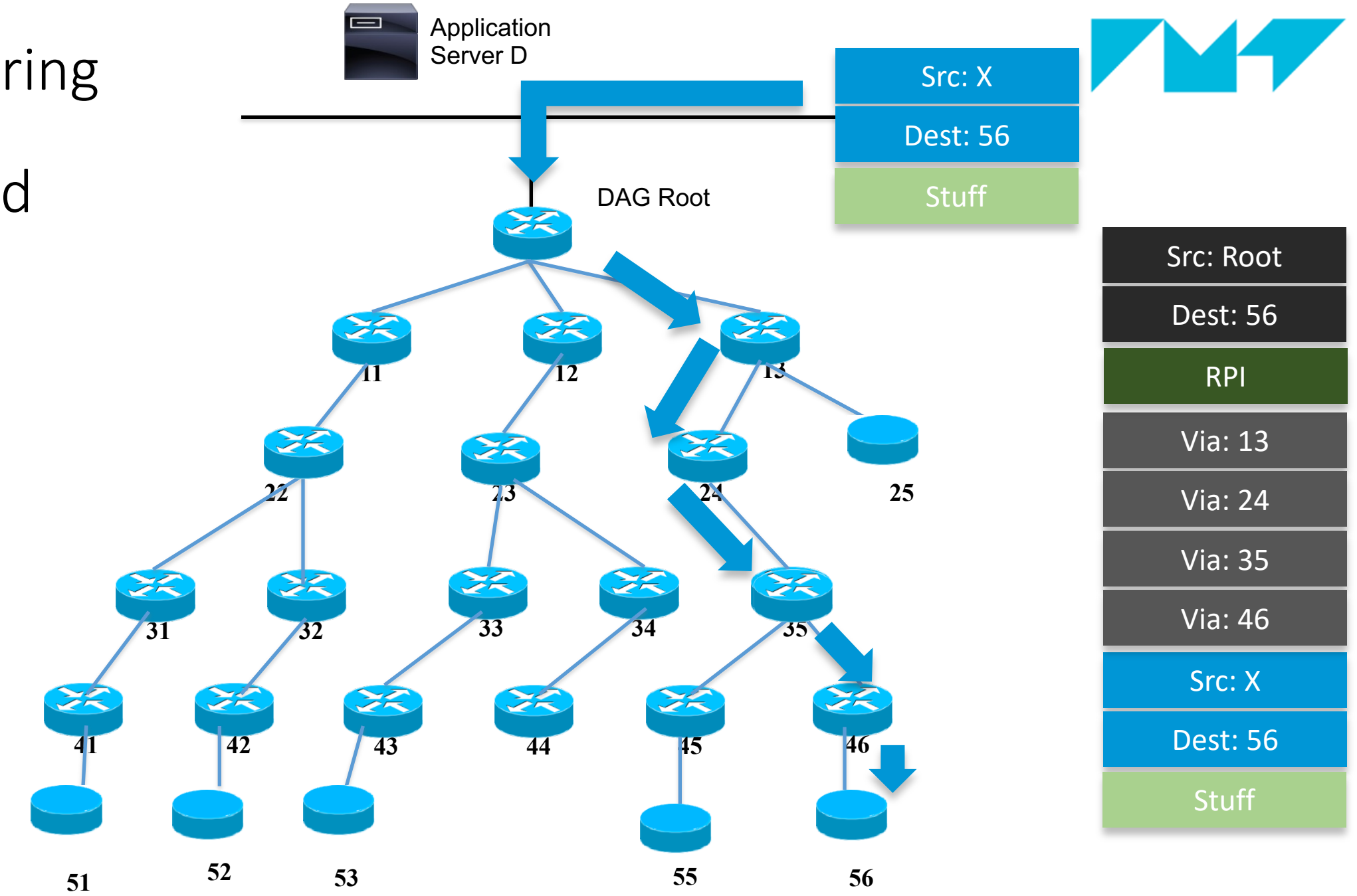
OR

Src: 51
Dest: 52
RPI
Stuff

Src: Root
Dest: 56
RPI
Via: 11
Via: 22
Via: 32
Via: 42

Src: X
Dest: 52
Stuff

Non-Storing Mode overhead



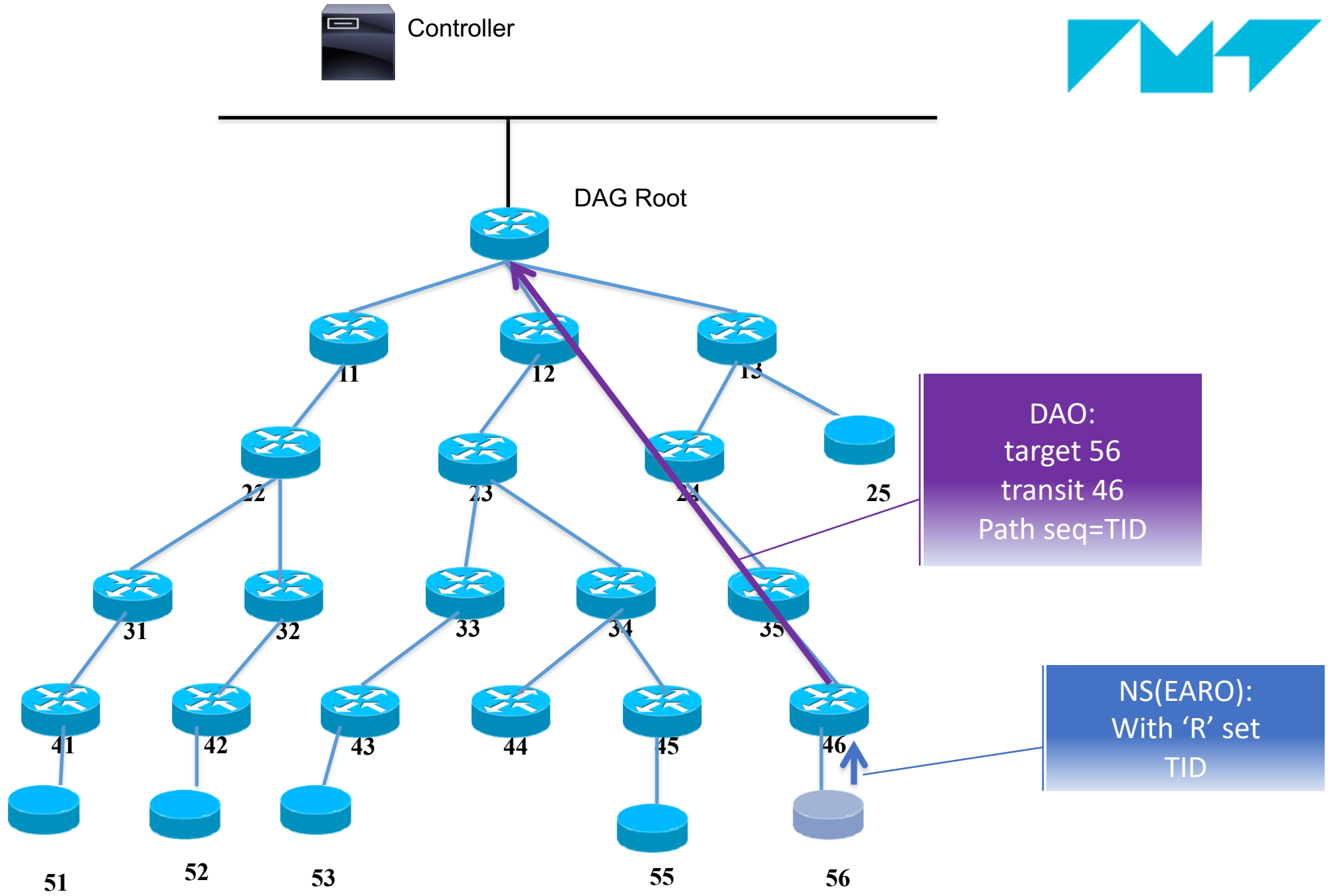
Discussion: What makes RPL green?

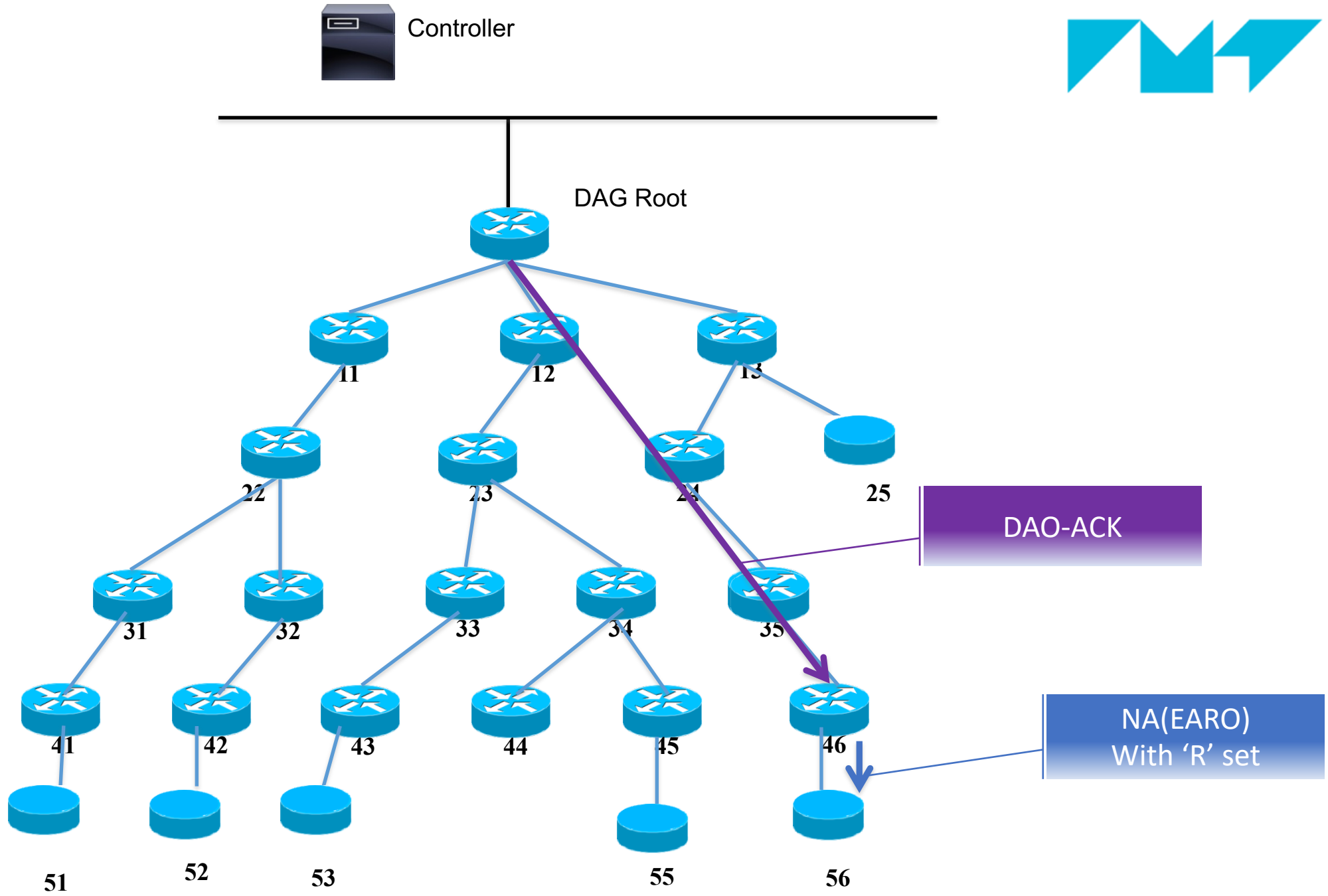
- Anisotropic (stretch)
- Neighbor selection (Trickle)
- DV (limited topological knowledge)
- Compatible with scheduled MAC (6TiSCH)
- Multipath (saves retries)

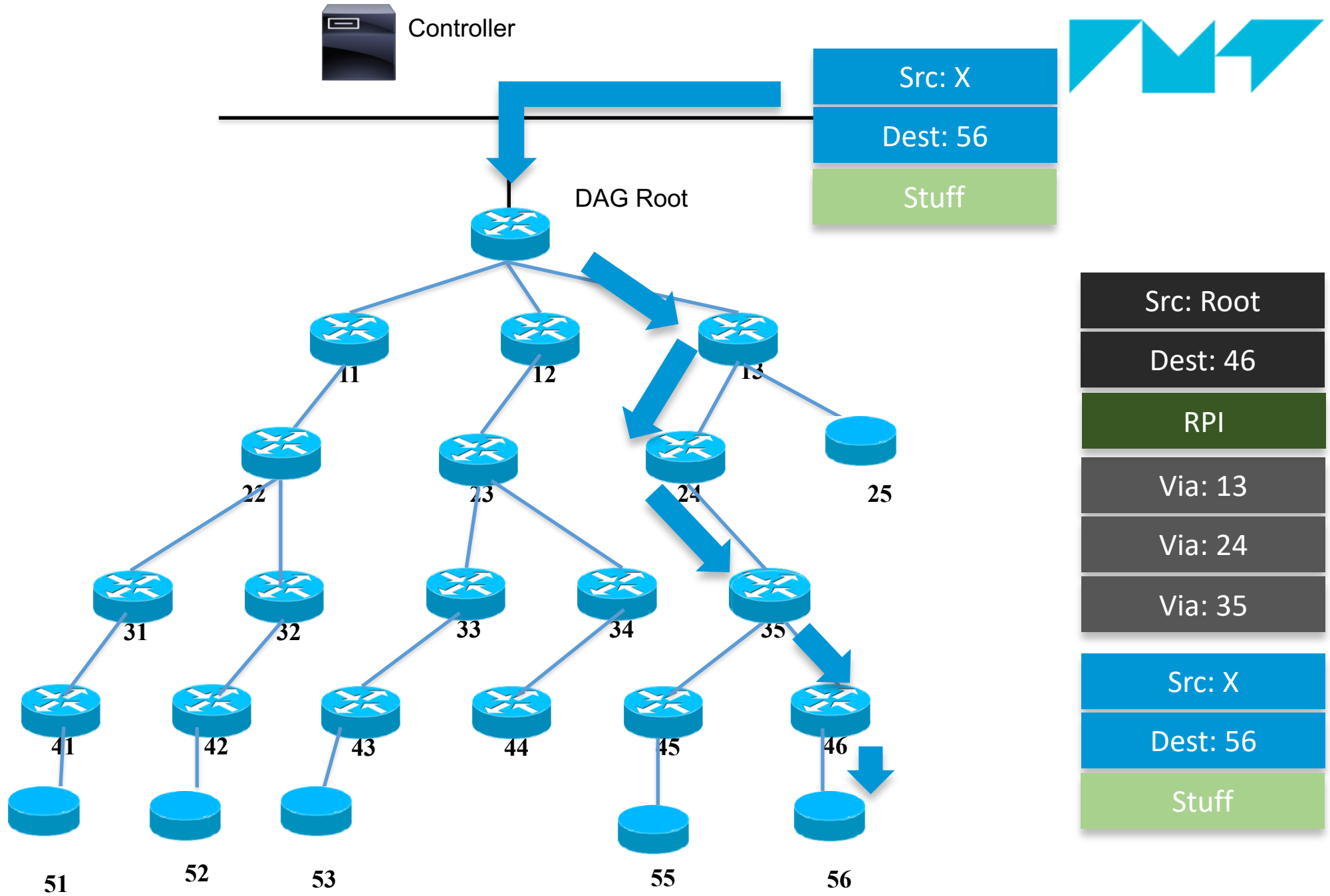
Applying SFAAC
to RPL

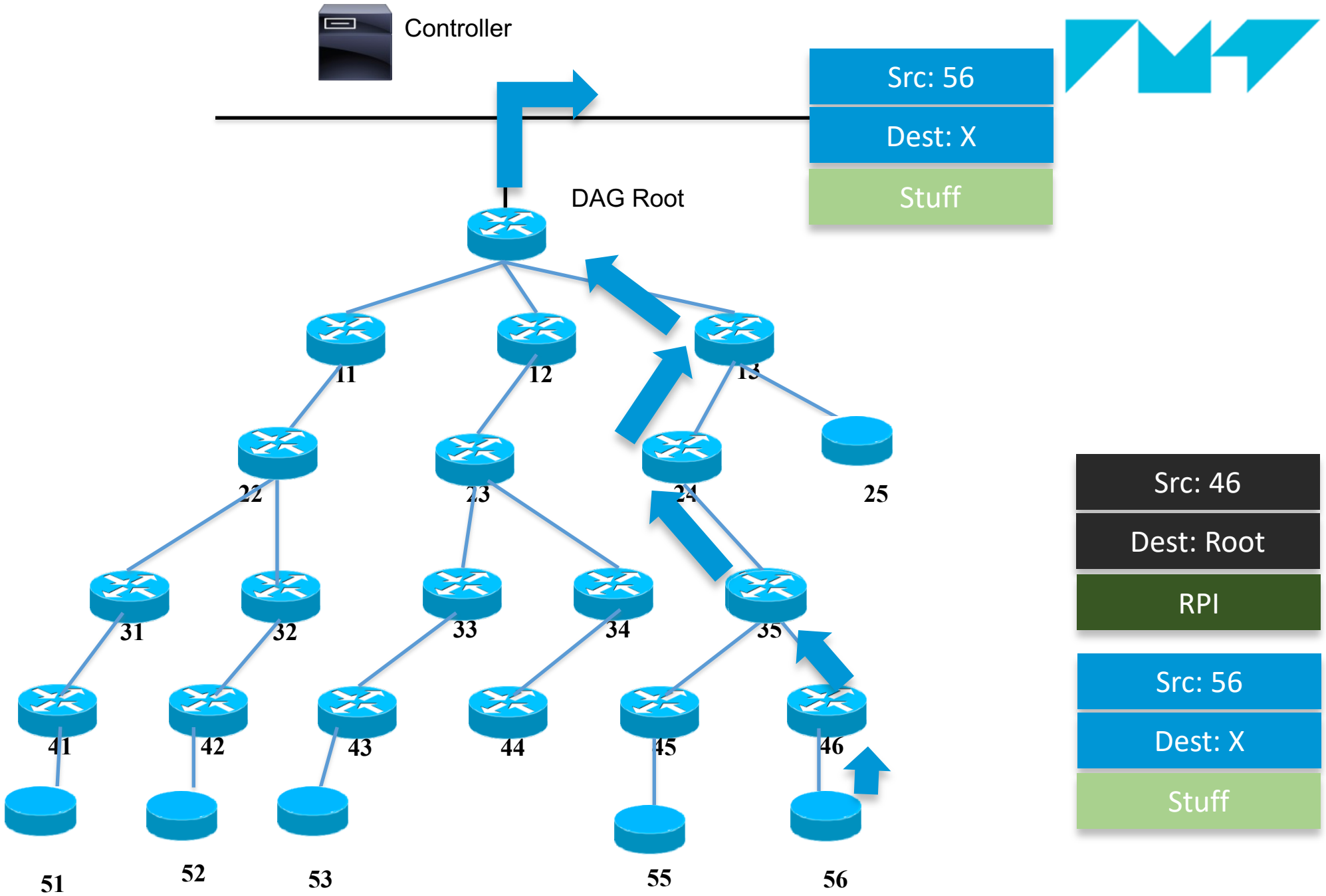
RPL Unaware Leaves (RFC 9010)

- Unaware Leaves Register their addresses
  - ‘R’ flag set: 6LR generates the DAO on behalf of RUL
  - DAO has ‘E’ external flag set to indicate external destination
- RFC 9008 provides the encapsulation rules
  - Describes external destination for which RPL is transit, e.g., RUL
  - RPL Non-Storing DAO signaling is used in all cases for RULs
  - Data Traffic tunneled between Root and 6LR, decapsulated and delivered
- draft-ietf-6lo-multicast-registration
  - Extends to multicast and anycast registrations
  - New RPL MOP for ingress replication in non storing mode


Controller
DAG Root
11
12
13
22
23
24
25
31
32
33
34
35
41
42
43
44
45
46
51
52
53
55
56
DAO:
target 56
transit 46
Path seq=TID
NS(EARO):
With 'R' set
TID


Controller
DAG Root
11
12
13
22
23
24
25
31
32
33
34
35
41
42
43
44
45
46
51
52
53
55
56
DAO-ACK
NA(EARO)
With 'R' set


Controller
Src: X
Dest: 56
Stuff
DAG Root
11
12
13
22
23
24
25
31
32
33
34
35
41
42
43
44
45
46
51
52
53
55
56
Src: Root
Dest: 46
RPI
Via: 13
Via: 24
Via: 35
Src: X
Dest: 56
Stuff

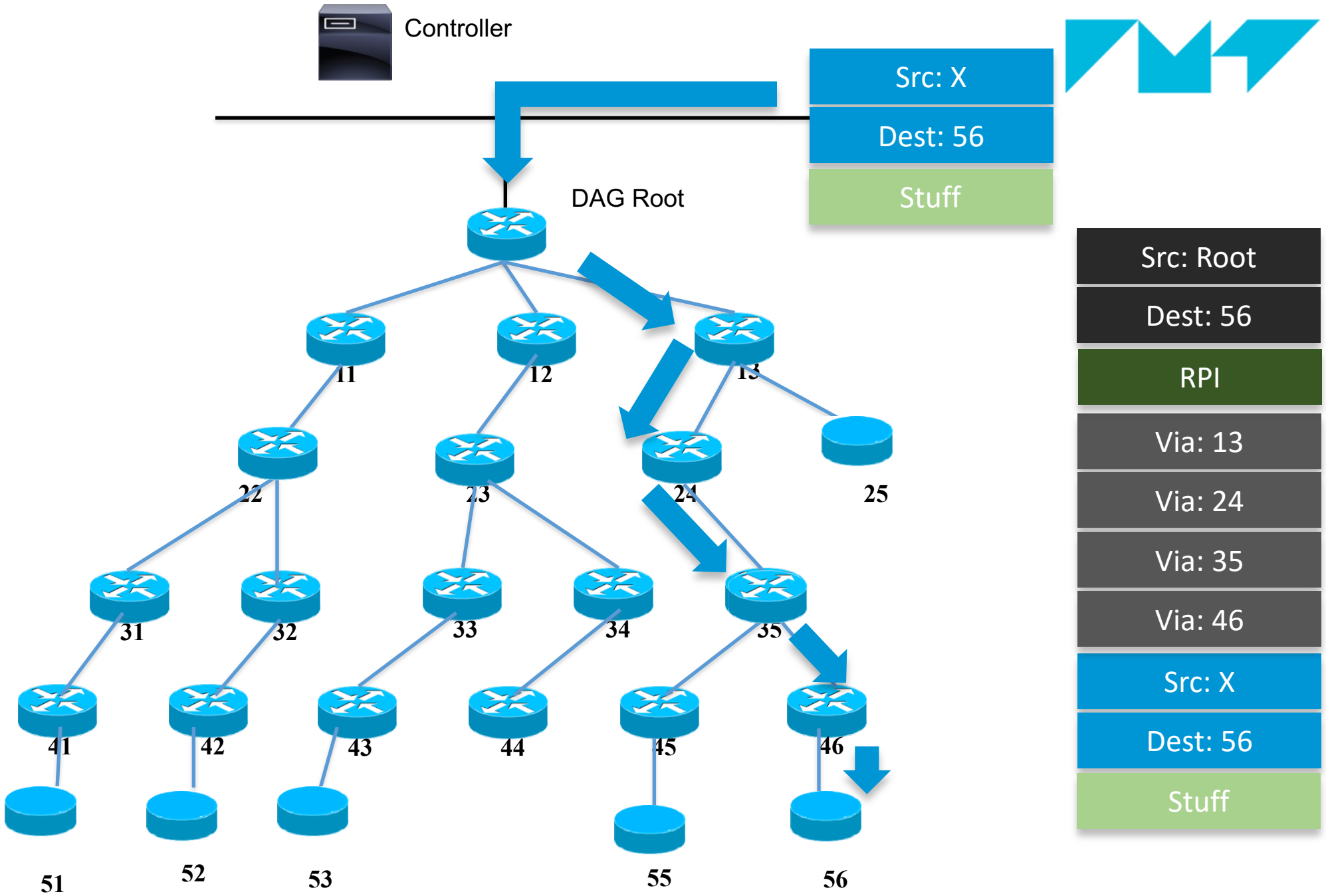

Controller
Src: 56
Dest: X
Stuff
DAG Root
11
12
13
22
23
24
25
31
32
33
34
35
41
42
43
44
45
46
51
52
53
55
56
Src: 46
Dest: Root
RPI
Src: 56
Dest: X
Stuff

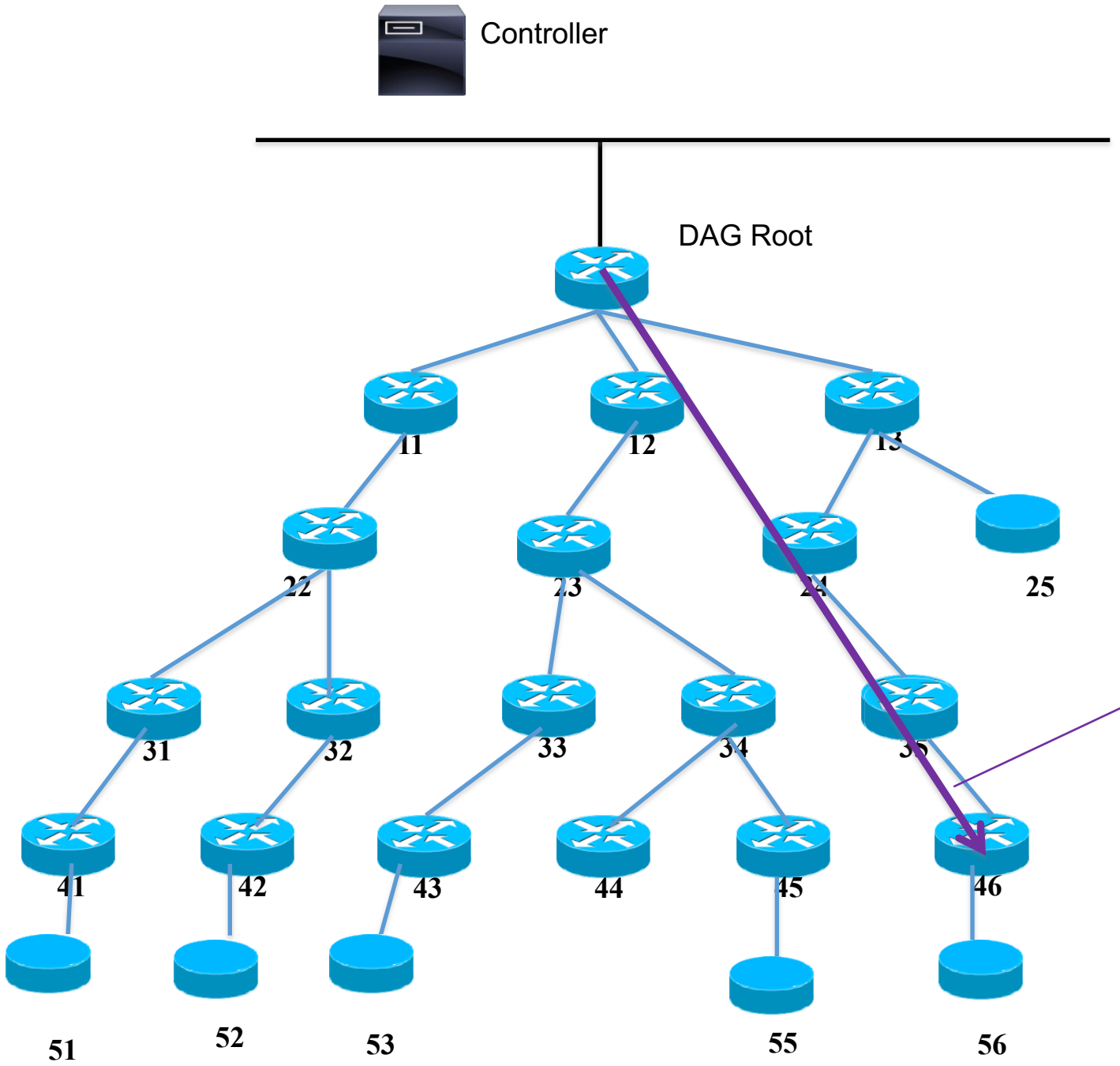
Part 2: RPL Next Steps

DAO Projection

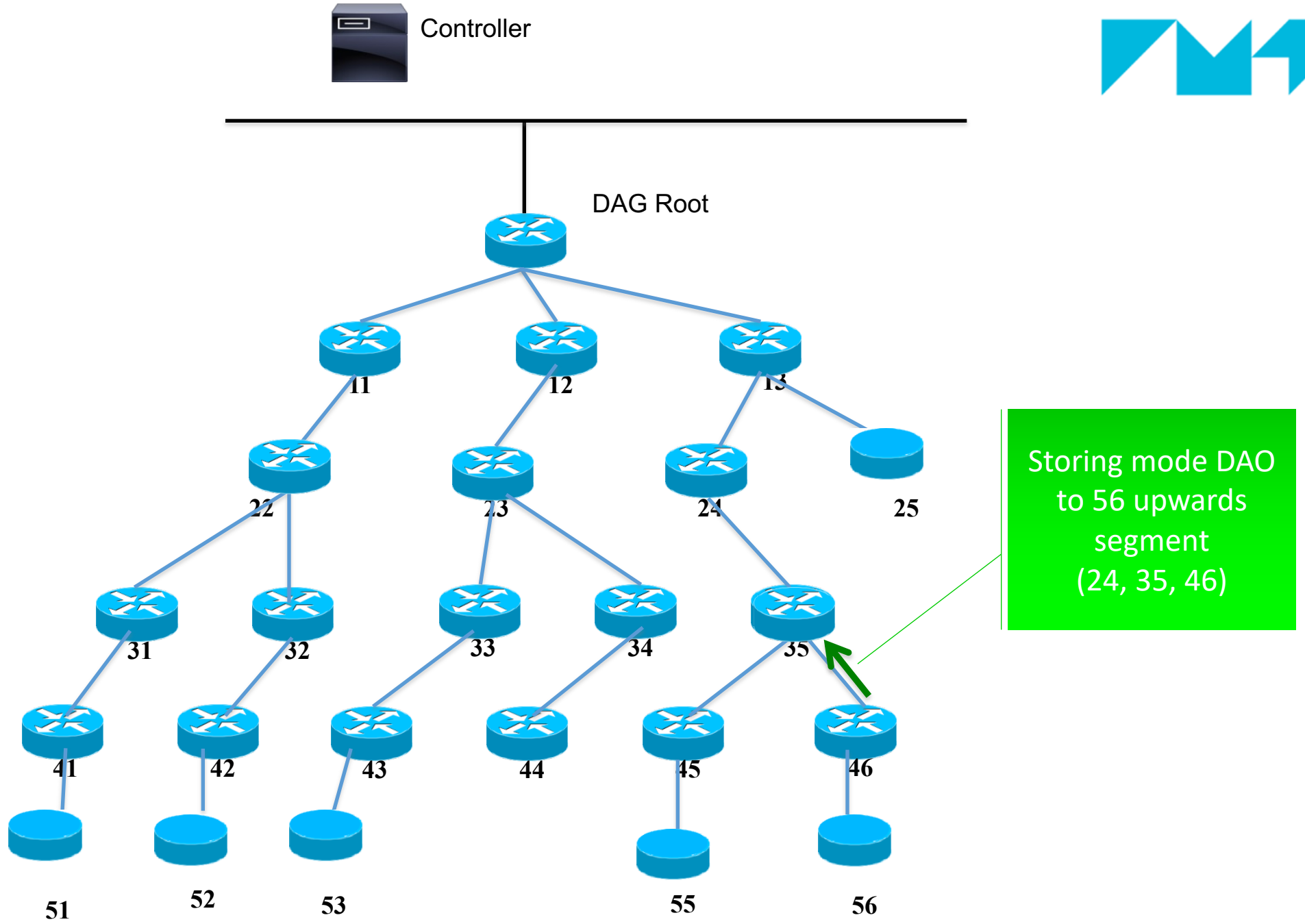
DAO Projection (Centralized RPL)

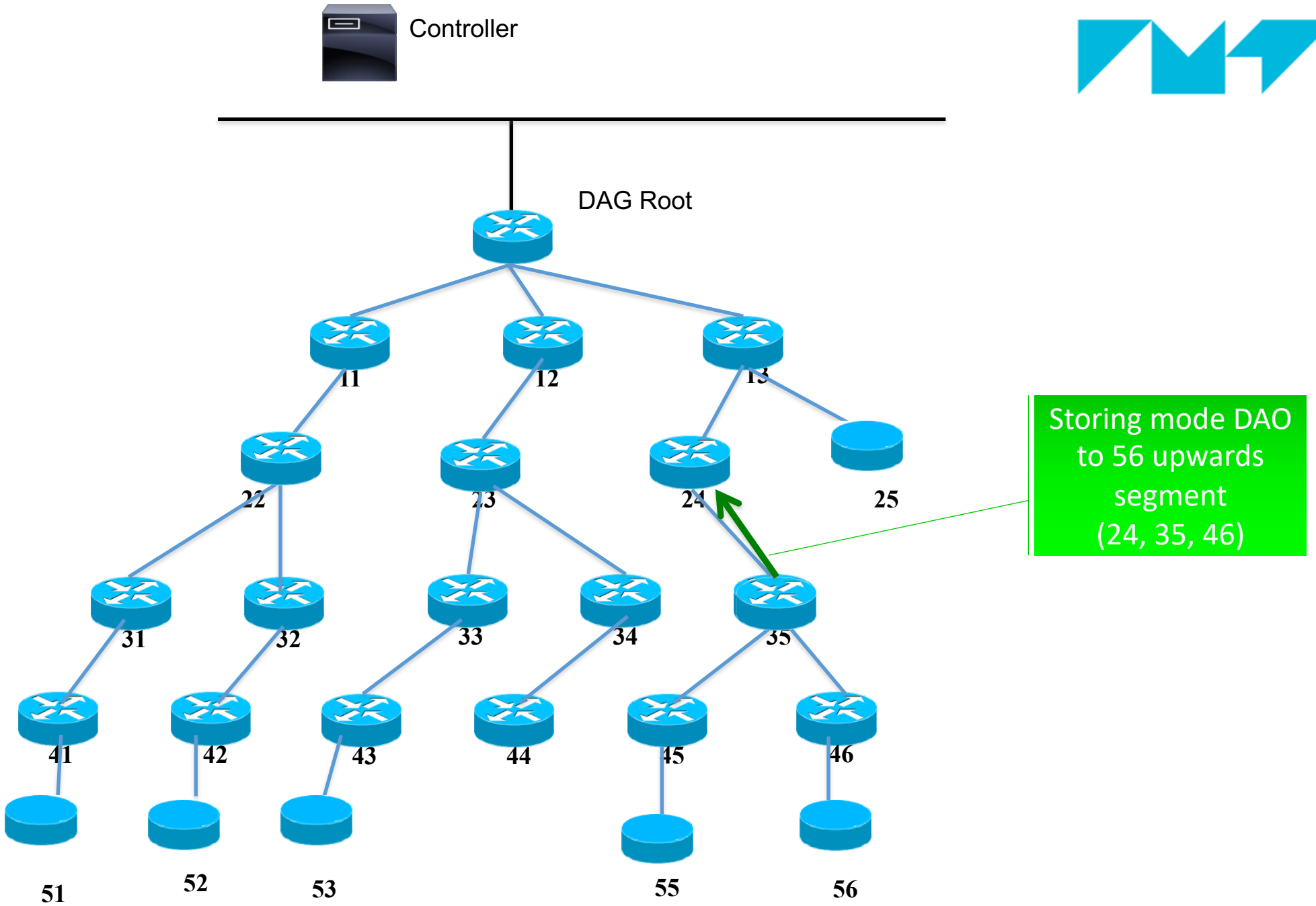
- Root connected-to or acting-as controller
  - Uses topological info from main DODAG
  - New Sibling Information Option (and P-DAO request)
  - Uses Projected DAO to install paths in the network
- Builds Segments to compress SHR
  - Compresses selected long paths in main DODAG
  - Uses Storing Mode Projected DAO to install strict (serial) paths
- Builds new DODAGs called Tracks
  - Enables optimized P2P (east – west) routing
  - Uses Non-Storing Mode Projected DAO to install loose (dotted-line) graphs
  - Leveraging Segments to complete the graph

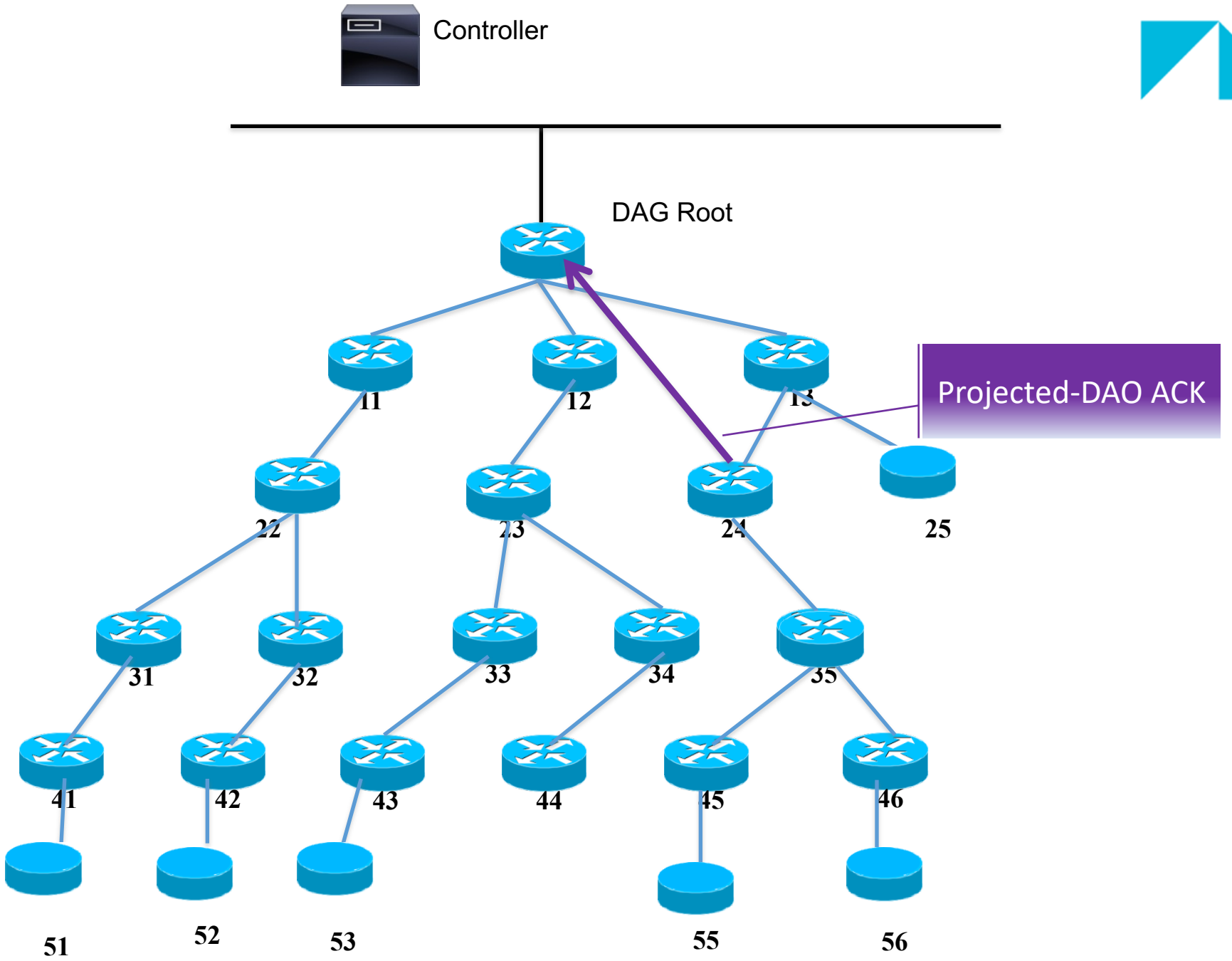

Controller
Src: X
Dest: 56
Stuff
DAG Root
11
12
13
22
23
24
25
31
32
33
34
35
41
42
43
44
45
46
51
52
53
55
56
Src: Root
Dest: 56
RPI
Via: 13
Via: 24
Via: 35
Via: 46
Src: X
Dest: 56
Stuff

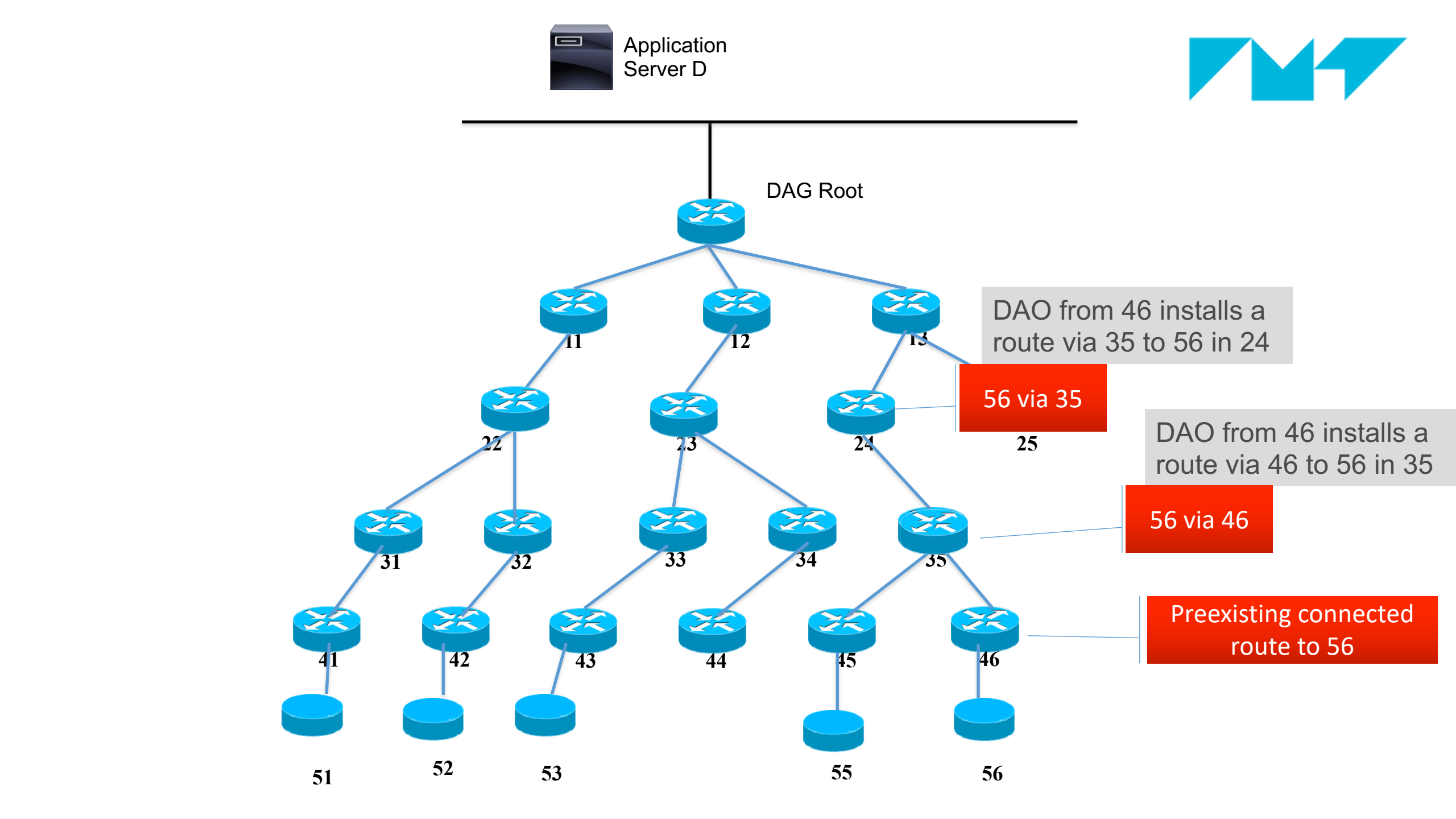


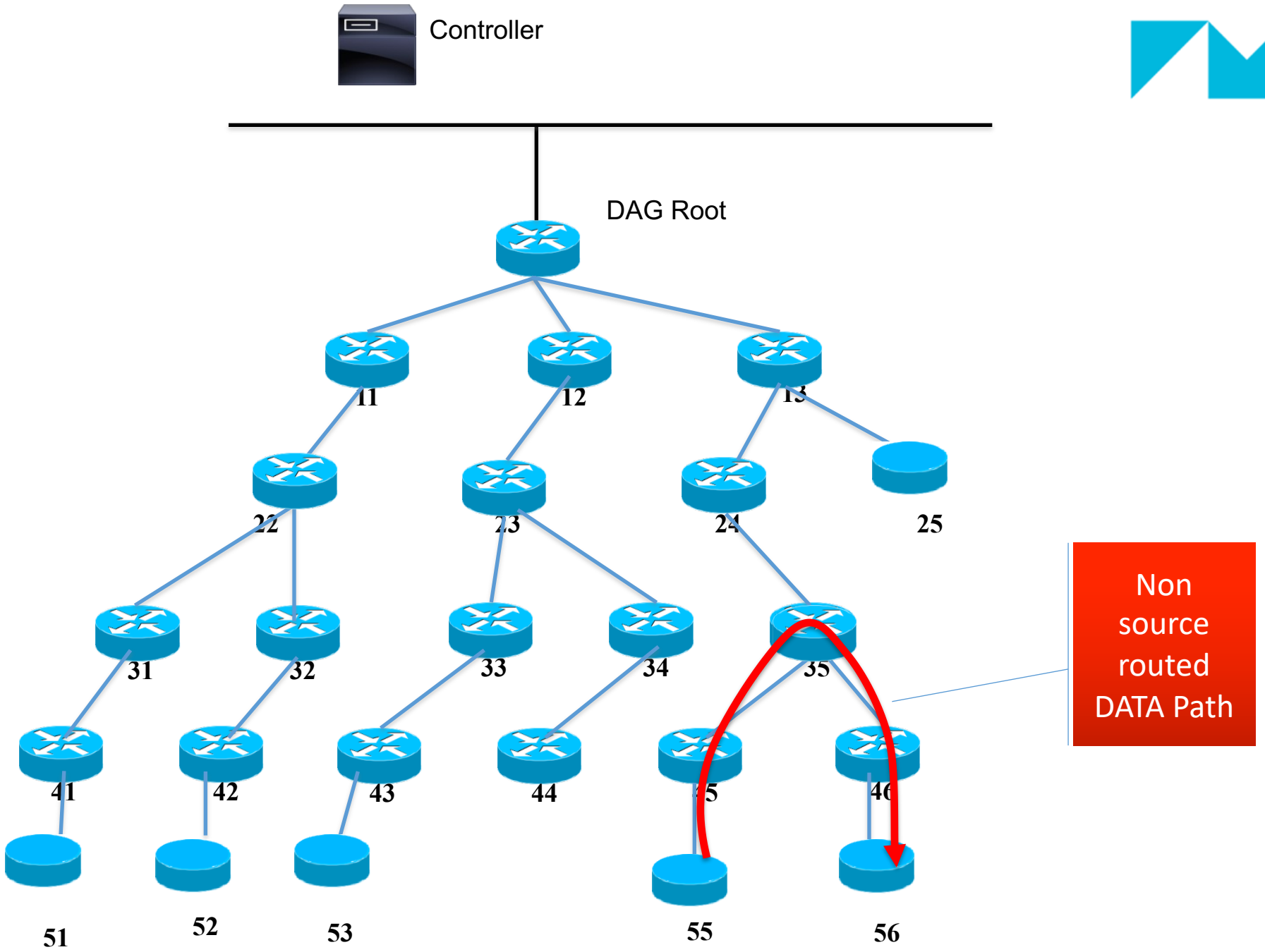
Projected-DAO to target 56 with path segment via 24 (ingress), 35, and then 46 (egress)

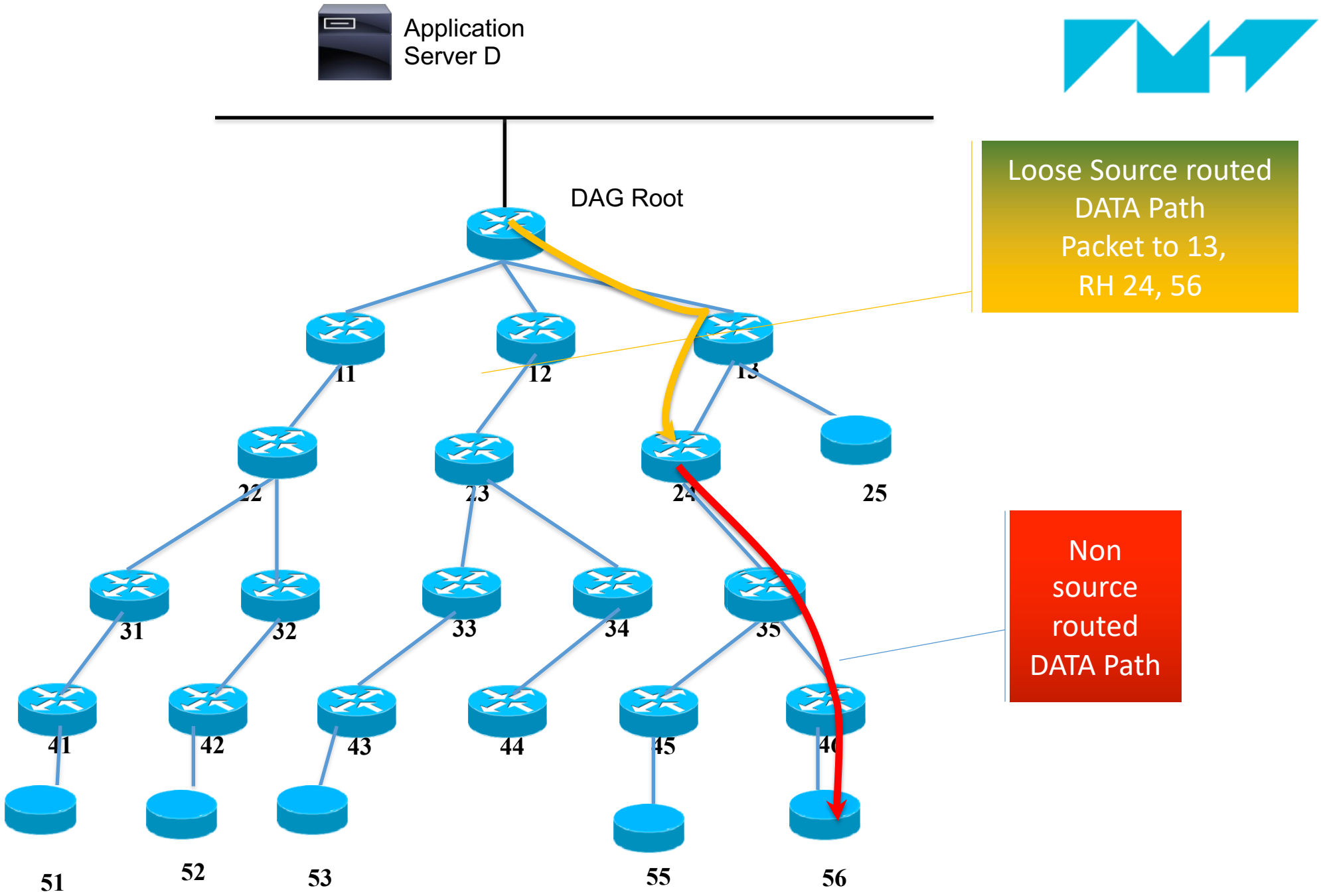

Controller
DAG Root
11
12
13
22
23
24
25
31
32
33
34
35
41
42
43
44
45
46
51
52
53
55
56
Storing mode DAO to 56 upwards segment (24, 35, 46)

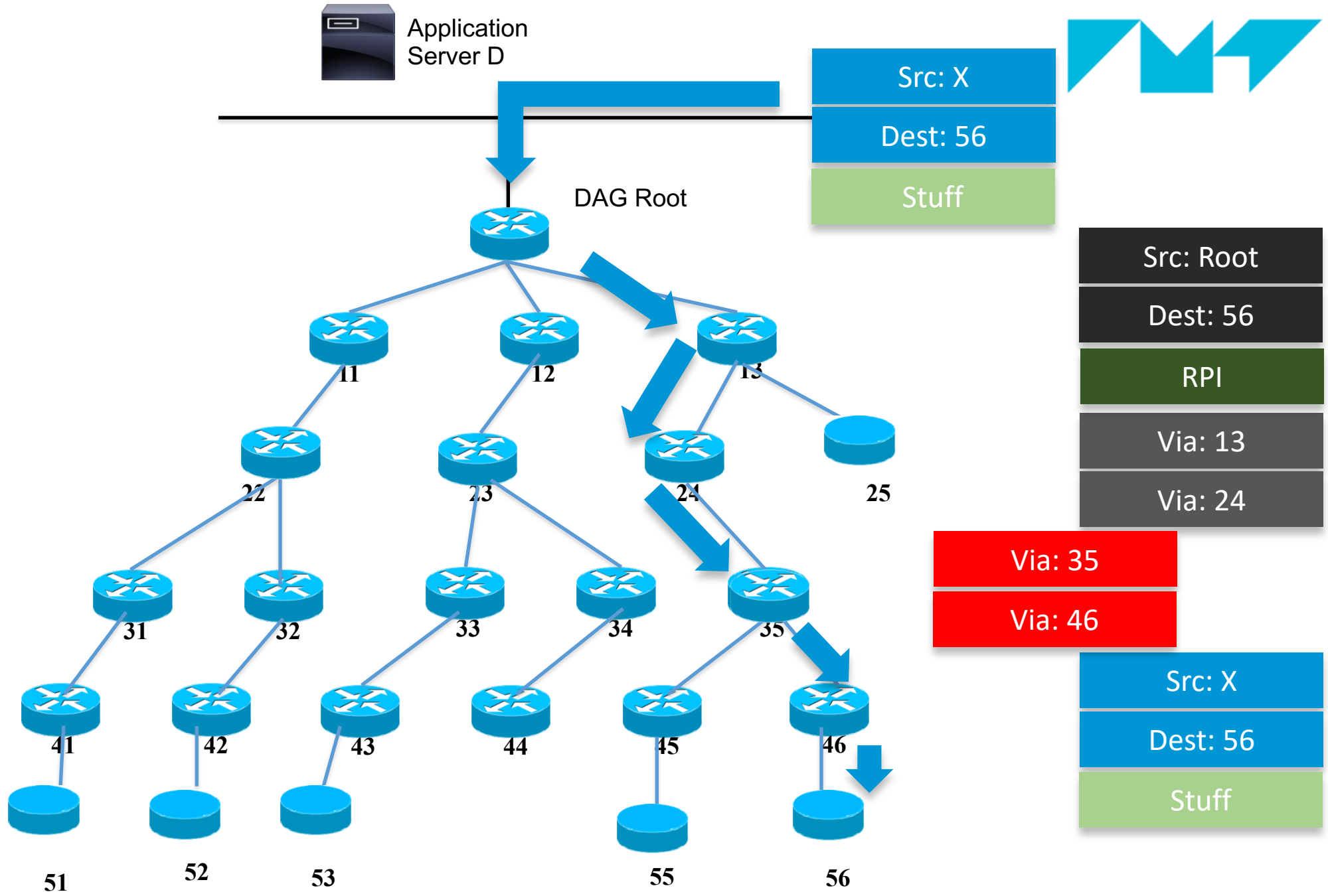

Controller
DAG Root
11
12
13
22
23
24
25
31
32
33
34
35
41
42
43
44
45
46
51
52
53
55
56
Storing mode DAO to 56 upwards segment (24, 35, 46)


Controller
DAG Root
Projected-DAO ACK
11
12
13
22
23
24
25
31
32
33
34
35
41
42
43
44
45
46
51
52
53
55
56


Application
Server D
DAG Root
11
12
13
22
23
24
25
31
32
33
34
35
41
42
43
44
45
46
51
52
53
55
56
DAO from 46 installs a route via 35 to 56 in 24
56 via 35
DAO from 46 installs a route via 46 to 56 in 35
56 via 46
Preexisting connected route to 56


Controller
DAG Root
11
12
13
22
23
24
25
31
32
33
34
35
41
42
43
44
46
51
52
53
55
56
Non source routed DATA Path

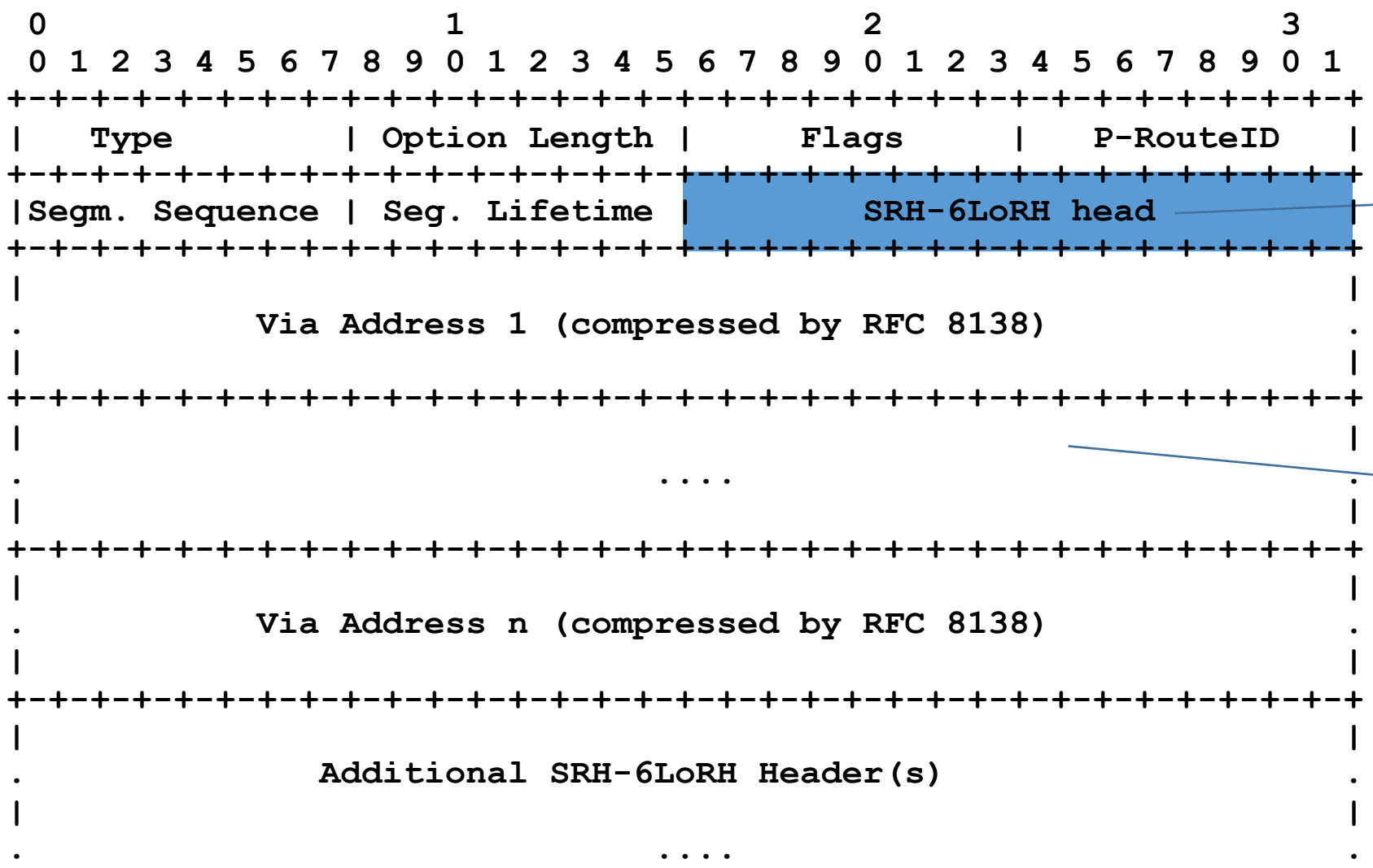

Application
Server D
DAG Root
11
12
13
22
23
24
25
31
32
33
34
35
41
42
43
44
45
46
51
52
53
55
56
Loose Source routed
DATA Path
Packet to 13,
RH 24, 56
Non
source
routed
DATA Path


Application
Server D
Src: X
Dest: 56
Stuff
DAG Root
Src: Root
Dest: 56
RPI
Via: 13
Via: 24
Via: 35
Via: 46
Src: X
Dest: 56
Stuff
11
12
13
22
23
24
25
31
32
33
34
35
41
42
43
44
45
46
51
52
53
55
56

P-DAO construction

- RPL Target Options can be factorized
- But there is one and only one VIO (SF-VIO or SR-VIO)
- So the Ack management is easier
- VIO sent to egress; SR-VIO sent to ingress
- Track ID is a RPL local instance ID
- Taken from the Track Egress Name Space

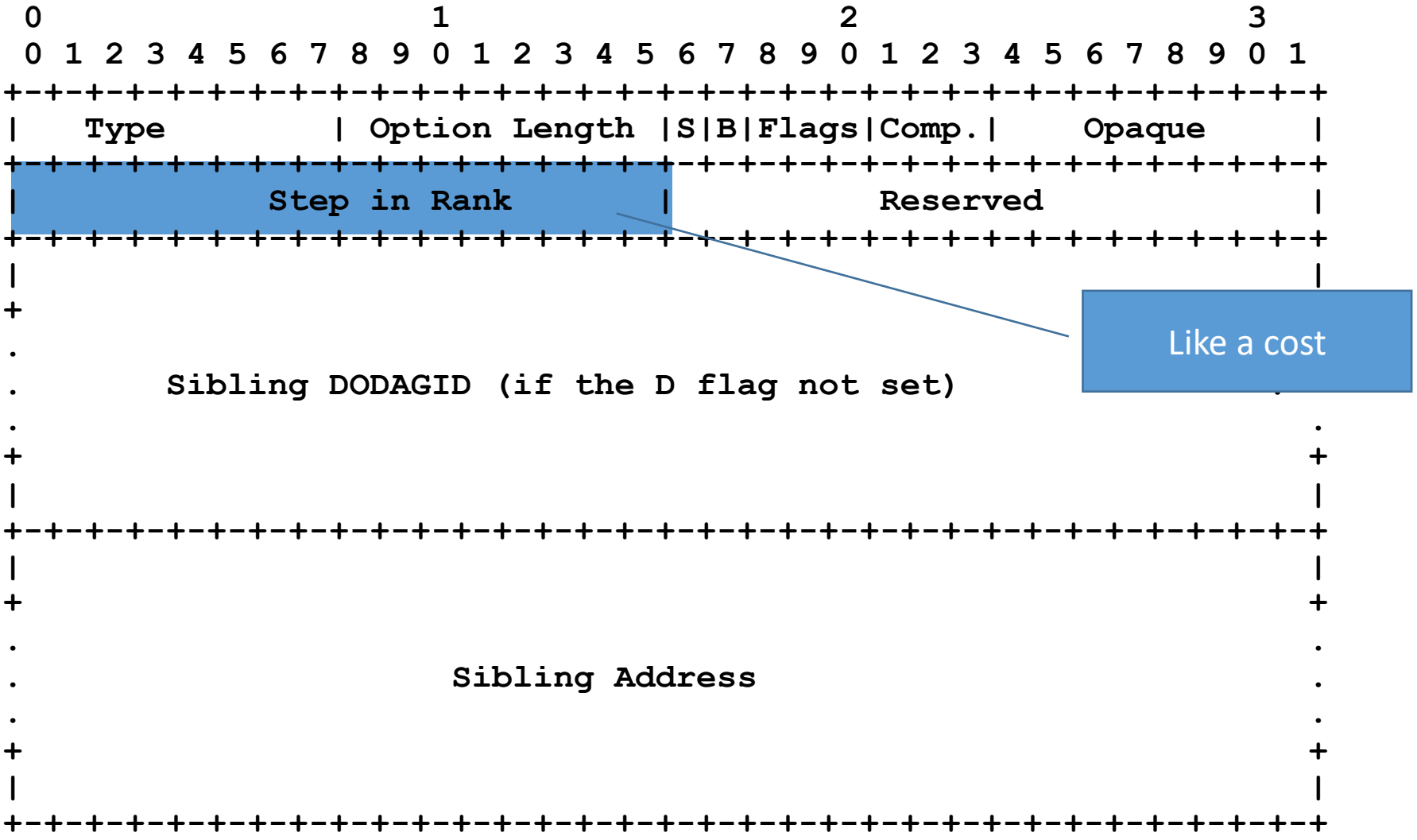
New Via Information Option Format



May be more than one in Non-storing Mode

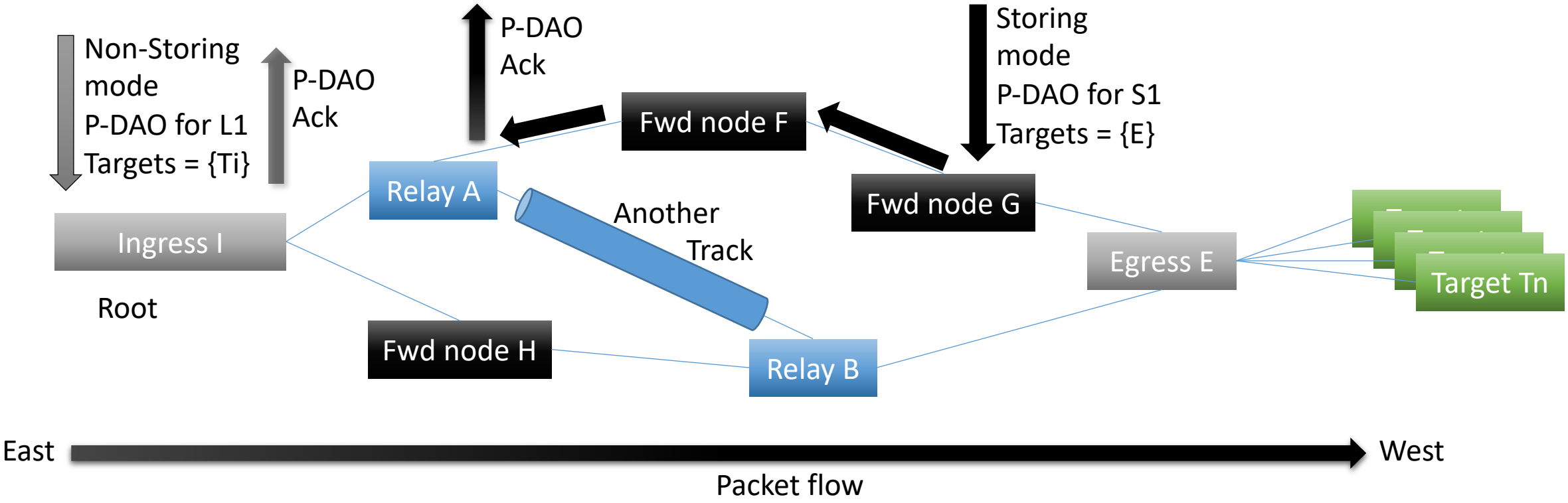
Must be optimized in Non-storing Mode, to be used as is in packets

New Sibling Information Option Format



In DAO and mcast DAO; mcast DAO allows indirect forwarding

The RPL Track: A local DODAG rooted at Ingress



Targets {Tx }

Legs L1 = I->A->E to {Ti} , L2 = I->B->E to {Ti} , L3 = I->A->B->E to {Ti}

Segments S1 = A=>F=>G to E , S2 = I=>H to B

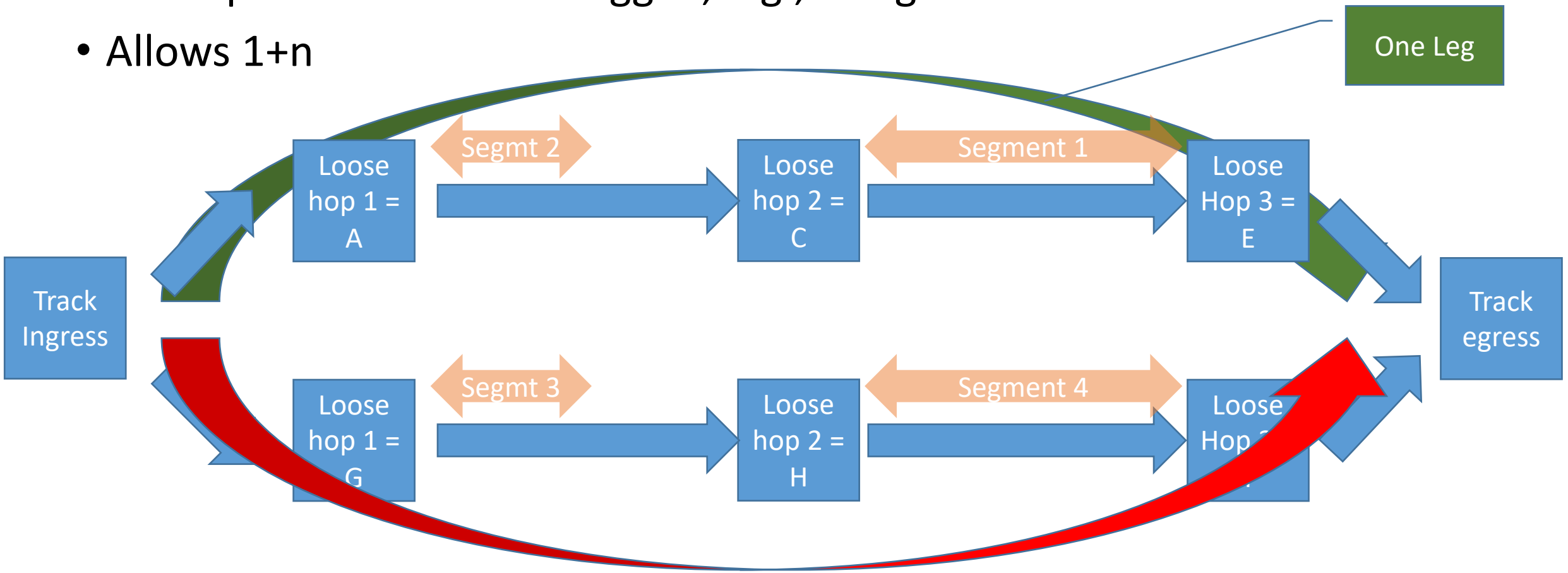
SubTracks Any Set ⊂ {L1, L2, L3} but { }

Some rules

- Track is set up by installing Legs and Segment
  - with the same Track ID
- Non-Storing Mode P-DAO signals a Leg
- Storing Mode P-DAO signals a Segment
- Storing Mode P-DAO enables loose hops
  - in Non-Storing main DODAG (typically TrackId is Global instance ID)
  - in Tracks (typically TrackId is Local instance ID to track Ingress)
- Track Egress is implicit Target in Non-Storing Mode
- Leg hop is either a Segment of this Track or another Track

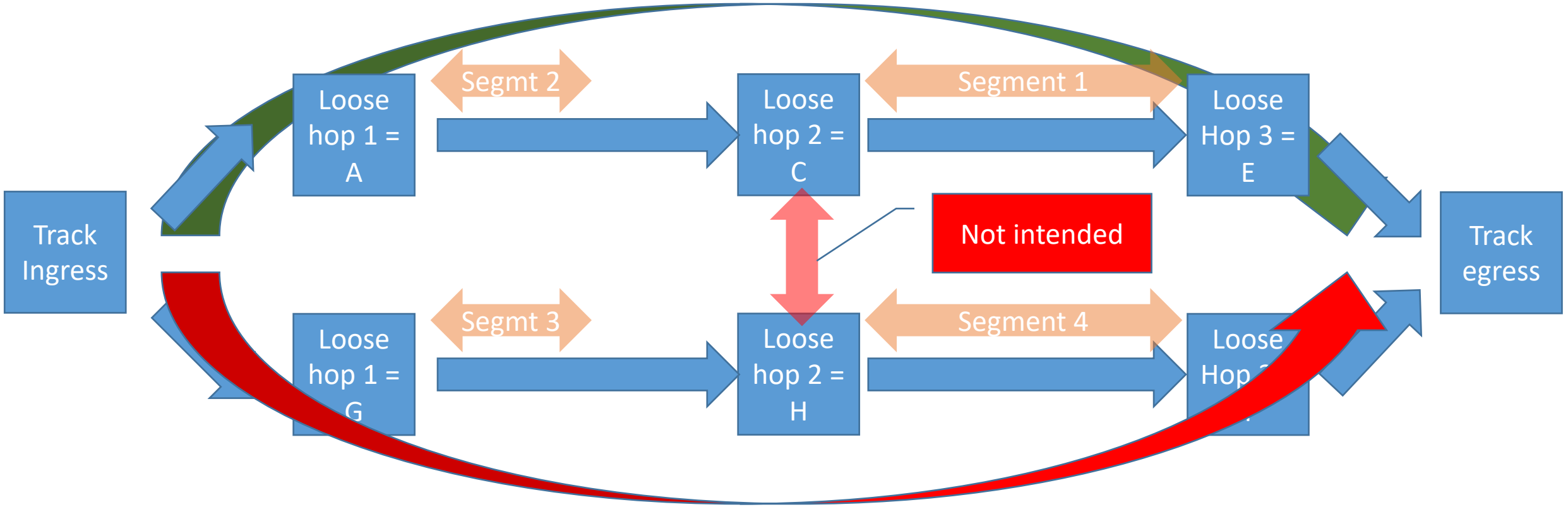
Complex track

- A complex track is multi-legged, e.g., 2 Legs below
- Allows 1+n



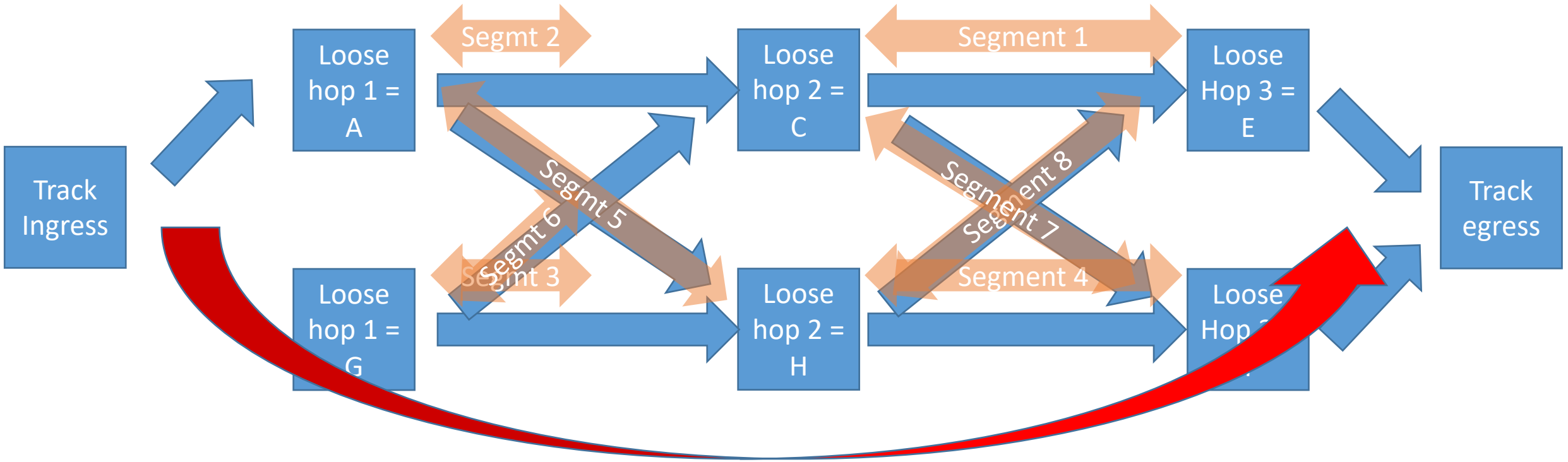
RPL vs RAW

- RPL has no North-South Segment



Inter Leg

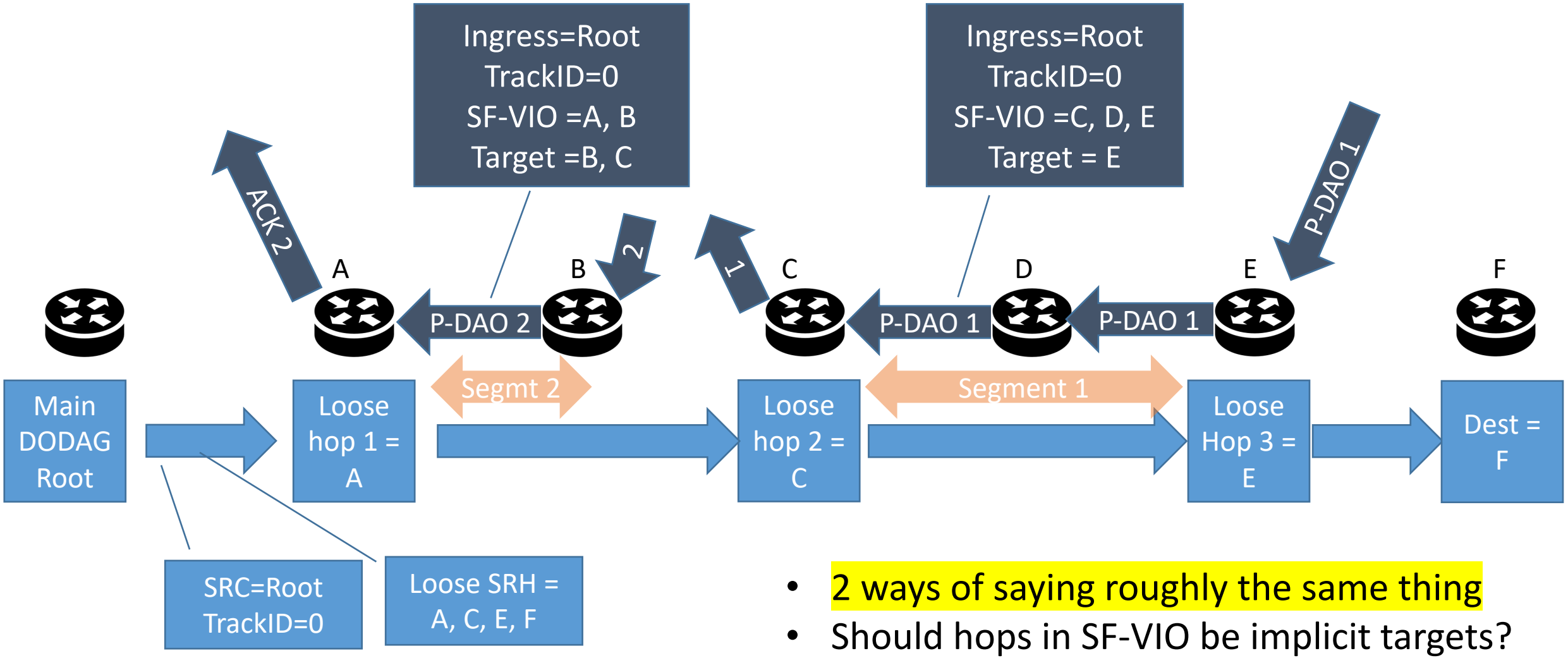
- RFC 6550 non-storing Target and Transit to indicate loose parent child relationship, many of them in one P-DAO



Encapsulation Details

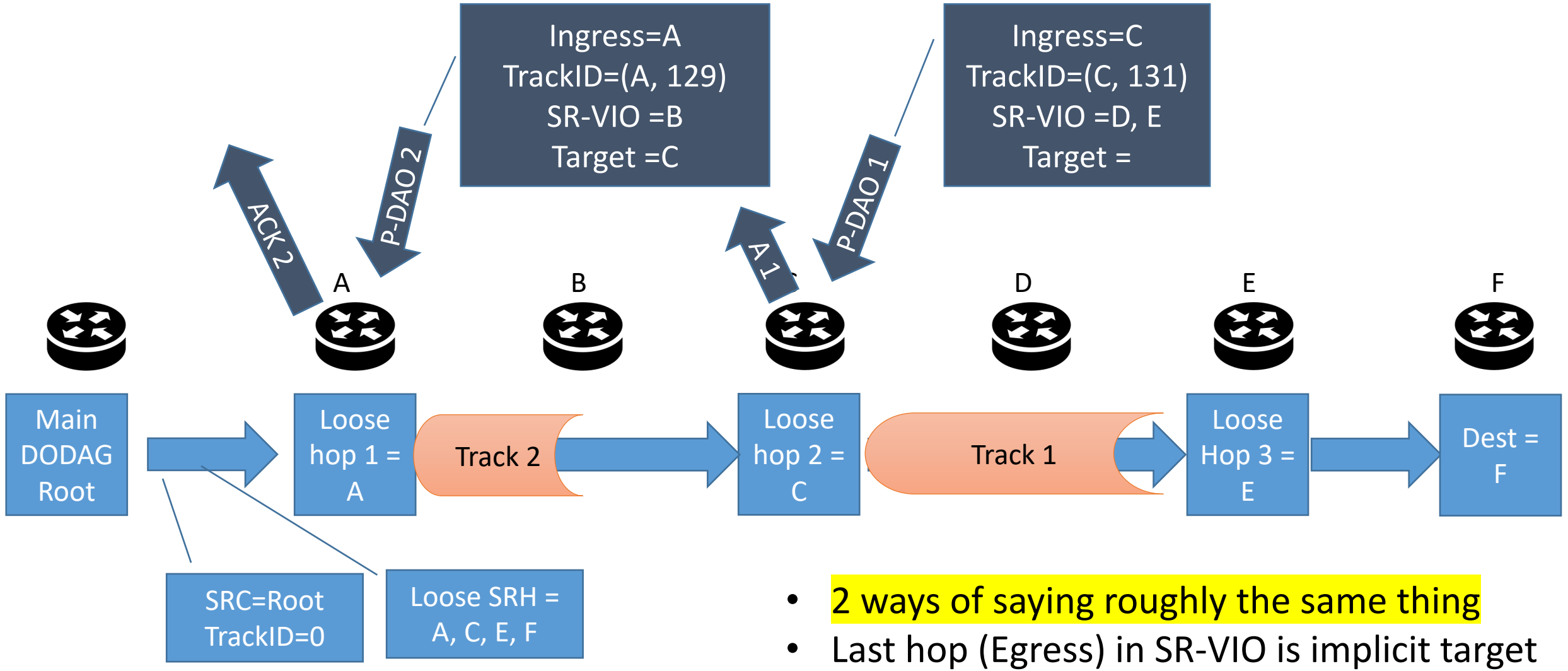
- Source of outer header MUST be Track Ingress- think DODAG Root
- RPL Instance ID in RPI MUST indicate TrackID (if not main DODAG)
- SR-VIO: Loose from Track Ingress, excluded, to Egress, included
  - Copied Verbatim in inserted SRH-6LoRH,
  - Requires encapsulation (can be recursive)
- SF-VIO: Strict from Segment Ingress to Egress, both included
  - No Encapsulation if Source and RPI both match Segment definition
  - A Segment is an Implicit Track if P-DAO Ingress == 1st SF-VIO entry
- TBD: matching rules, Flow Info option, when to tunnel?

Profile 1:
Compress SRH in main DODAG with strict SM Segments



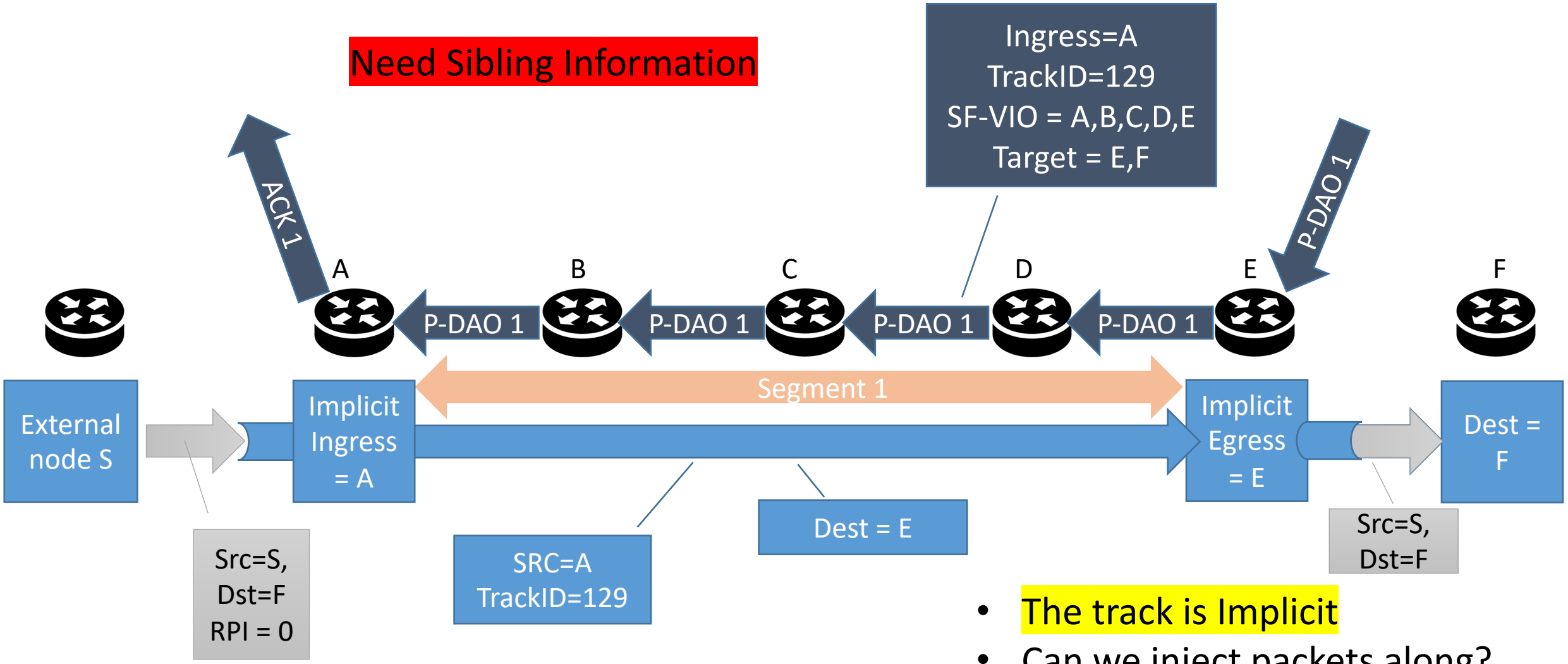
- 2 ways of saying roughly the same thing
- Should hops in SF-VIO be implicit targets?

Profile 2:
Compress SRH in main DODAG with Strict NSM Tracks



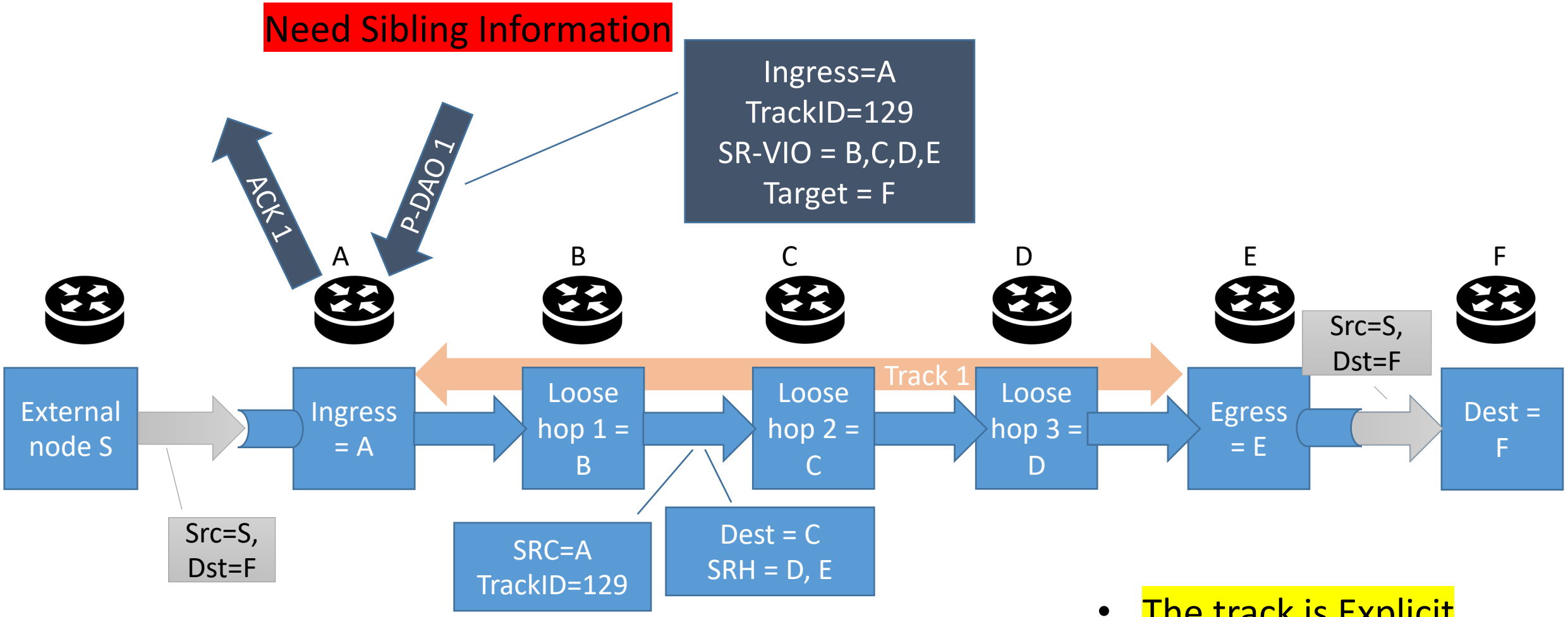
- 2 ways of saying roughly the same thing
- Last hop (Egress) in SR-VIO is implicit target

Profile 3: Implicit Track with Strict SM Segments,



- The track is Implicit
- Can we inject packets along?

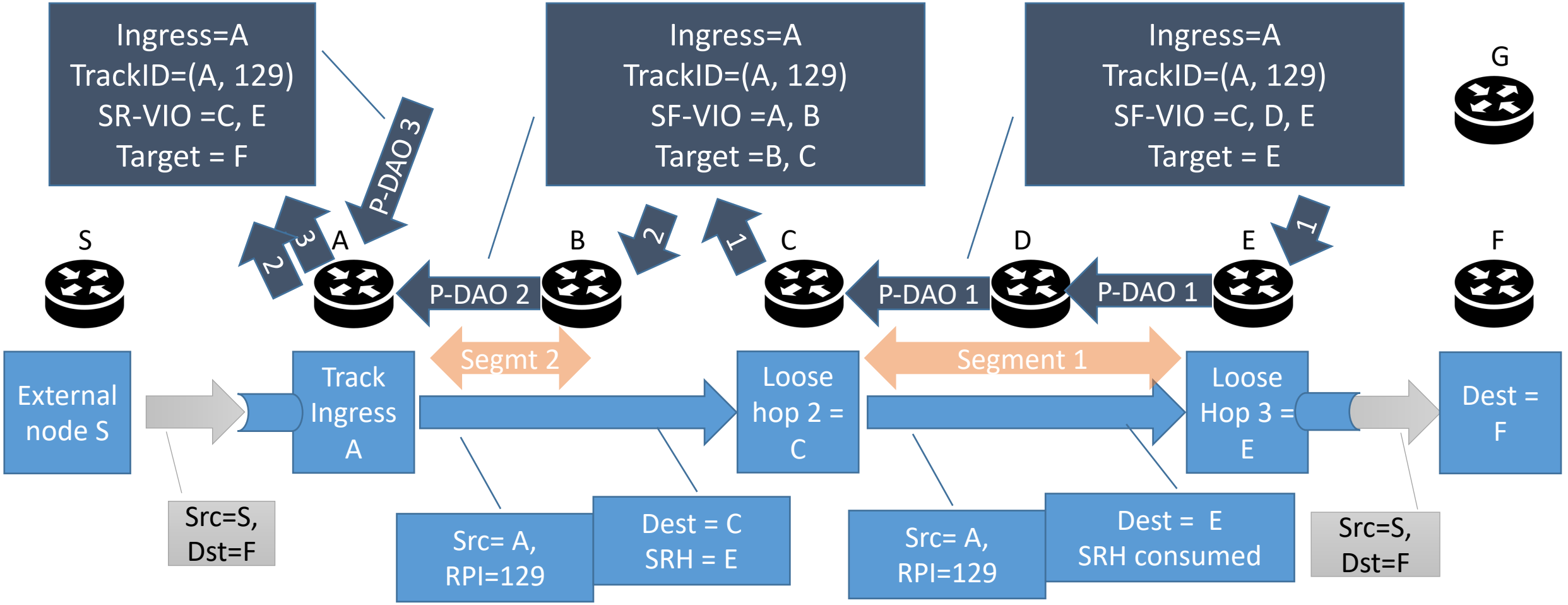
Profile 4: Strict NSM Explicit Track



- The track is Explicit
- Same encap as profile 2

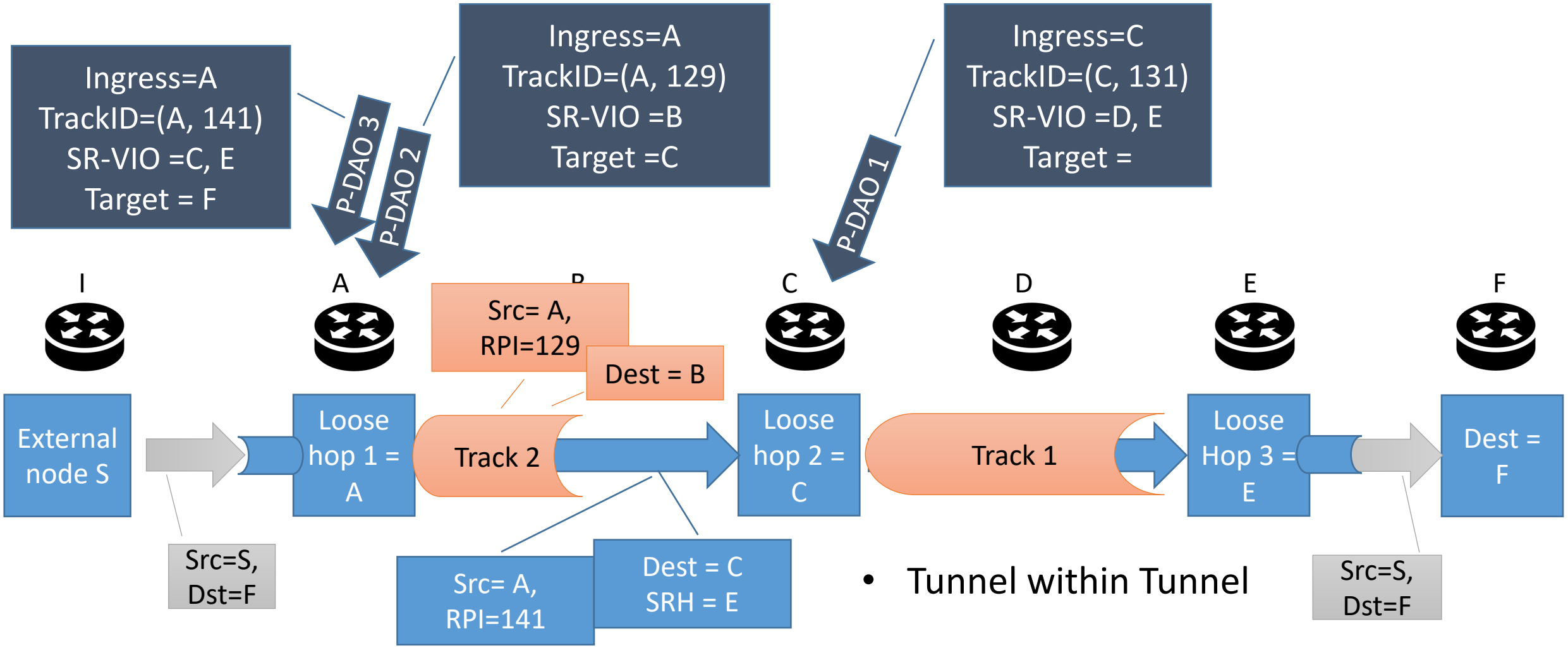
Profile 5:
Compress SRH in Track with Strict SM Segments

Need Sibling Information



- Same as Profile 1, but for Track

Profile 6:
Compress SRH in Track with NSM Tracks (Recursive?)



- Tunnel within Tunnel

RPL key concepts

RPL is an adaptable anisotropic IPv6 DV protocol
- Proactive routing for MP2P, P2MP, and stretched P2P
- P2P (reactive) and SDN (centralized) variations

RPL specifically designed for LLNs
- Agnostic to underlying link layer technologies (802.15.4, PLC, Low Power WiFi)
- Adapted to use case(s) with Objective Functions

Objective Functions => instantiation per constraints/metrics

Anisotropic Routing => minimum topological awareness

Data Path validation => lazy / reactive maintenance

Trickle Algorithm => Optimized Diffusion over NBMA

Non-Equal Cost Multipath Forwarding

Multi-topology proactive routing

Autonomic (carries its own configuration)


CISCO

Questions ?

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

CISCO

Controlling the control … by design

Distance Vector as opposed to Link State

- Knowledge of SubDAG addresses and children links
- Lesser topology awareness => lesser sensitivity to change
- No database Synchronization => Adapted to movement

Optimized for Edge operation

- Optimized for P2MP / MP2P, stretch for arbitrary P2P
- Anisitropic Least Overhead Routing Approach via common ancestor

Proactive route installation, reactive failure discovery

- Actually both with AODV (P2P) draft

Datapath validation

- Loop Avoidance and inconsistency detection