ERICSSON

P802.1Qca D1.1 Tutorial

Explicit Path Control

János Farkas
janos.farkas@ericsson.com

November 4, 2014

Preface

- This presentation is an update of http://www.ieee802.org/1/files/public/docs2014/ca-farkas-d0-8-tutorial-0714-v02.pdf
  - Note that this presentation is also in-line with http://www.ieee802.org/1/files/public/docs2013/ca-farkas-d0-4-operation-v01.pdf
- D1.1 is still not the final standard!

Outline

- Introduction
- Explicit Trees
  - Tree structures
  - Explicit ECT Algorithms
- Getting the trees
- Getting the VIDs
- Getting the MACs
- Summary
- Background

Presentation Objectives

- Explore the operation of explicit tree establishment as described in P802.1Qca D1.1 through examples
- Focus on the Explicit ECT Algorithms
- Explore the features provided

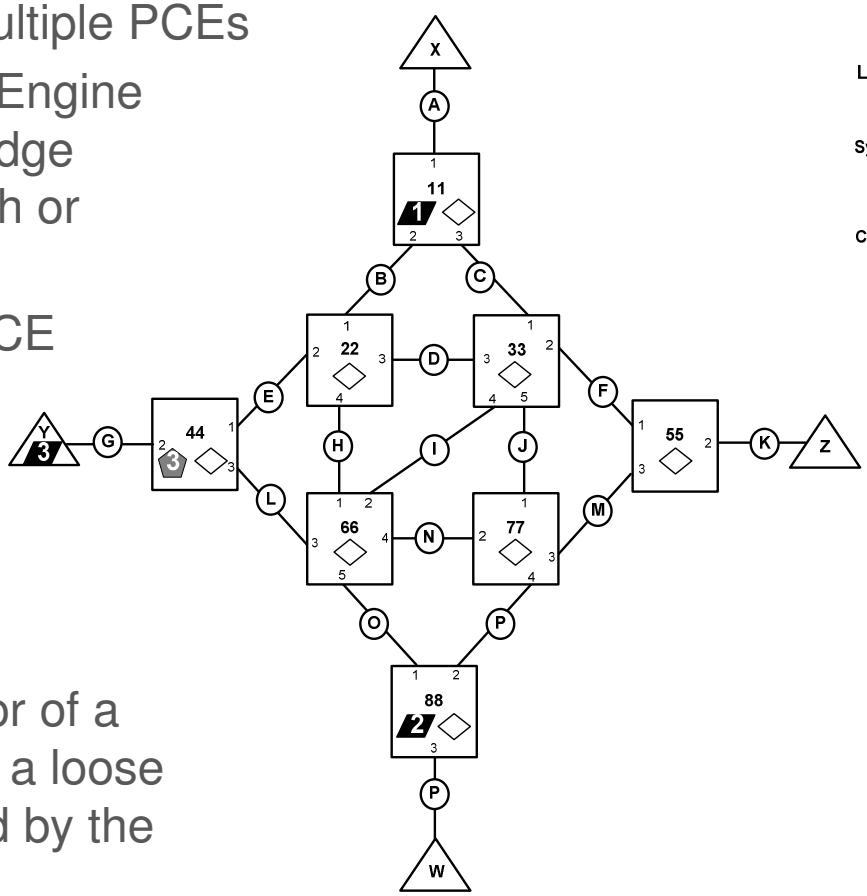
Highlights

- 802.1Qca is an extension to IS-IS
- It is control plane
- Main goal: establishment of explicit trees
  - 802.1Qca D1.1 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
  - A PCE may use a Path Control Agent (PCA) for the control of ETs via IS-IS, e.g. PCE 3 – PCA 3
- 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 strict or loose
  - strict and loose cannot be mixed
- A strict ET is computed and described by its owner PCE, and then installed by IS-IS
- A PCE provides the descriptor of a loose tree where each hop is a loose hop; the ET is then computed by the BLCEs, installed by IS-IS



Getting the Trees

Topology Description: Ordered List of System IDs

Topology sub-TLV (Figure 45-5)

- Type
- Length
- # Base VIDs
- Base VID 1
- …
- Base VID n
- Hop sub-TLV 1
- Hop sub-TLV 2
- …
- Hop sub-TLV i
- …
- Hop sub-TLV m
- Constraint sub-TLV opt.

Hop sub-TLV (Figure 45-8)

- Type
- Length
- ***Flags*** — 1 byte
- **System ID 1** — 6 bytes
- *Circuit ID 1* — 4 bytes
- *Further Opt. fields*

Circuit ID is only used if parallel links have to be distinguished

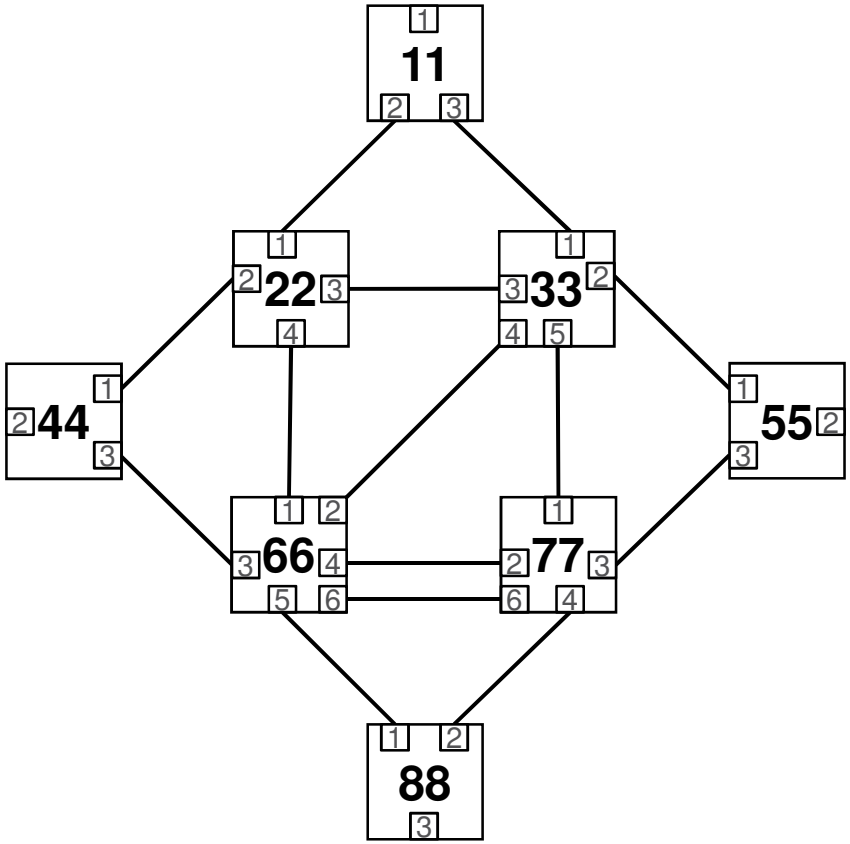
- This ‘transformed’ version is used in the following:

System ID 1, Circuit ID 1; ***Flags Set***	Opt.
System ID 2, Circuit ID 2; ***Flags Set***	Opt.
…	
System ID i, Circuit ID i; ***Flags Set***	Opt.
…	
System ID m, Circuit ID m; ***Flags Set***	Opt.
Constraint	

1-bit ***Flags***:

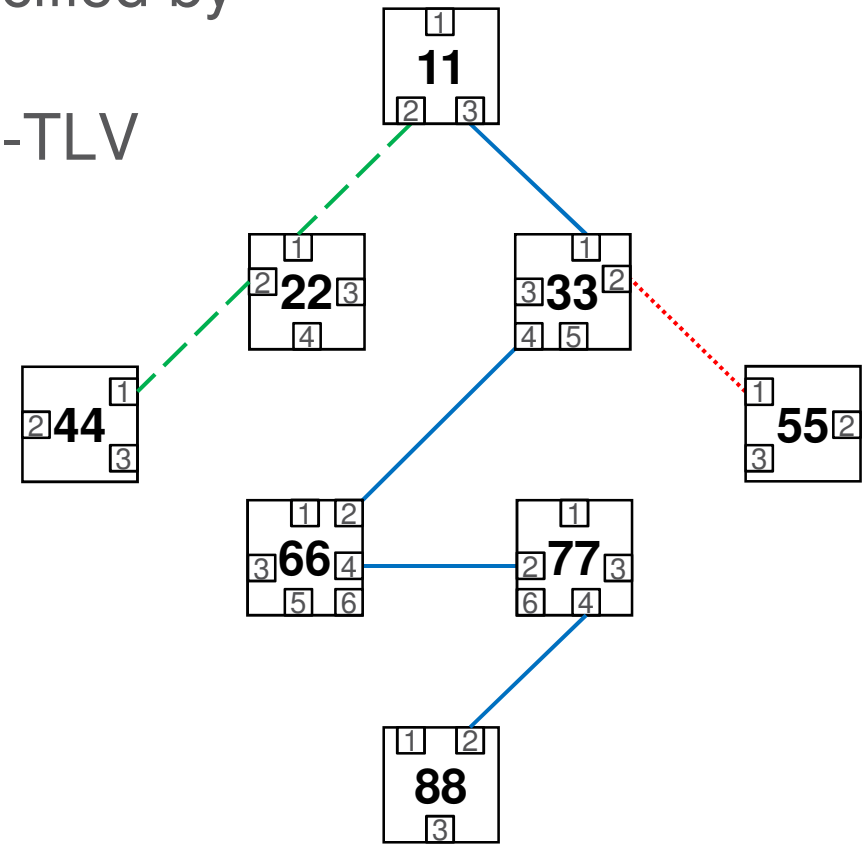
reserved	reserved	Exclude	Leaf	Root	Traffic End Point (TEP)	VID	Circuit
		Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1

Example SPT Region Used in The Following



A Strict Spanning Tree

- Each hop of a strict explicit tree is exactly specified by its descriptor Topology sub-TLV



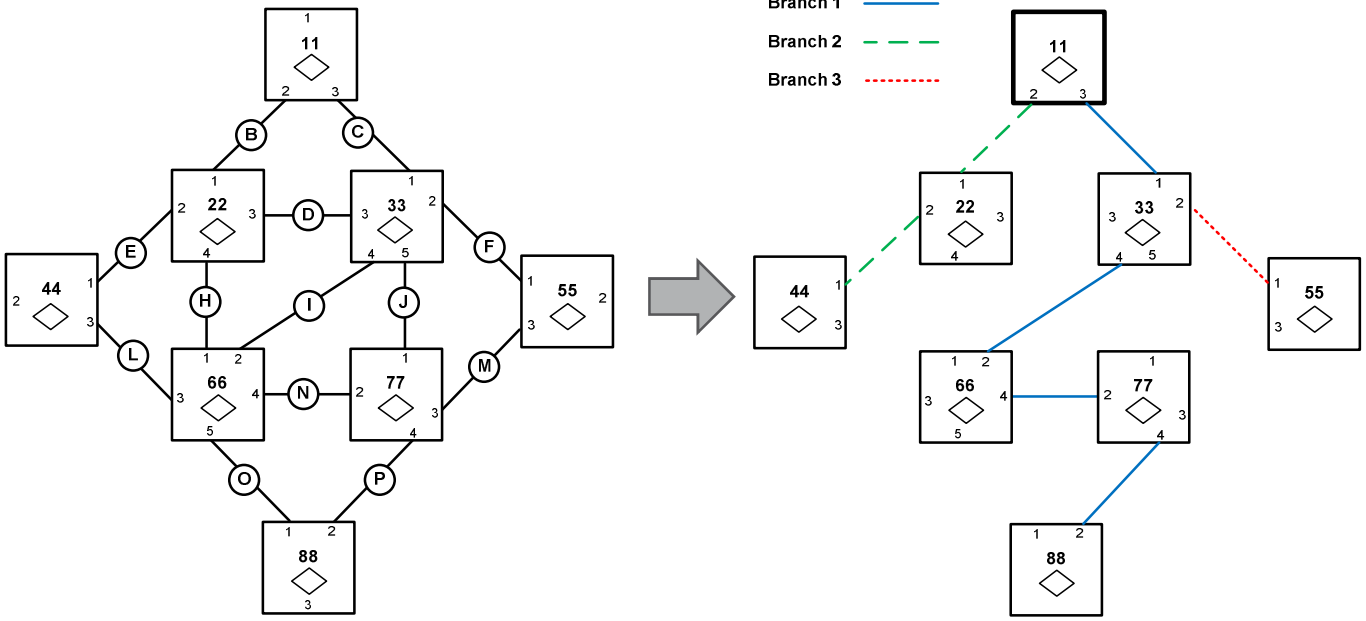
Descriptor

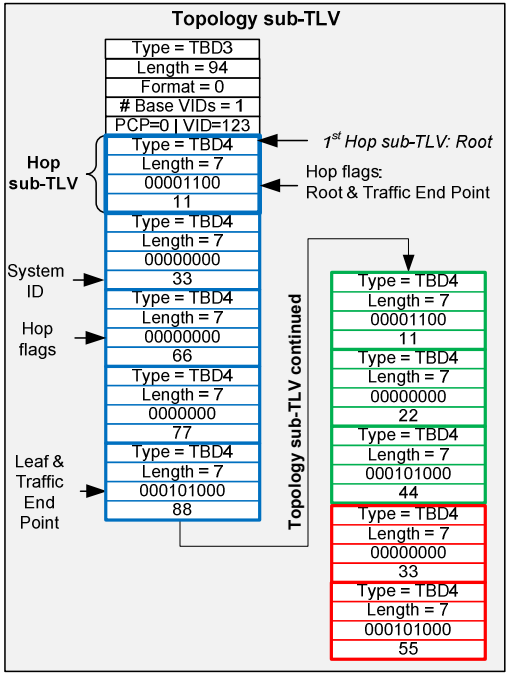
11; *Root, TEP*
33
66, 4; *Circuit*
77
88; *Leaf, TEP*
11; *Root, TEP*
22
44; *Leaf, TEP*
33
55; *Leaf, TEP*

Branch 1
Branch 2
Branch 3

Figure 45-6 of D1.1 Shows a Strict Spanning Tree

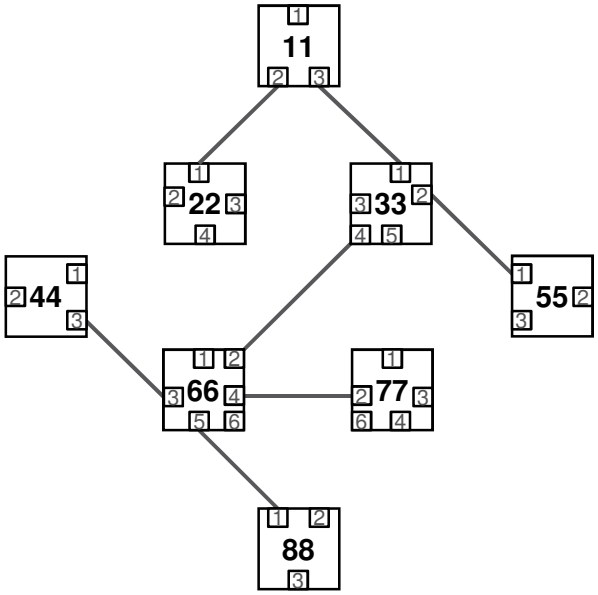
- There are no parallel links in the example topology used in D1.1 (→Circuit ID is not needed), which is the only difference compared to the above example



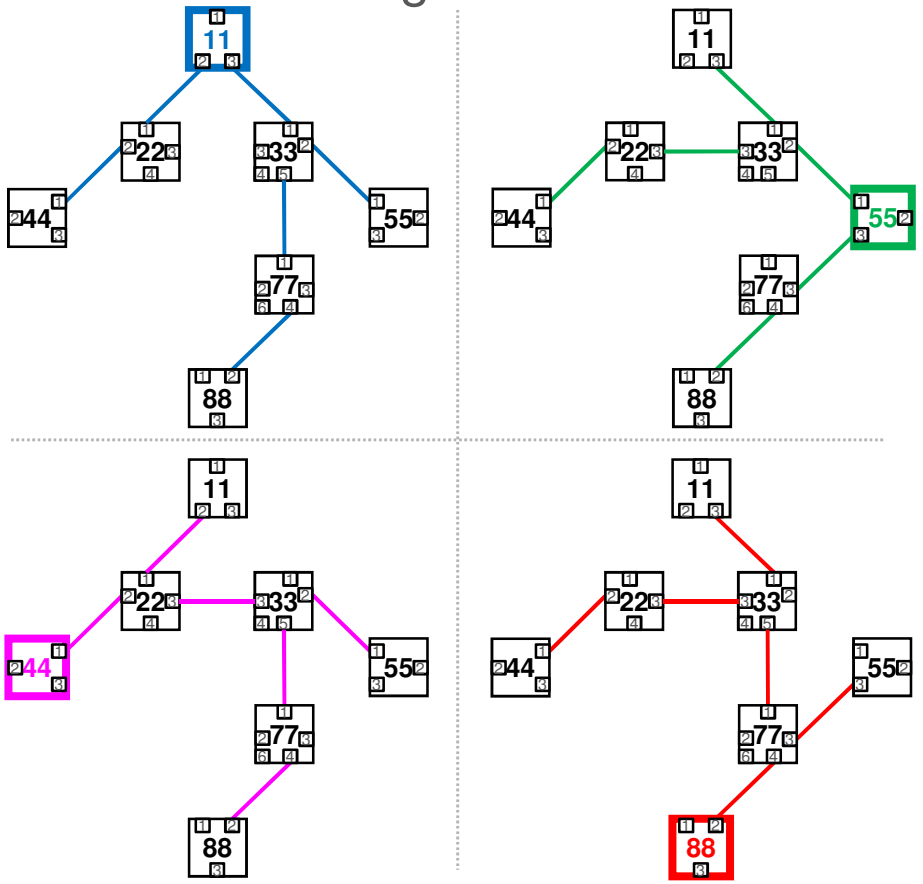


Tree Structures

- **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
Table 45-1

1. Strict Tree – ST 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

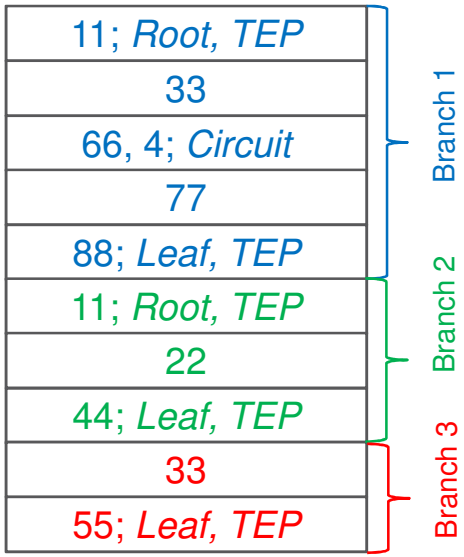
Strict Tree ECT Algorithm

- A single strict explicit tree
  - This is the “fully nailed down” one
- All computation is done by PCE
- 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 (appropriate for simple devices)

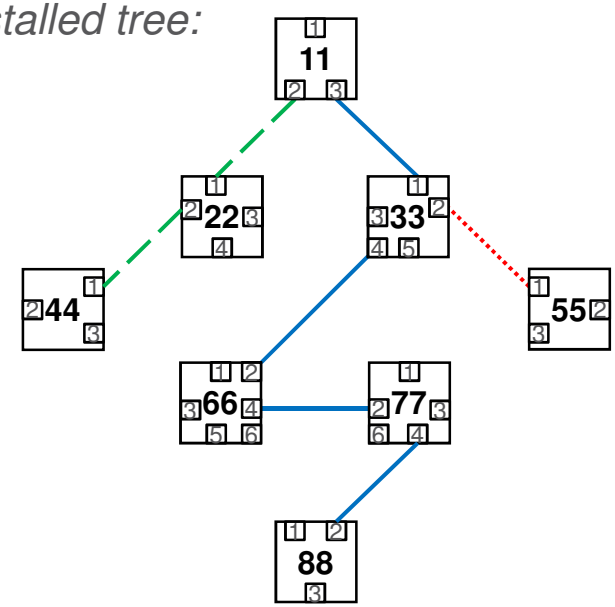
Strict Tree ECT Algorithm – cont'd

- Branch decomposition
- Each branch is specified by an ordered list of hops
- The first hop is the Root
- Circuit ID is only used in case of parallel links (e.g. 66 ⇔ 77)

Descriptor:



Installed tree:



Loose Tree ECT Algorithm

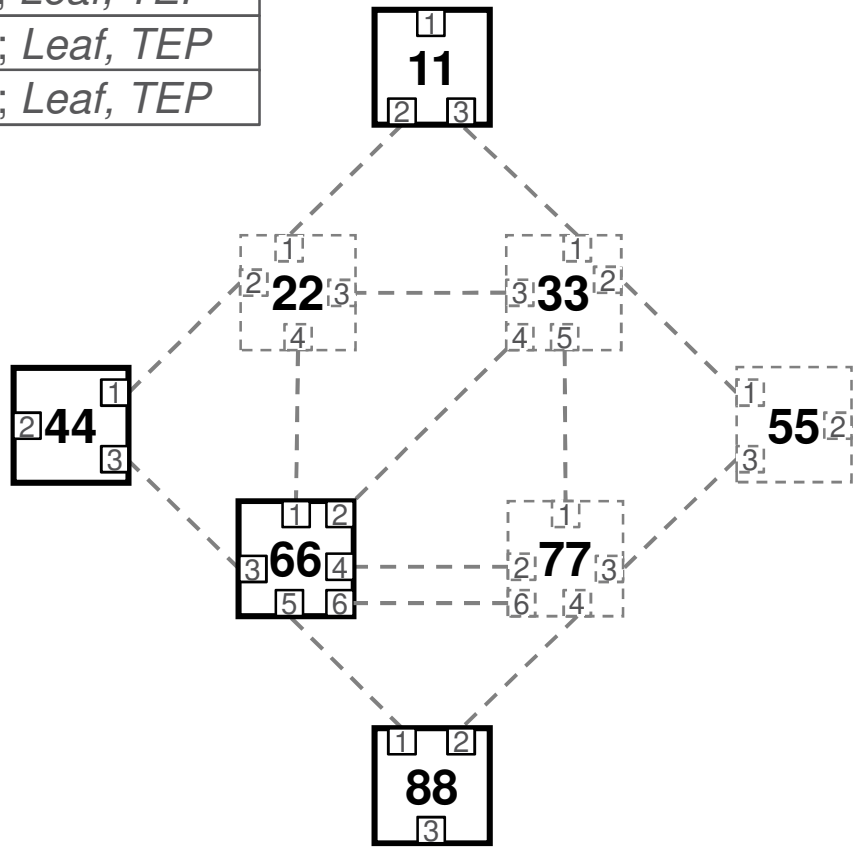
- A single loose explicit tree
- A loose explicit tree is always entirely loose, i.e. each hop is a loose hop
  - If at least a hop of an explicit tree is loose, then each hop is considered loose
  - The Root and the Leaves are specified by the Topology sub-TLV
  - Traffic End Points are specified by the Topology sub-TLV
  - A bridge to be excluded can be specified by the Exclude flag
- BLCEs compute the tree
- Constrained routing is used if Topology sub-TLV conveys constraint, e.g. Admin Group or Exclude
- Loose trees are restored by IS-IS
- see examples in the following slides

Loose Tree ECT Algorithm Example 1: Intermediate Root

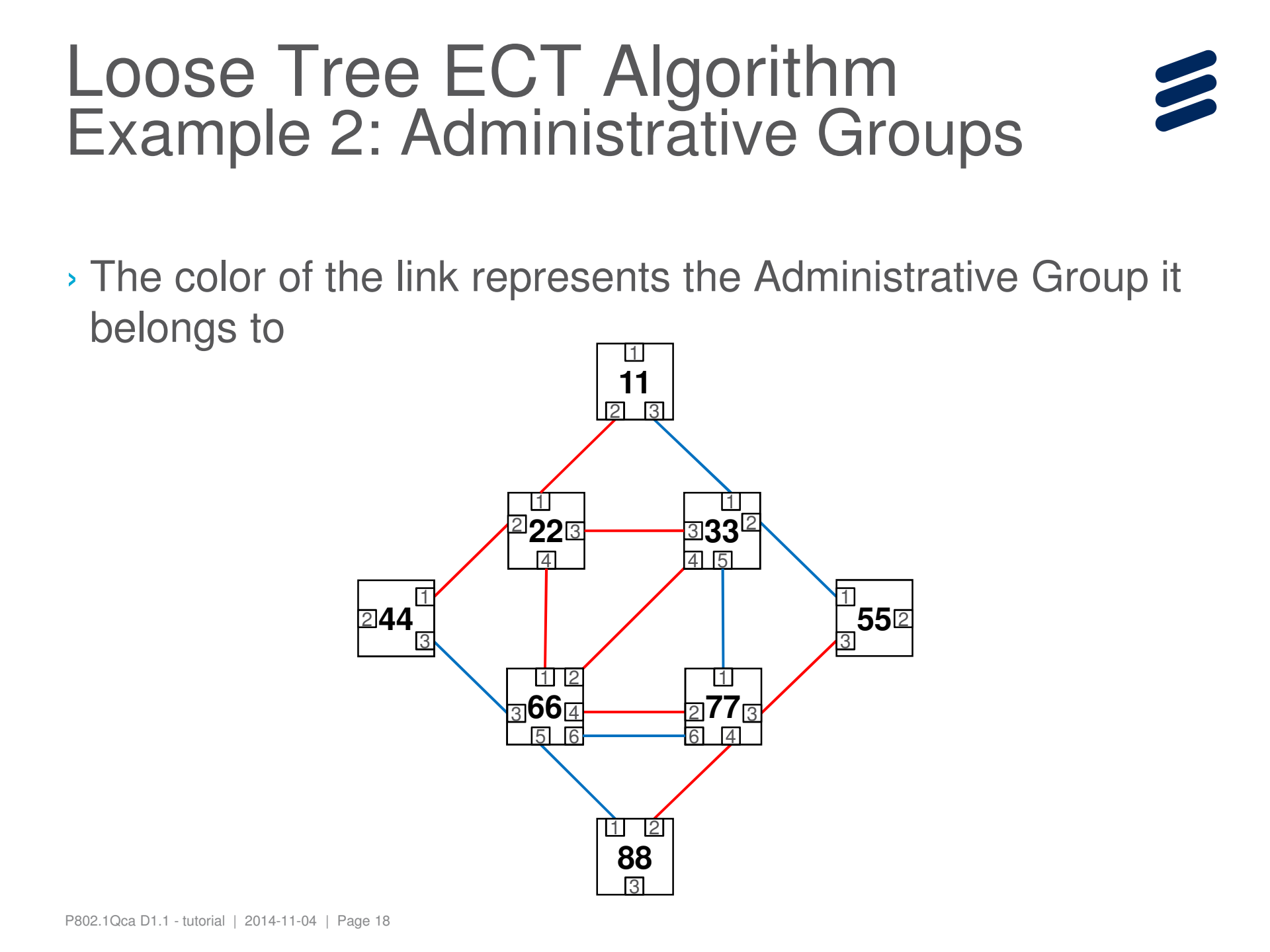
- The tree to span 11, 44, 88, and 66; such that 66 is the Root
- The first hop is the Root in the Topology sub-TLV
- The Root is not a traffic end point in this example

Descriptor:

66; *Root*
11; *Leaf, TEP*
44; *Leaf, TEP*
88; *Leaf, TEP*



Loose Tree ECT Algorithm
Example 2: Administrative Groups



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

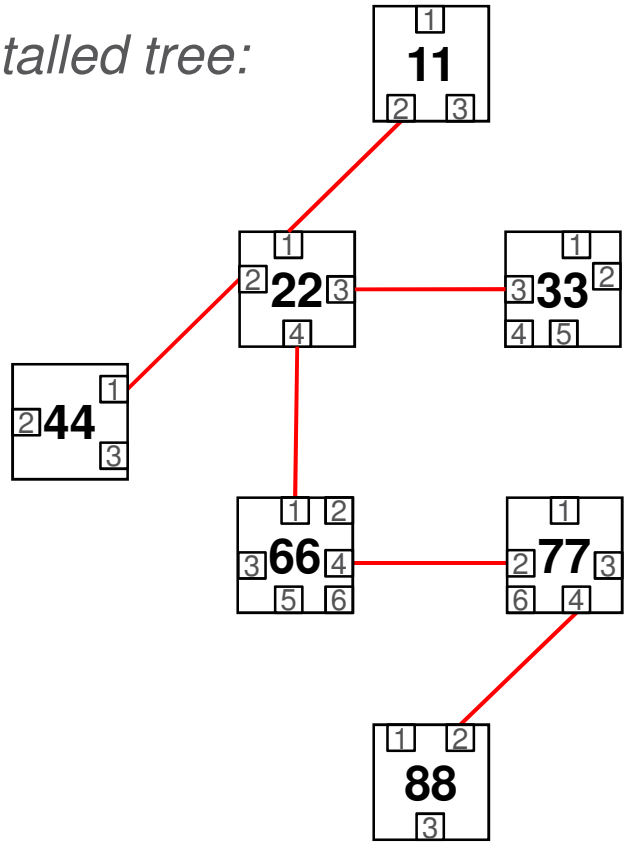
Loose Tree ECT Algorithm
Example 2: Constrained Routing

- 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:

22; *Root*
11; *Leaf, TEP*
44; *Leaf, TEP*
88; *Leaf, TEP*
Constraint = Red

Installed tree:



Loose Tree Set ECT Algorithm

- A set of loose explicit trees
  - an individual tree for each traffic end point, i.e.
  - each traffic end point roots its own 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

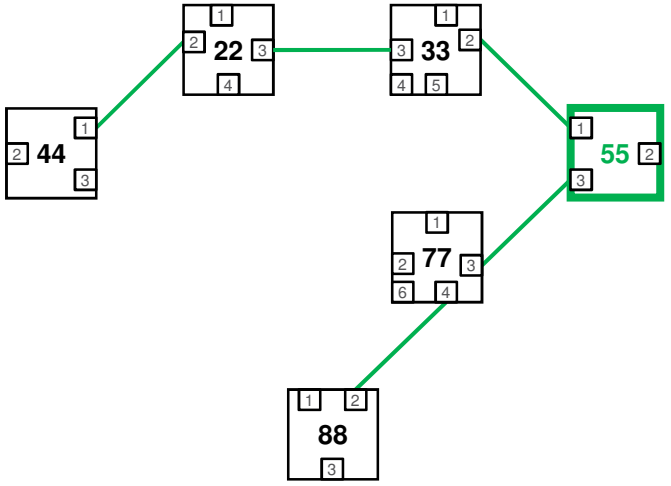
LTS ECT Algorithm
Example: Excluding a Bridge

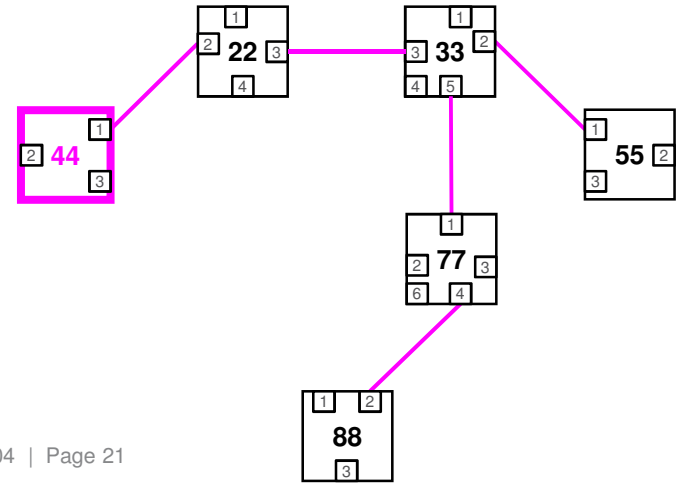
- Each traffic end point roots its own tree
- Bridge 66 is an Exclude Hop

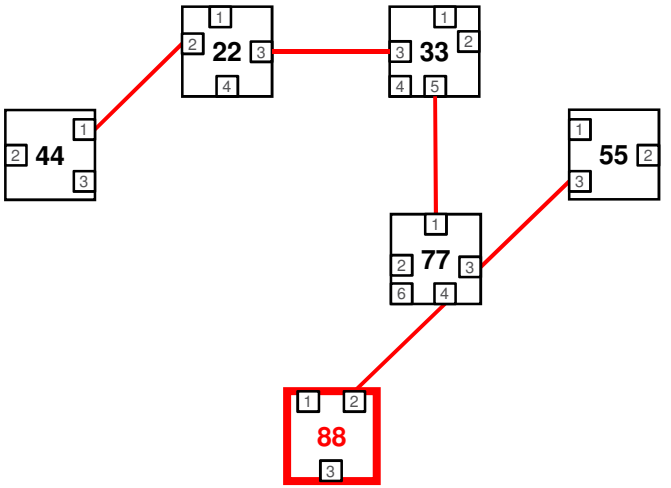
Descriptor:

44; *TEP*
55; *TEP*
88; *TEP*
66; *Exclude*

Installed trees:







Maximally Redundant Trees ECT Algorithm

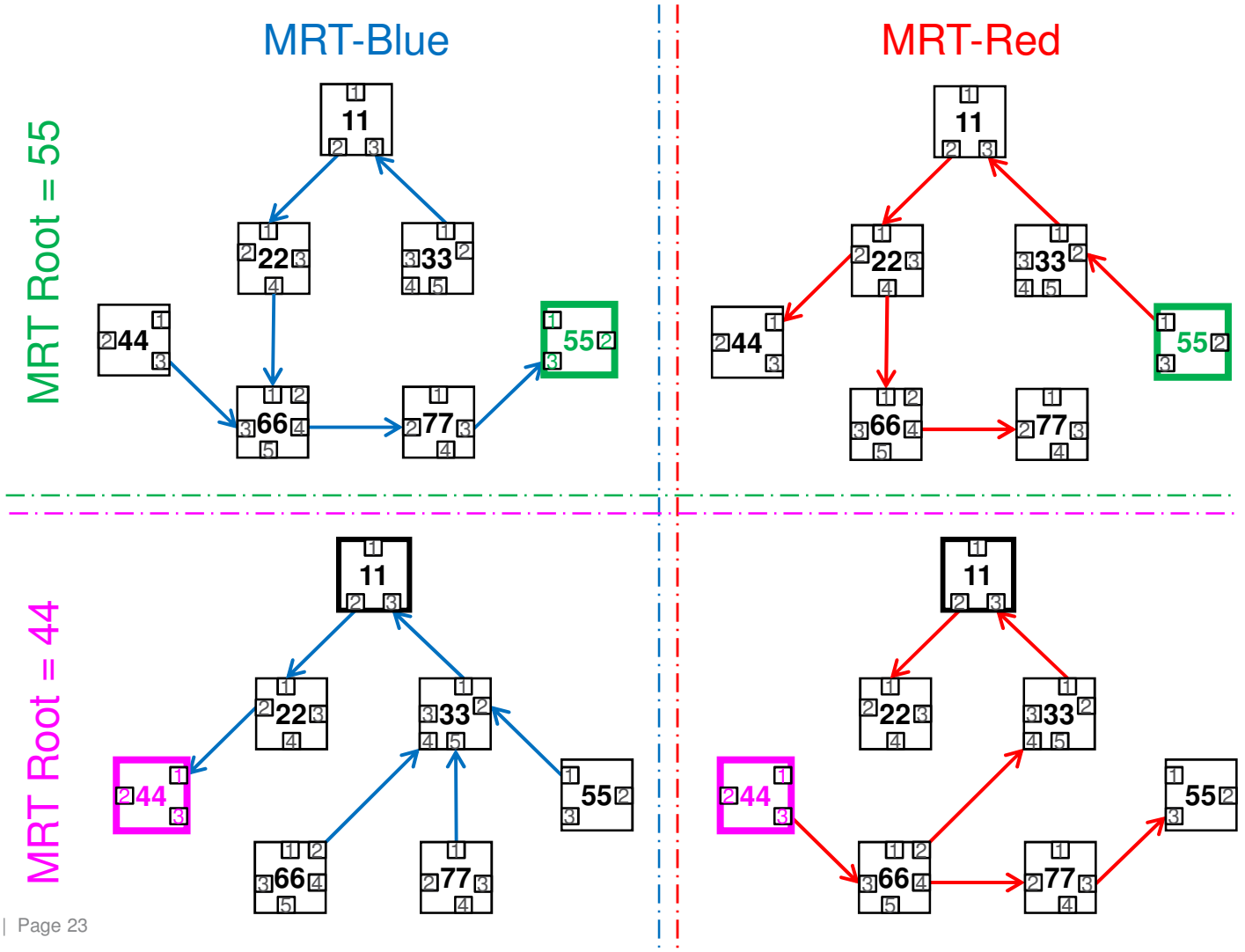
- Maximally Redundant Trees (MRTs) are loose trees; each MRT Root roots both an MRT-Blue and an MRT-Red
- The MRTs are computed together with the corresponding GADAG by the BLCE of SPT Bridges
  - → Completely distributed operation
- MRTs are cautiously restored by ISIS-PCR
- Two options
  1. Each SPT 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 excluded

Descriptor:

44; *Root, TEP*
55; *Root, TEP*
88; *Exclude*

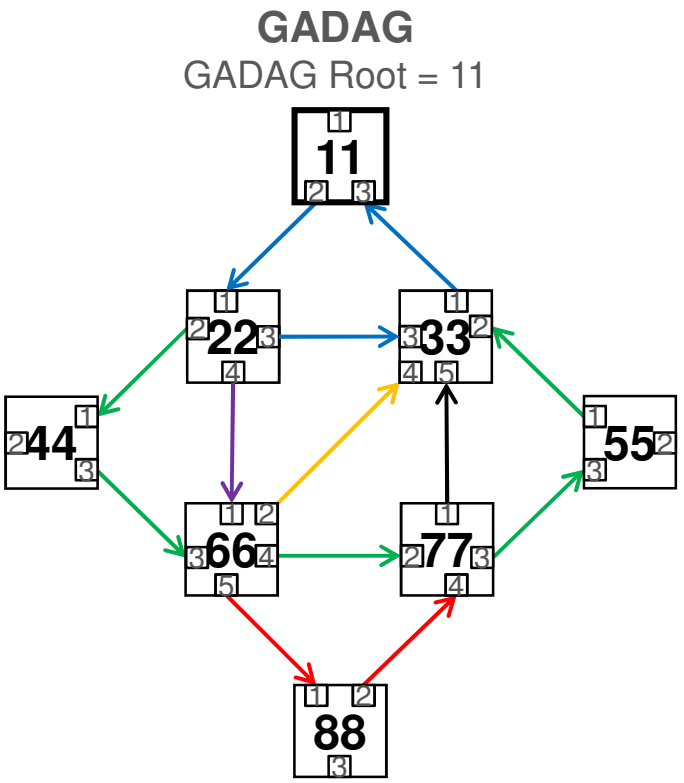


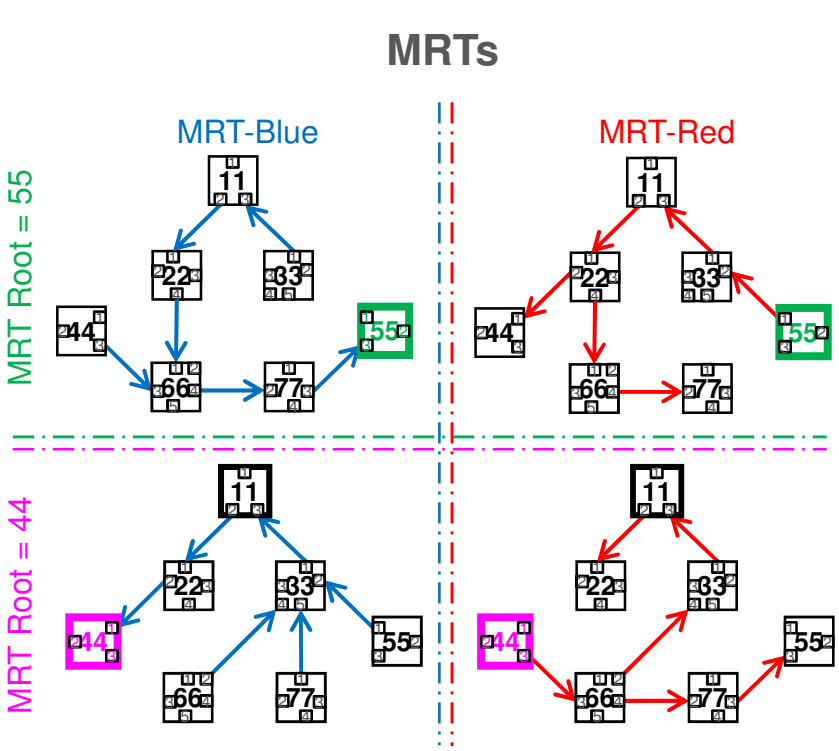
Maximally Redundant Trees with GADAG ECT Algorithm

- GADAG is computed centrally by GADAG Computer, e.g. PCE
  - → Centralized GADAG computation
- GADAG Computer specifies GADAG in Topology sub-TLV
  - Directed ear decomposition
  - MRT Roots are also specified
- MRTs are then computed by the BLCE of SPT Bridges based on the GADAG received from GADAG Computer
  - → 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 Strict Tree ECT Algorithm)

MRTG ECT Algorithm Example

Descriptor:
11; *TEP*
22
33
11 ; *Leaf, TEP*
22
44; *Root*; *TEP*
66
77
55; *Root*; *TEP*
33; *Leaf*
66, 4; *Circuit*
88; *Exclude*
77; *Leaf*
22
66; *Leaf*
66
33; *Leaf*
77
33; *Leaf*





- **First Hop** is the GADAG Root
- *Root* flag indicates MRT Root
- *Leaf* flag indicates end of ear

Getting the VIDs

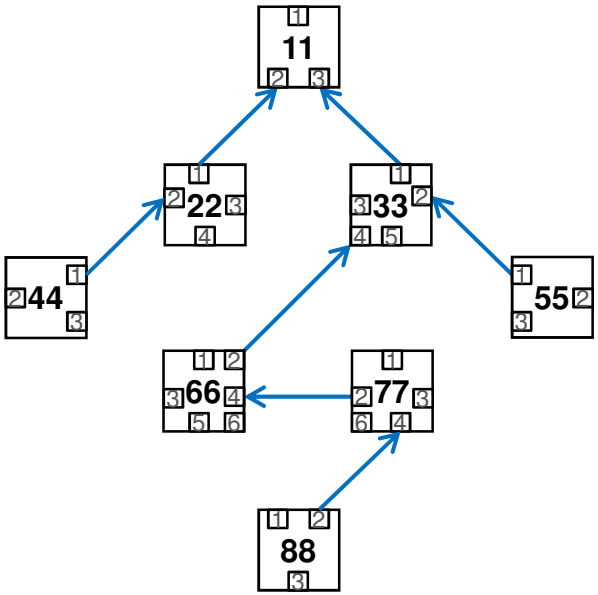
A VLAN's VID and VID Direction

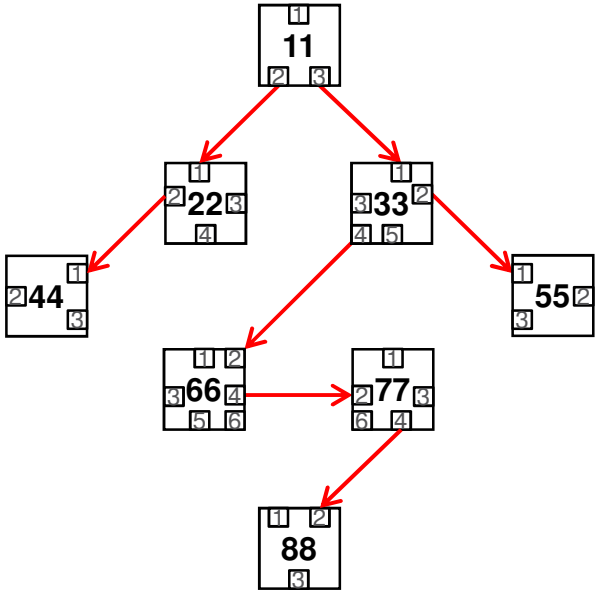
- A VLAN is associated with a particular explicit tree by the inclusion of the VLAN's Base VID in the Topology sub-TLV (preceding the Hop sub-TLVs)
- Further VIDs can be associated with the VLAN by the SPB Instance sub-TLV (28.12.5)
- Each VID is bidirectional by default
  - Each Traffic End Point bridge 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 VID's T/R flags in the Hop sub-TLV of the Traffic End Point bridge

Directed VIDs Example

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

Descriptor:		
11; *Root, TEP*	VID1: R	VID2: T
33		
66, 4; *Circuit*		
77		
88; *Leaf, TEP*	VID1: T	VID2: R
11; *Root, TEP*	VID1: R	VID2: T
22		
44; *Leaf, TEP*	VID1: T	VID2: R
33		
55; *Leaf, TEP*	VID1: T	VID2: R





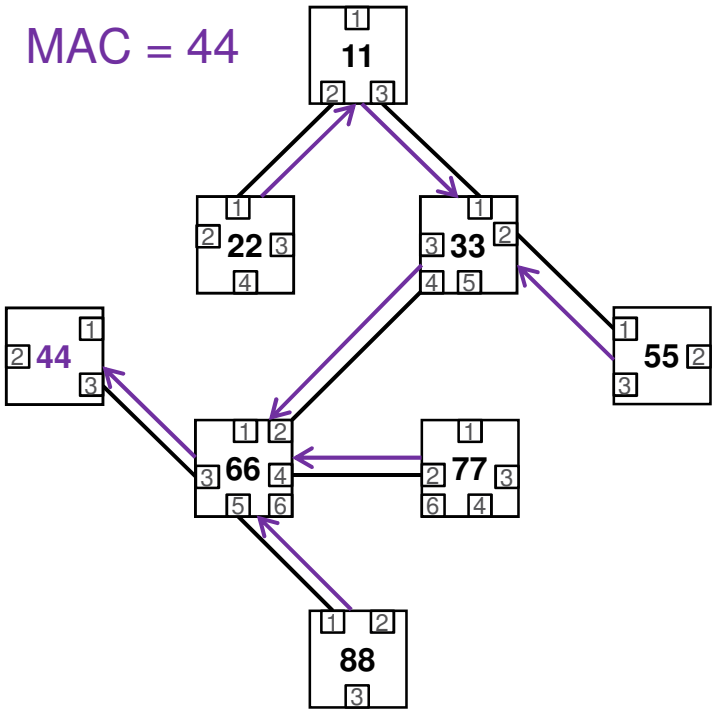
Getting the MACs

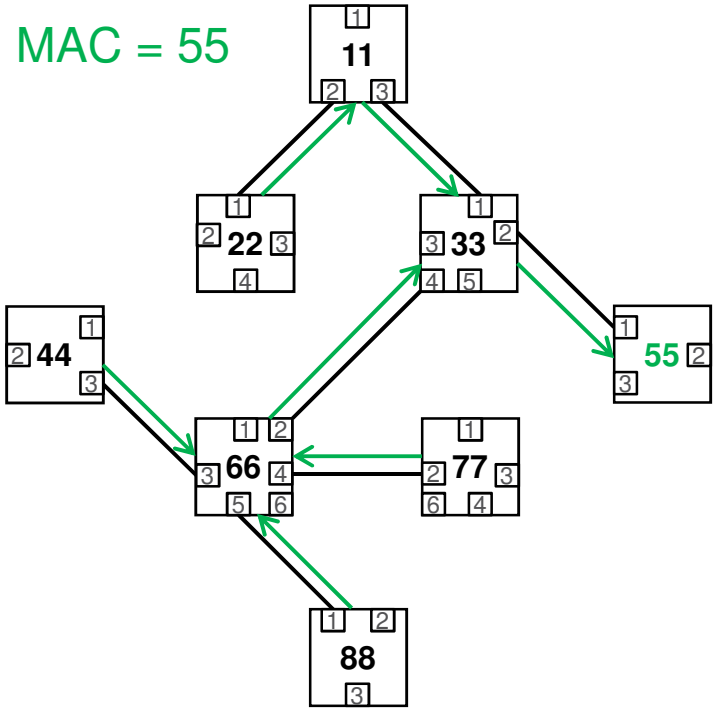
MAC Gives Direction

- 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 0.8: http://www.ieee802.org/1/files/private/ca-drafts/d0/802-1Qca-d0-8.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