P802.1Qca D0.6 Tutorial

Explicit Path Control

János Farkas
janos.farkas@ericsson.com

March18, 2014

Outline

Presentation Objectives

- Explore the operation of explicit tree establishment as described in P802.1Qca D0.6 through examples
- Focus on the Explicit ECT Algorithms
- Explore the features provided
  - Simplifications are possible

- Note that this presentation and http://www.ieee802.org/1/files/public/docs2013/ca-farkas-d0-4-operation-v01.pdf essentially say the same just from a little bit different angle

Disclaimer

- The operation presented here is not the final standard!
- There are open items and items under debate

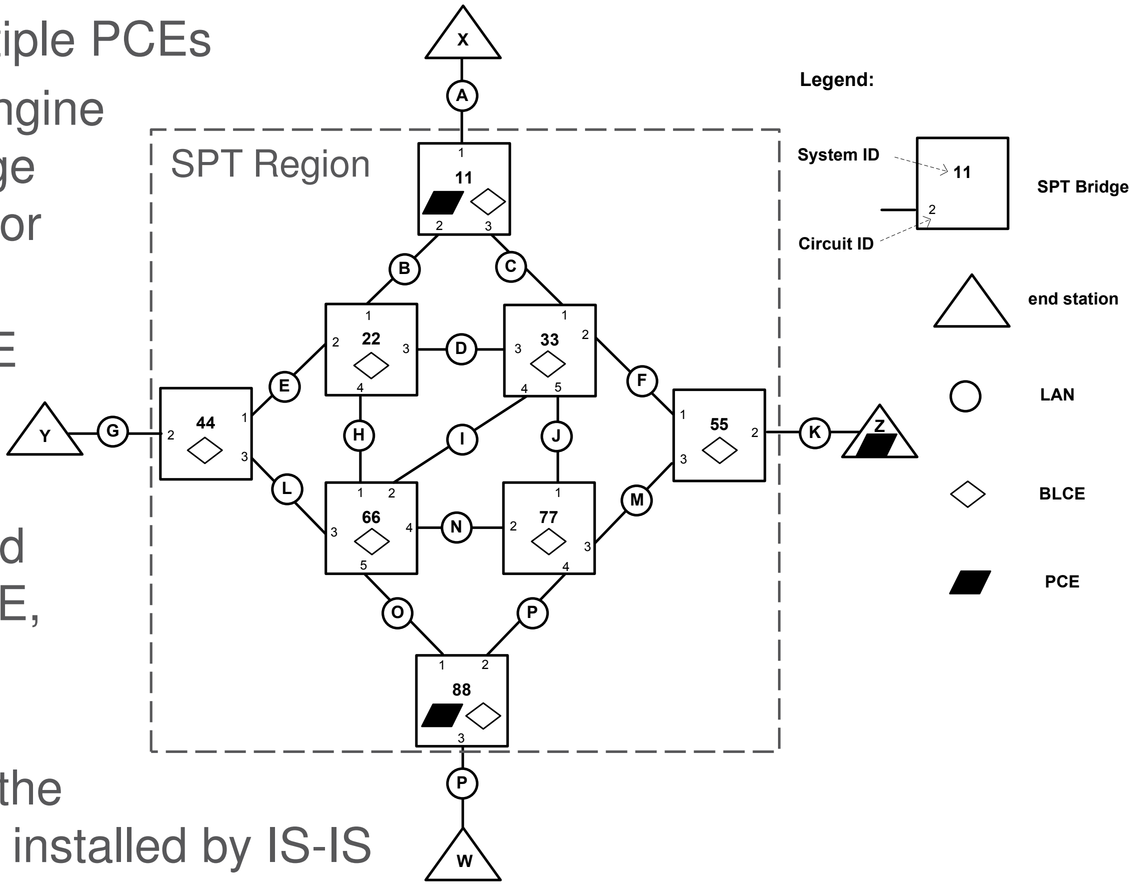
Highlights

- 802.1Qca is an extension to IS-IS
- It is control plane
- Main goal: establishment of explicit trees
  - 802.1Qca D0.6 is suitable for more generic explicit graphs
- An explicit tree is an undirected loop free graph
- Explicit trees do not require hardware changes!
- Forwarding is made directed (unidirectional) by MAC
- Forwarding can be made directed (unidirectional) by VID

- The algorithm the PCE uses for path computation is not specified by 802.1Qca

Explicit Trees

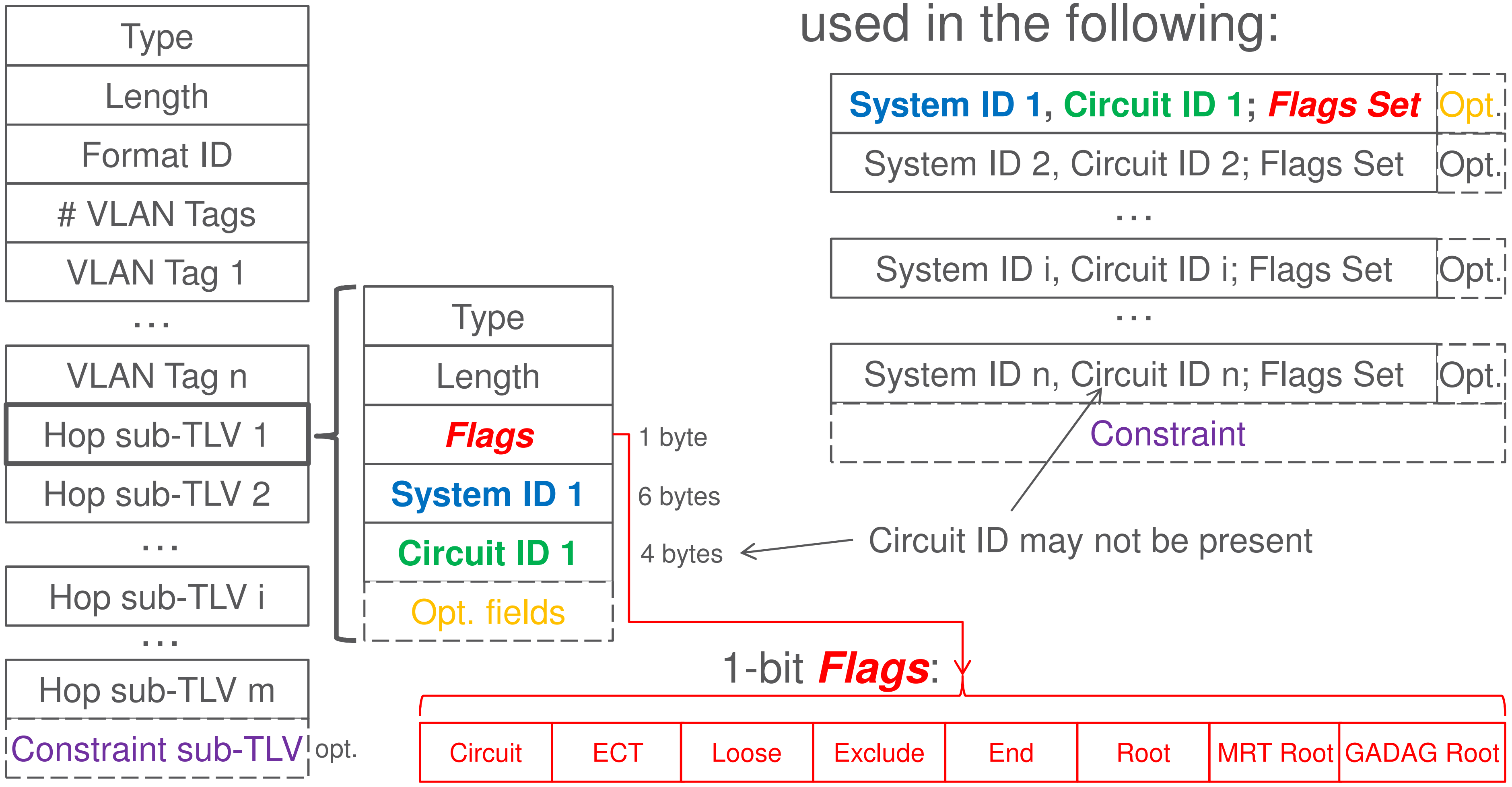
- An Explicit Tree (ET) is controlled by a Path Computation Element (PCE) via IS-IS
- A PCE is a higher layer entity in a bridge or an end station
- An SPT Region may have multiple PCEs
- A Bridge Local Computation Engine (BLCE) is hosted by each bridge for (constrained) shortest path or MRT computation
- An ET is controlled by one PCE
- An ET is either fully specified or completely loose
- A fully specified ET is computed and described by its owner PCE, and then installed by IS-IS
- A completely loose ET only comprises the End Points and the ET is computed by the BLCEs, installed by IS-IS
- Loose and strict hops can be only mixed in a p2p path (as per D0.6)



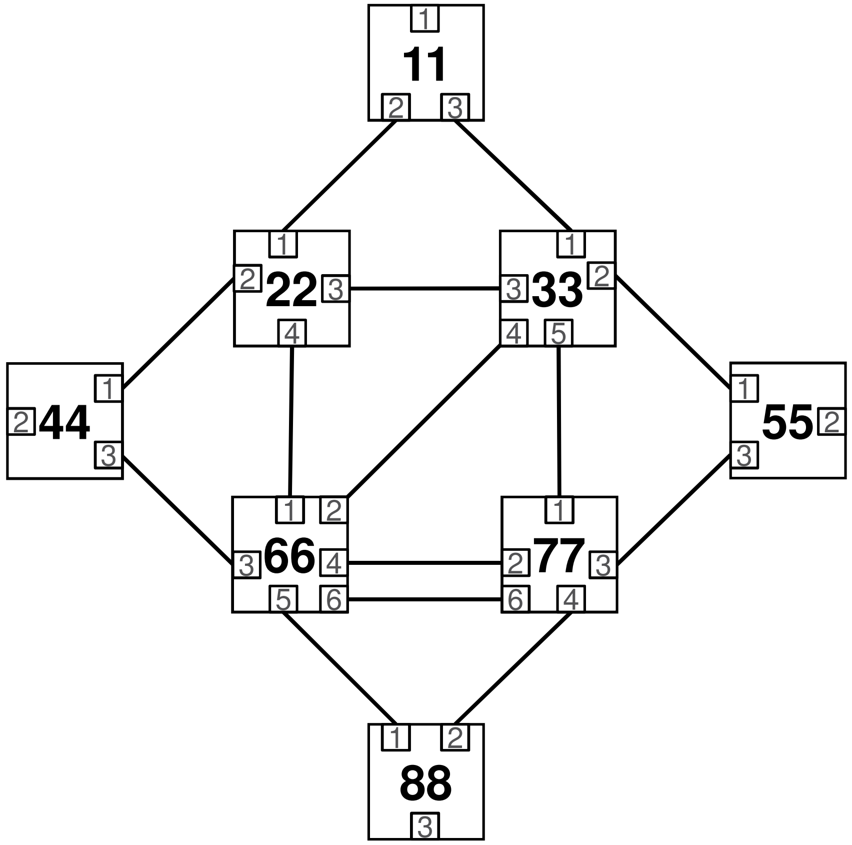
Getting the Trees

Topology Description

- Topology sub-TLV
- This 'translated' version is used in the following:

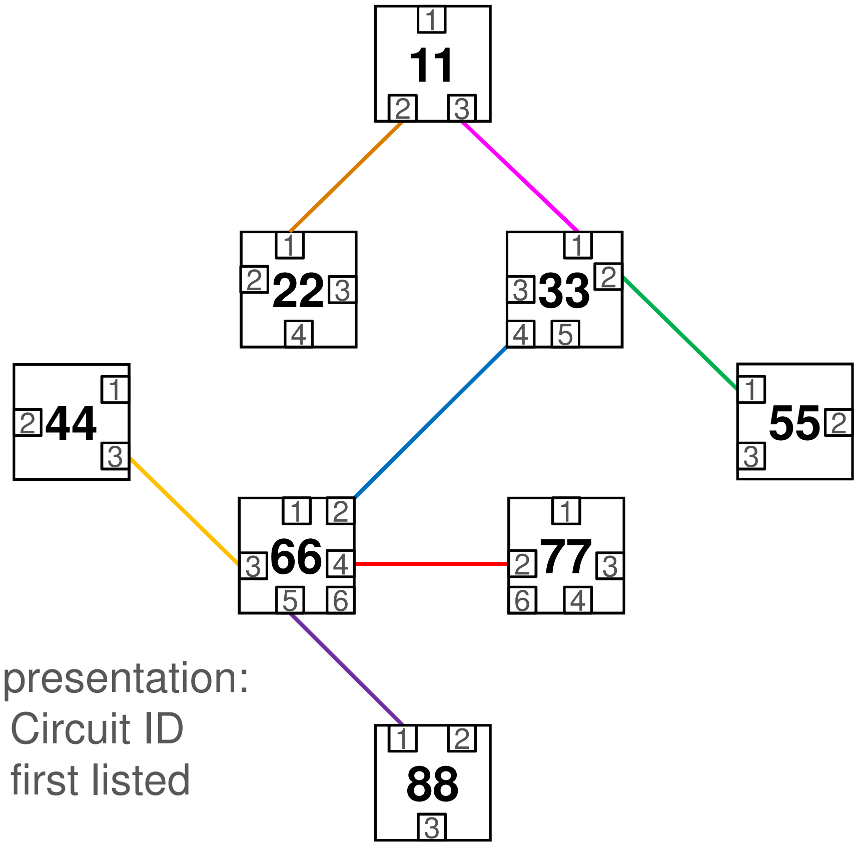


Example SPT Region Used in The Following



A Fully Specified Spanning Tree

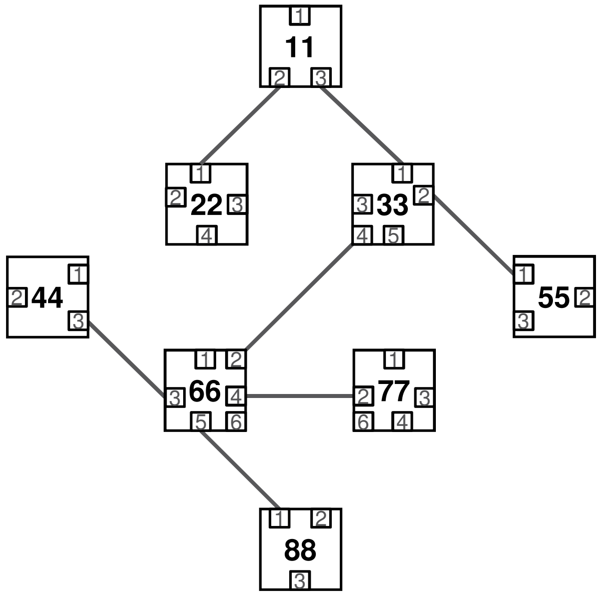
arbitrary order
11, 2; Circuit, End
11, 3; Circuit, End
44, 3; Circuit, End
55, 1; Circuit, End
88, 1; Circuit, End
33, 4; Circuit
66, 4; Circuit



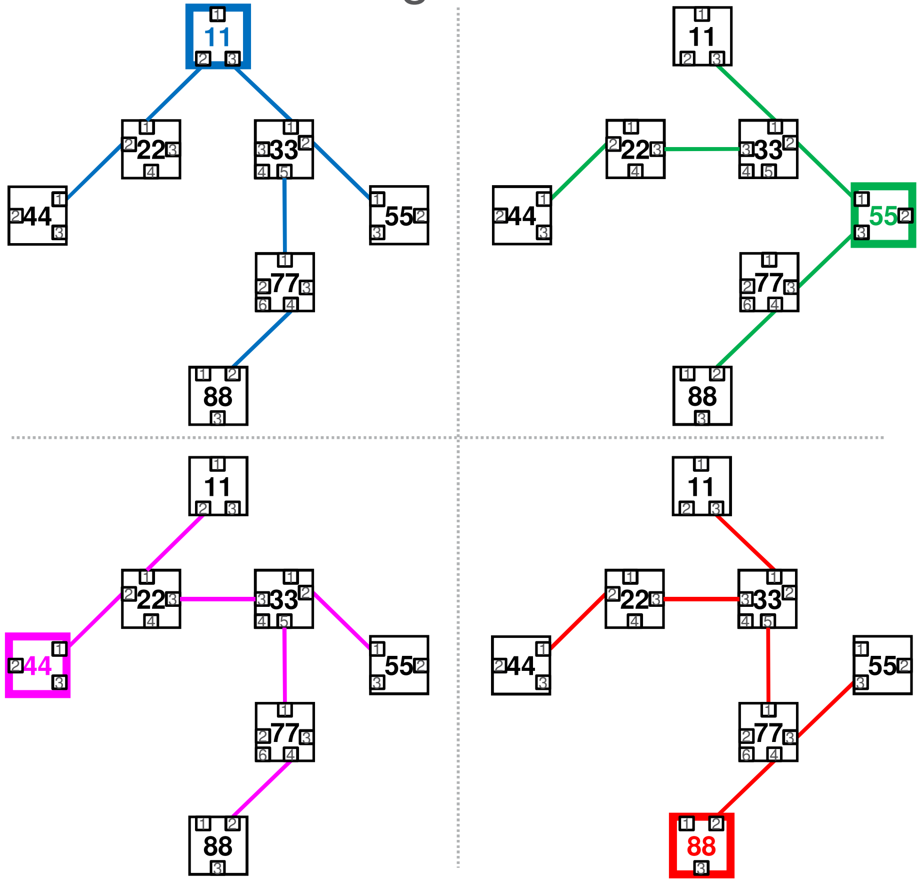
The order applied in this presentation:
Ascending in System ID, Circuit ID
such that End Points are first listed

Tree Structures

- **Ad-hoc tree**
- A single tree in an arbitrary structure, e.g.



- **Template trees**
- A set of trees following a template; e.g. each edge bridge roots an SPT such that Bridge 66 is excluded



- (802.1aq SPB template = each bridge roots an SPT)

Explicit ECT Algorithms

1. Static Explicit – SE ECT Algorithm
2. Loose Tree – LT ECT Algorithm
3. Loose Tree Set – LTS ECT Algorithm
4. Maximally Redundant Trees – MRT ECT Algorithm
5. Maximally Redundant Trees with GADAG – MRTG ECT Algorithm
6. Maximally Disjoint Paths – MDP ECT Algorithm

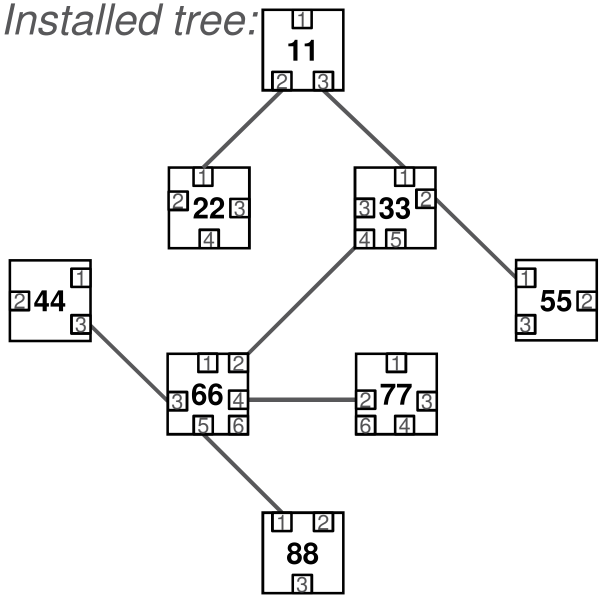
Static Explicit ECT Algorithm

- A single static explicit tree that does not contain any loose hops
  - This is the “fully nailed down” one
- The descriptor fully specifies the tree
  - → no loose hops
  - → no IS-IS update on its own → static
- The owner PCE can only update the tree
  - PCE has to detect topology change
  - PCE computes new tree
    - Algorithm is only the PCE’s business
  - PCE floods new descriptor
- SPT Bridges have no other task but install the appropriate FDB entries

Descriptor flooded by PCE:

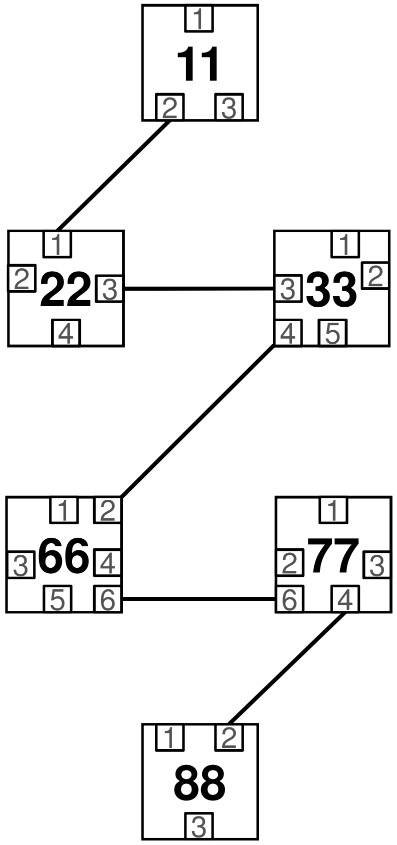
11, 2; Circuit, End
11, 3; Circuit, End
44, 3; Circuit, End
55, 1; Circuit, End
88, 1; Circuit, End
33, 4; Circuit
66, 4; Circuit

Installed tree:



Static Explicit ECT Algorithm – cont’d

- Exact order has to be followed if Circuit ID is not present in the descriptor of a p2p path
- Circuit ID is always used in case of multipoint-to-multipoint ET
- Circuit ID is always used in case of parallel links (e.g. 66 ⇔ 77)



arbitrary order

11, 2; Circuit, End
88, 2; Circuit, End
22, 3; Circuit
33, 4; Circuit
66, 6; Circuit

exact order

11; End
22
33
66, 6; Circuit
77
88; End

Loose Tree ECT Algorithm

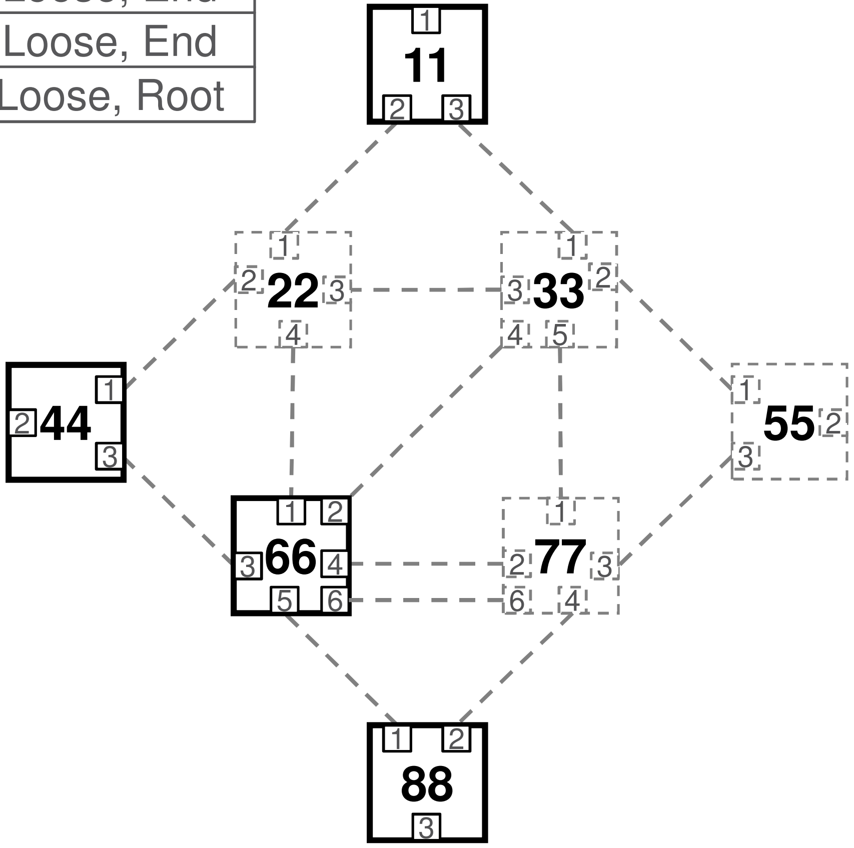
- A single explicit tree that includes one or more loose hops
- A loose multipoint-to-multipoint ET must always be entirely loose, i.e. the descriptor can only comprise the End Points; each of them is a loose hop
- BLCEs compute the tree
  - → Root has to be specified by the Topology sub-TLV
- Constrained routing is used if Topology sub-TLV conveys constraint, e.g. Admin Group or Exclude Hop
- Loose hops are restored by IS-IS
- Loose and strict hops can be mixed in a p2p path (as per D0.6)
- see examples in the following slides

Loose Tree ECT Algorithm
Example 1: A Completely Loose Tree

- The tree to span
  11, 44, 88, and 66;
  such that 66 is the Root

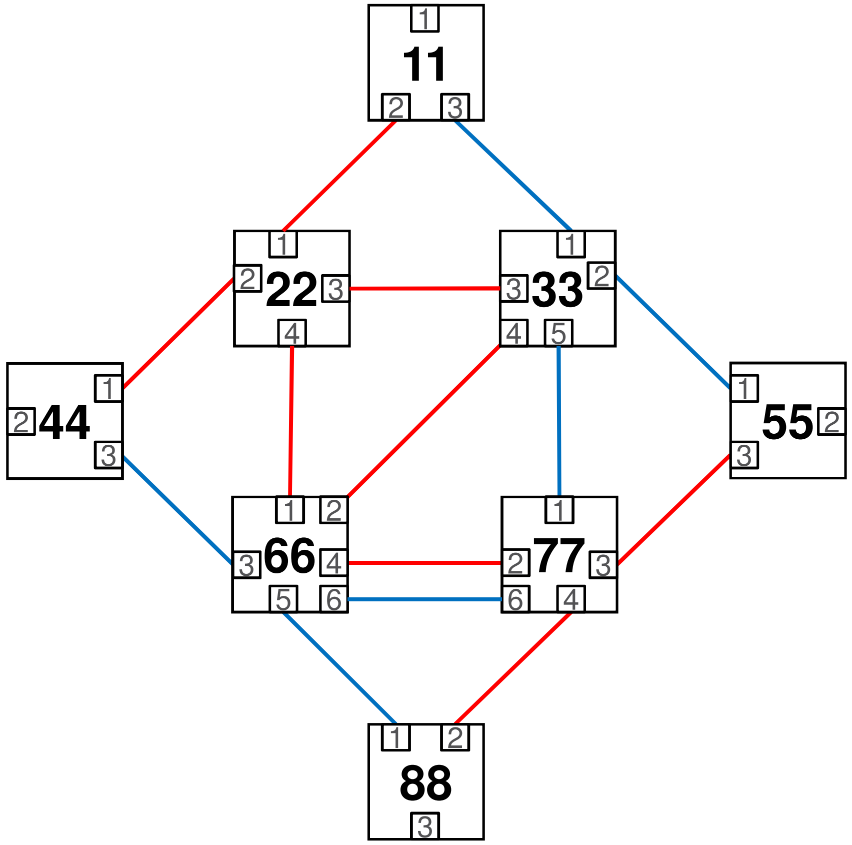
Descriptor flooded by PCE:

11; Loose, End
44; Loose, End
88; Loose, End
66; Loose, Root



Loose Tree ECT Algorithm
Example 2: Administrative Groups

- The color of the link represents the Administrative Group it belongs to

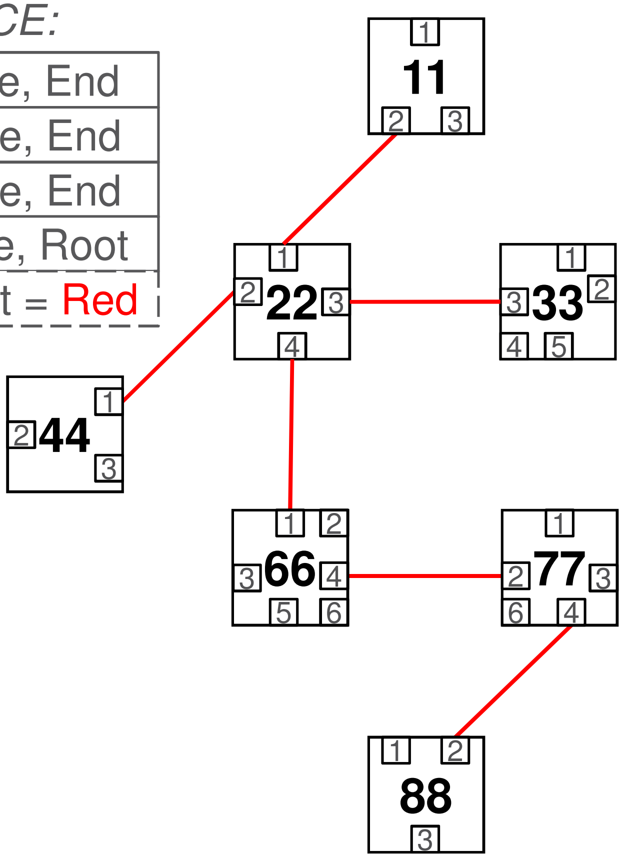


Loose Tree ECT Algorithm Example 2: Constrained Routing

- The Topology sub-TLV conveys an Administrative group sub-TLV (Type = 3), which specifies the Red group
- The descriptor specifies that the tree to span 11, 22, 44, 88, such that 22 is the Root

Descriptor flooded by PCE:

11; Loose, End
44; Loose, End
88; Loose, End
22; Loose, Root
Constraint = Red



Loose Tree Set ECT Algorithm

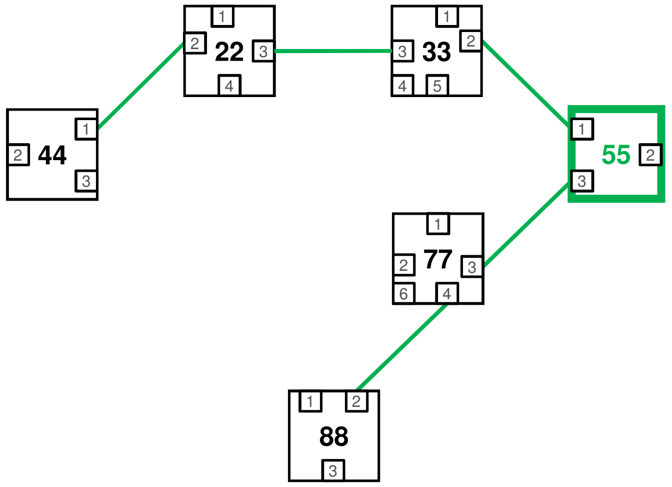
- A set of completely loose explicit trees, which set comprises an individual tree for each end point specified by the descriptor of the explicit tree
- Each tree is computed by the BLCE of SPT Bridges
- Each tree is restored by IS-IS in case of a topology change
- These are template trees
- The LTS ECT Algorithm can be used
  - If only a subset of edge bridges are to be connected by template trees
  - If the template trees are not SPTs because a constraint has to be applied on them, e.g. Admin Group or Exclude Hop

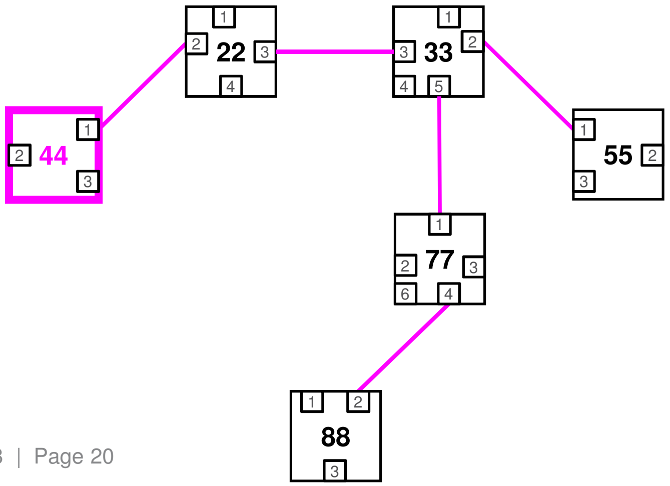
LTS ECT Algorithm Example: Excluding a Bridge

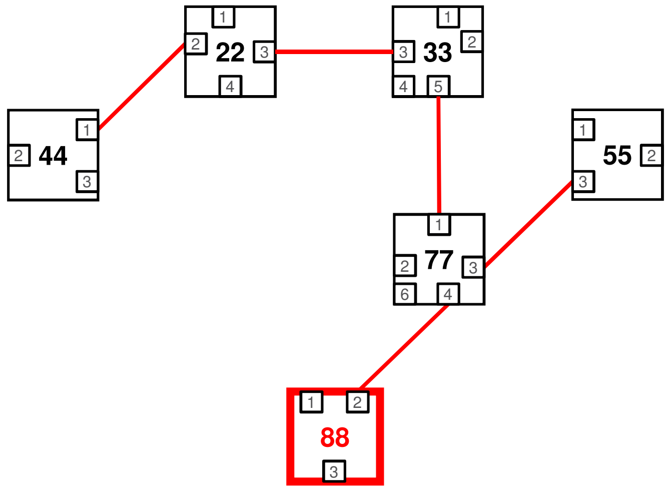
- Bridge 66 is an Exclude Hop

Descriptor flooded by PCE:

44; Loose, End
55; Loose, End
88; Loose, End
66; Exclude







Maximally Redundant Trees ECT Algorithm

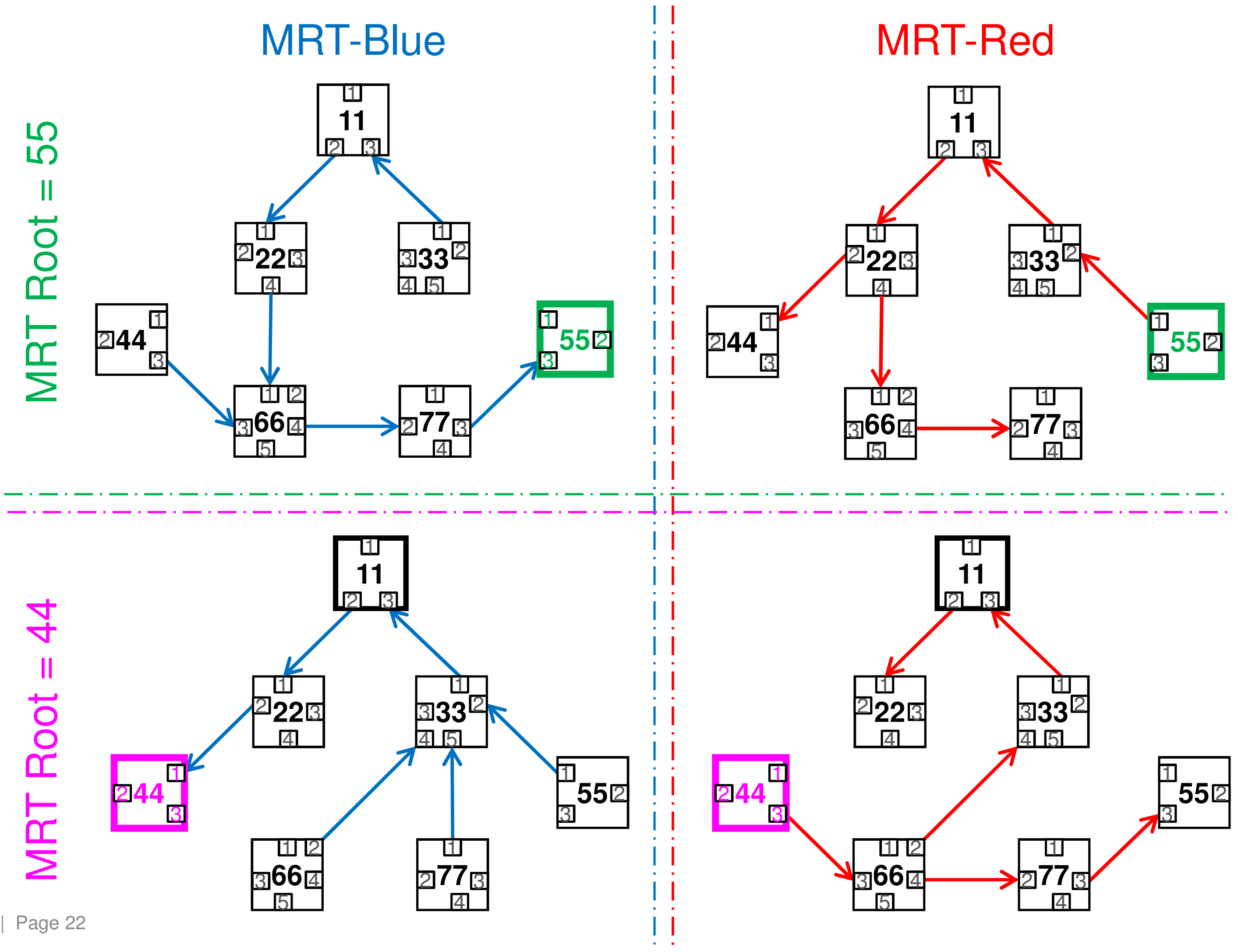
- Maximally Redundant Trees (MRTs) are completely loose trees for each MRT Root
- The MRTs are computed together with the corresponding GADAG by the BLCE of SPT Bridges
  - → Completely distributed operation
- MRTs are cautiously restored by ISIS-PCR
- Two options
  1. Each SPT Root is an MRT Root as well
     - No Topology sub-TLV; in fact no 802.1Qca sub-TLV
     - **Base VID is associated with the MRT ECT Algorithm** in the SPB Base VLAN-Identifiers sub-TLV; and that's all
  2. MRT Roots are specified by Topology sub-TLV
- This is Mode A of http://www.ieee802.org/1/files/public/docs2014/ca-farkas-mrt-0114-v01.pdf

MRT ECT Algorithm Example: MRT Roots Specified

- MRT Roots:
  - 44 and 55
- 88 is not included

Descriptor flooded by PCE:

44; Loose, End, MRT Root
55; Loose, End, MRT Root
88; Exclude



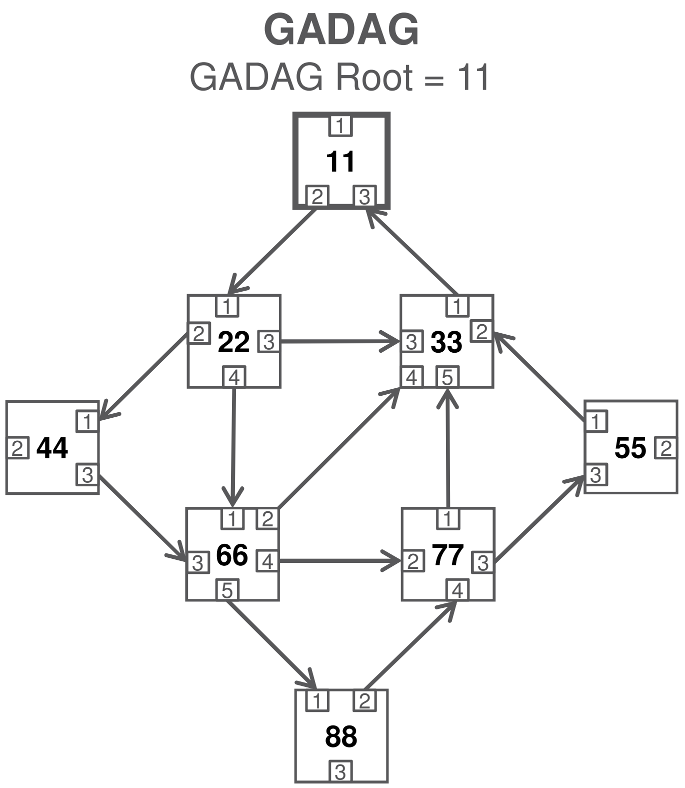
Maximally Redundant Trees with GADAG ECT Algorithm

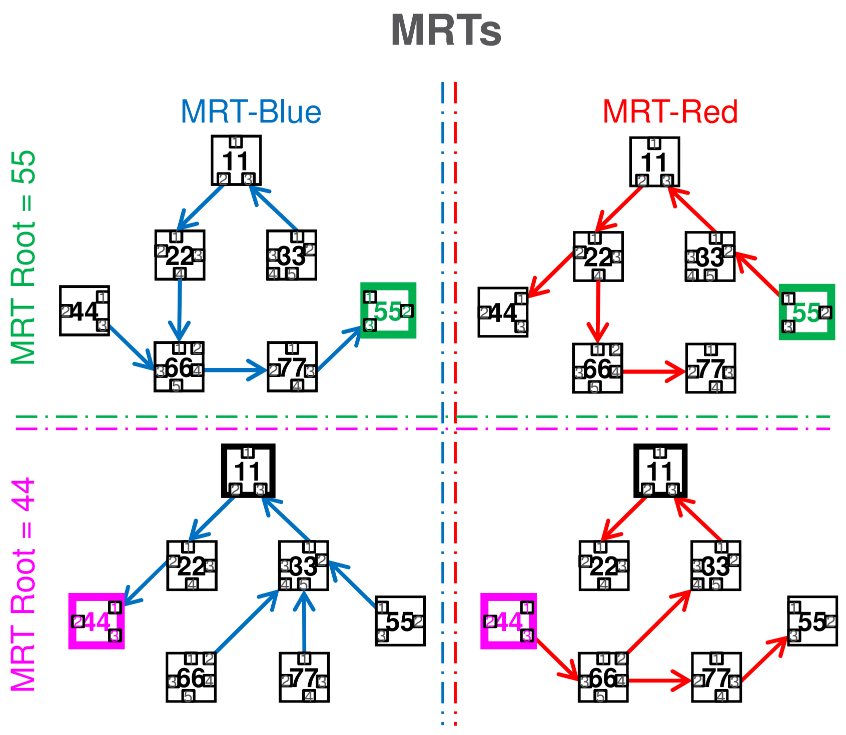
- GADAG is computed centrally by GADAG Computer, e.g. PCE
  - → Centralized GADAG computation
- GADAG Computer floods GADAG descriptor
  - MRT Roots are also specified by the Topology sub-TLV specifying the GADAG
- MRTs are then computed by the BLCE of SPT Bridges based on the GADAG
  - → Distributed MRT Computation
- MRTs are cautiously restored upon reception of a new GADAG from the GADAG Computer
- This is Mode B of http://www.ieee802.org/1/files/public/docs2014/ca-farkas-mrt-0114-v01.pdf
  - (Mode C can be implemented by the Static Explicit ECT Algorithm)

MRTG ECT Algorithm Example

Descriptor flooded:

11, 2; Circuit, End GADAG Root
22, 2; Circuit
22, 3; Circuit
22, 4; Circuit
33, 1; Circuit
44, 3; Circuit, End MRT Root
55, 3; Circuit, End MRT Root
66, 2; Circuit
66, 4; Circuit
66, 5; Circuit
77, 1; Circuit
77, 3; Circuit
88, 2; Circuit





Maximally Disjoint Paths ECT Algorithm

- Maximally Disjoint Paths (MDPs) are a pair of point-to-point paths
- The paths are computed as specified by 45.3.5
- The loose hops are cautiously restored by IS-IS

Getting the VIDs

VID Direction

- A VID can be made associated with a particular explicit tree by the inclusion of the corresponding VLAN Tag in the Topology sub-TLV (preceding the Hop sub-TLVs)

- Each VID is bidirectional by default
  - Each End Point bridge both Transmits (T) and Receives (R) on a VID
  - It is the default behavior → No filed for it in the sub-TLVs

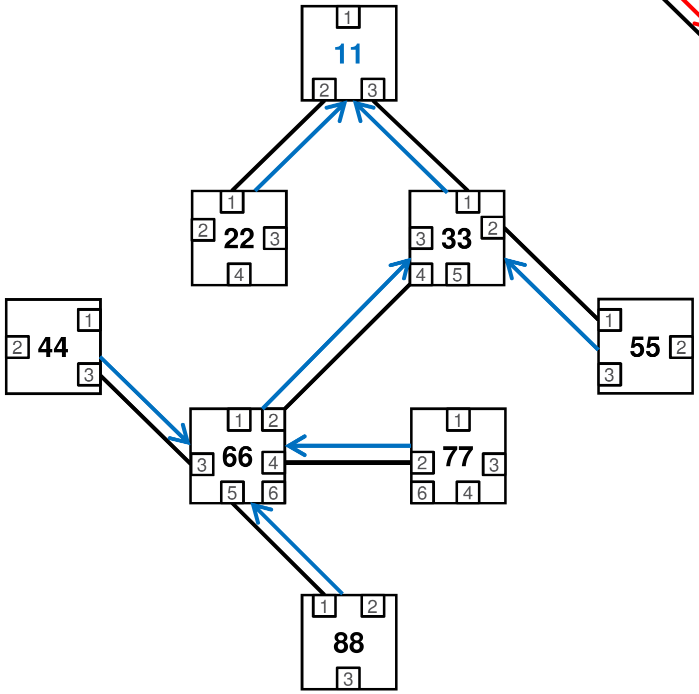
- Different behavior can be configured by setting the VIDs T/R flags in the Hop sub-TLV of the End Point bridge

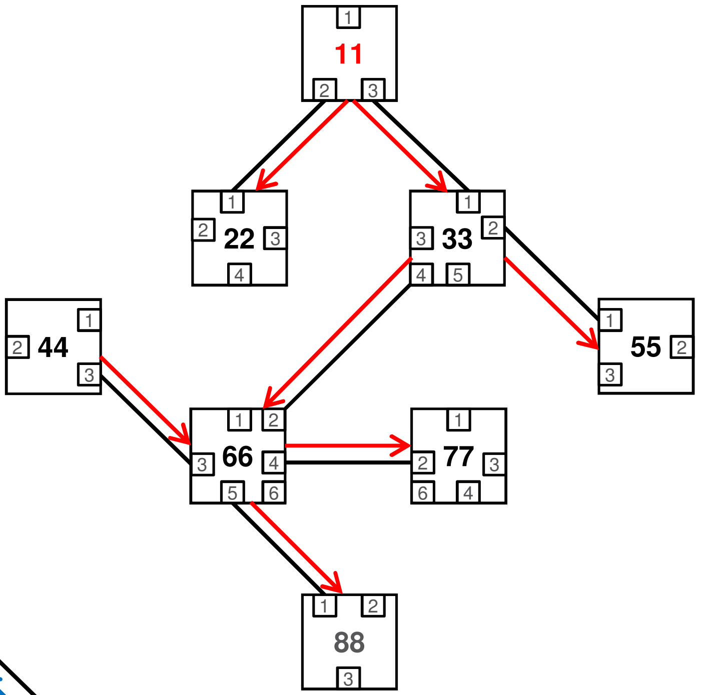
Directed VIDs Example

- VID1 is directed to 11
- VID2 is directed from 11

Descriptor flooded:

11, 2; Circuit, End	VID1: R	VID2: T
11, 3; Circuit, End	VID1: R	VID2: T
44, 3; Circuit, End	VID1: T	VID2: R
55, 1; Circuit, End	VID1: T	VID2: R
88, 1; Circuit, End	VID1: T	VID2: R
33, 4; Circuit		
66, 4; Circuit		





ERICSSON

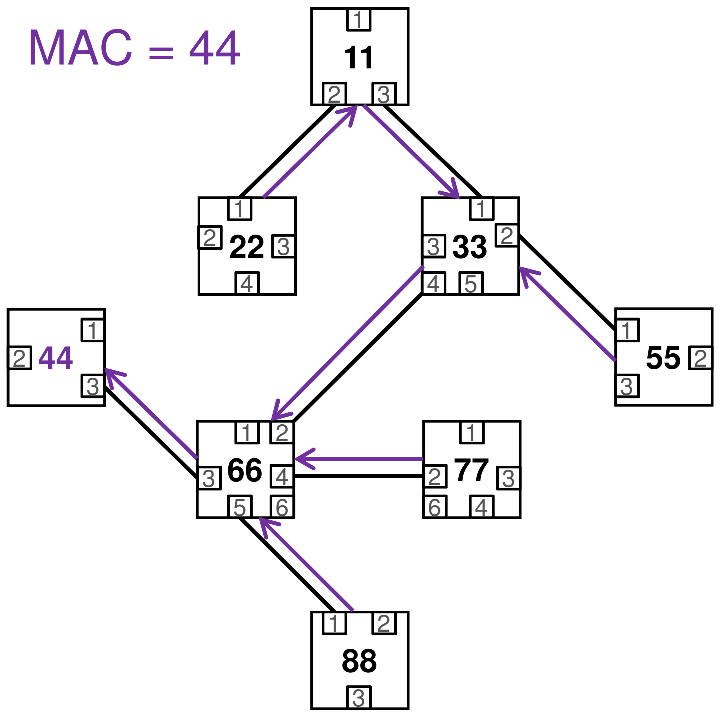
Getting the MACs

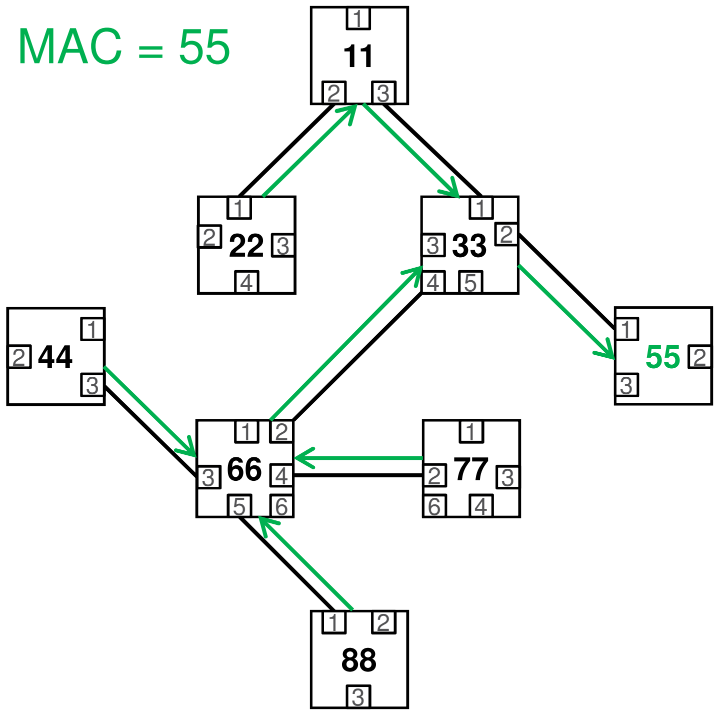
MAC Gives Direction

- Learning VID
  - VID → SPBV-MSTID
  - MAC learnt from data frames
- Non-learning VID
  - VID → SPBM-MSTID
  - MAC associated with a VID is learnt from SPBV MAC Address sub-TLV
  - MAC associated with an I-SID is learnt from SPBM Service Identifier and Unicast sub-TLV

Directed by MAC Example

- The topology provided by the FDB entries to an Individual MAC is a destination rooted tree within the region (irrespectively of the means the bridges become aware of the location of the MAC)





ERICSSON

Summary

It Is Simple

- A very few pieces (= IS-IS TLVs) of the puzzle provide the full picture!
- SPT Bridge declares:
  - VID for explicit path control (VID → an explicit ECT Algorithm in the SPB Base VLAN-Identifiers sub-TLV)
  - MACs it Transmits / Receives
    - VID scope: SPBV MAC Address sub-TLV
    - I-SID scope: SPBM Service Identifier and Unicast sub-TLV
- PCE provides the Explicit Tree for the VID (Topology sub-TLV)
- Brides get all this information → install FDB entries

Background

Reading

- P802.1Qca Path Control and Reservation (PCR)
  - http://www.ieee802.org/1/pages/802.1ca.html
  - Draft 0.6: http://www.ieee802.org/1/files/private/ca-drafts/d0/802-1Qca-d0-6.pdf
  - Tutorial on Draft 0.4: http://www.ieee802.org/1/files/public/docs2013/ca-farkas-d0-4-operation-v01.pdf
- IEEE 802.1aq Shortest Path Bridging (SPB)
  - 802.1Qca builds upon the architecture and concepts specified by SPB and uses some SPB sub-TLVs (see subclause 5.4.6 of Qca);
    however, full SPB implementation is not required for Qca
  - http://standards.ieee.org/getieee802/download/802.1aq-2012.pdf
  - http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118148665.html
  - http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5594687
  - http://en.wikipedia.org/wiki/IEEE_802.1aq
- IEEE 802.1Q (802.1Qca is an amendment to 802.1Q)
  - 802.1Q-2011: http://standards.ieee.org/getieee802/download/802.1Q-2011.pdf
  - 802.1Q-REV: http://www.ieee802.org/1/pages/802.1Q-rev.html
  - Tutorials: http://www.ieee802.org/802_tutorials/2013-03/8021-IETF-tutorial-final.pdf
    - http://www.ieee802.org/1/files/public/docs2014/Q-farkas-SDN-support-0314-v01.pdf