Traffic Engineering for Multiple Spanning Tree Protocol in Large Data Centers

***Ho Trong Viet*, Yves Deville, Olivier Bonaventure, Pierre François**

ICTEAM, Université catholique de Louvain (UCL), Belgium.

Belgian Constraints Group @UCLouvain

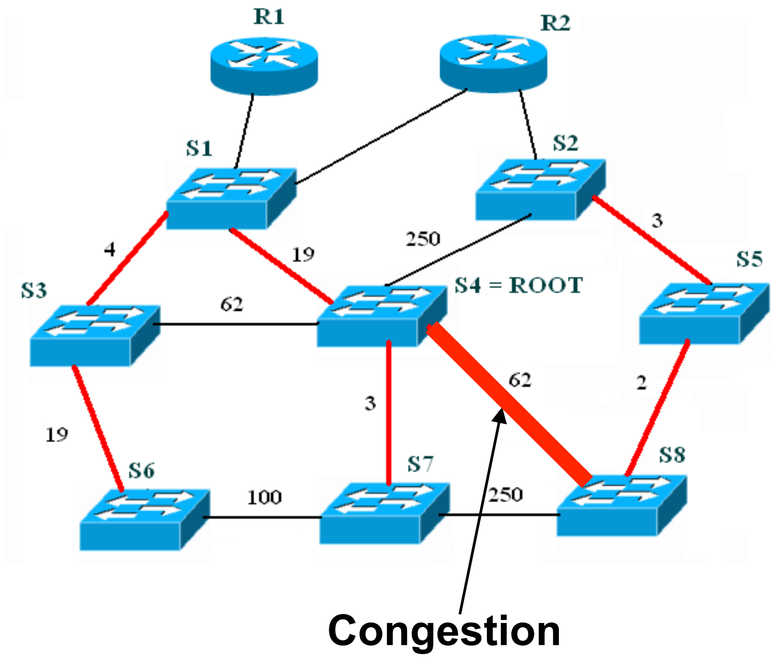
Introduction : Traffic Engineering

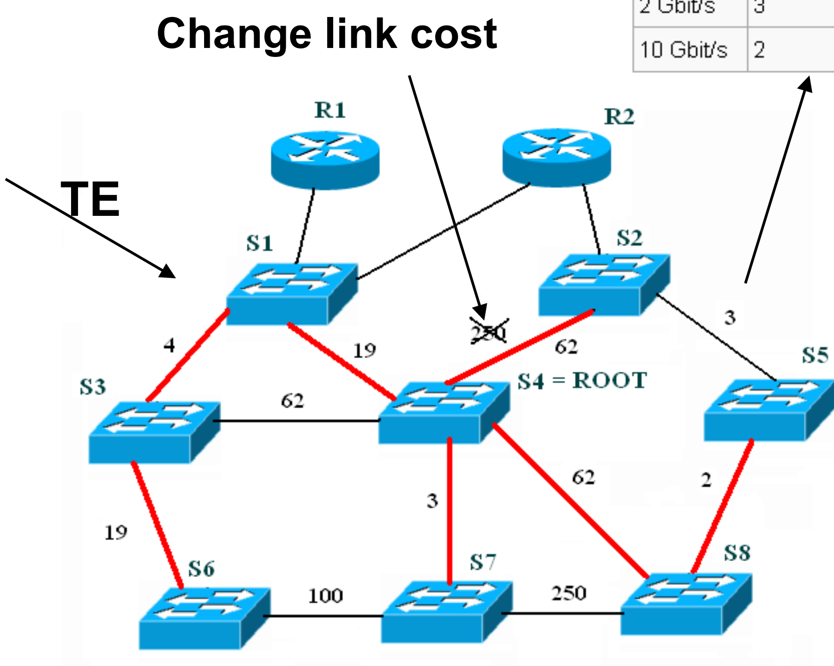
- Ethernet switches implement IEEE 802. 1d Spanning Tree Protocol (STP): reduces the network topology to a spanning tree
- Need to find an efficient use of available resources → sustain the increasing traffic demand without having overloaded links

→Traffic Engineering (TE) (NP-hard)

  - Configure link cost → avoid or lighten congestion

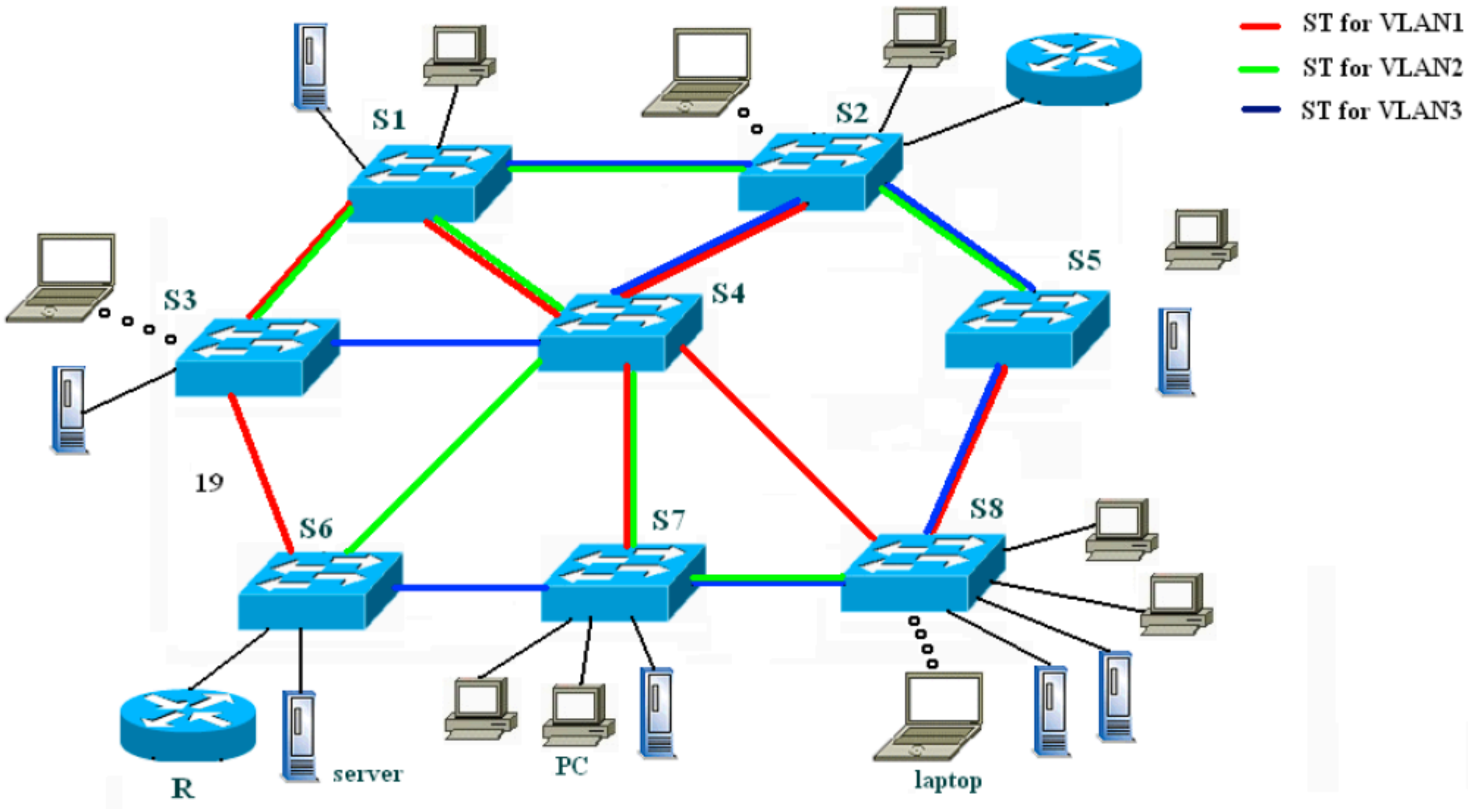
Data rate	STP Cost (802.1D-1998)
4 Mbit/s	250
10 Mbit/s	100
16 Mbit/s	62
100 Mbit/s	19
1 Gbit/s	4
2 Gbit/s	3
10 Gbit/s	2





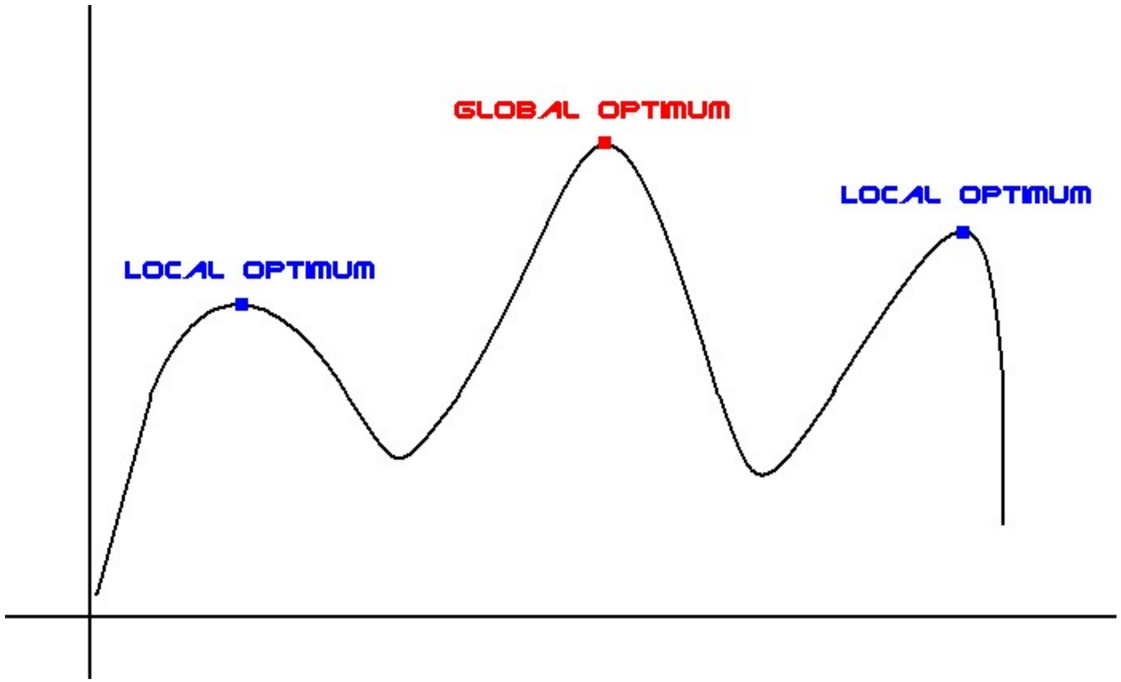
TE in Large Data Centers

- Problem: optimization of the choice of multiple spanning trees by 802.1s in Ethernet
  - Input: k VLANs, k traffic demand matrices
  - Output: k spanning trees minimize the maximal utilization (load/ bandwidth) U_{max}



Introduction : Local Search

- Local Search (LS) is a powerful method for solving combinatorial optimization problems such as traffic engineering
- LS has ability to find an intelligent path from a low quality solution to a high quality one in a huge search space
  - Iterate a heuristic of exploration to the neighborhood solutions
- COMET: Optimization Platform for LS



TE in Large Data Centers
Load Balancing Case

State of the art

- **IEEE 802.1s Multiple Spanning Tree Protocol**
- [Xiaoming He et al.] Traffic Engineering for Metro Ethernet Based on Multiple Spanning Trees: network ≤ 25 nodes
- [Wentao Chen et al.] Design of Multiple Spanning Trees for Traffic Engineering in Metro Ethernet: US Network 12 nodes
- [Aref Meddeb] Multiple Spanning Tree Generation and Mapping Algorithms for Carrier Class Ethernets: 7-node network
- [M. Padmaraj et al.], Metro Ethernet Traffic Engineering Based on Optimal Multiple Spanning Trees: 30-node network
- *[Ho et al.] Using Local Search for Traffic Engineering in Switched Ethernet Networks: 1 VLAN (802.1d), solution for Portland and Fat Tree with* ***320 nodes***

Result

- ***Optimizing spanning trees instead of link weights*** → search space size ↓
- ***Incremental link loads computation*** → avoid the all pairs paths computation
- ***Local search approach*** extended from *[Ho et al.]* using a Constraint-Based environment (Comet)
- Good solution for Data Centers with **10K servers, 564 switches, 16 VLANs**

Search Space

- ***The search space is made of spanning trees, not link costs***
- Link costs for each spanning tree are configured after optimization phase

  → Allowed much broader exploration

- For each spanning tree, ***any switch can be selected as the Root*** (do not affect the link load computation)

  → Reduced the search space

Plan

- Introduction
- Problem description
- Local Search Algorithm for Multiple Spanning Tree Protocol problem – LSA4MSTP
- Experimental Results
- Conclusion

Problem description

Input

- Network topology: G=(N,E)
  - N set of switches (nodes), E set of links (arcs)
- k VLANs, k initial link cost matrices $W_1, W_2, \ldots, W_k$
- Bandwidth matrix BW
- k traffic demand matrices $TD_1, TD_2, \ldots, TD_k$

Objective

- Find k spanning trees for k VLANs minimizing the maximal link utilization U_{max}

$$U_{max} = \max \{\text{Utilization } U_e \mid \text{for all link } e \text{ in } E\}$$

- Deduce k associate configurations of link cost $W^*_1, W^*_2, \ldots, W^*_k$ (straightforward)

Plan

- Introduction
- Problem description
- Local Search Algorithm for Multiple Spanning Tree Protocol problem – LSA4MSTP
- Experimental Results
- Conclusion

LSA4MSTP: Root Selection & Initial Solution

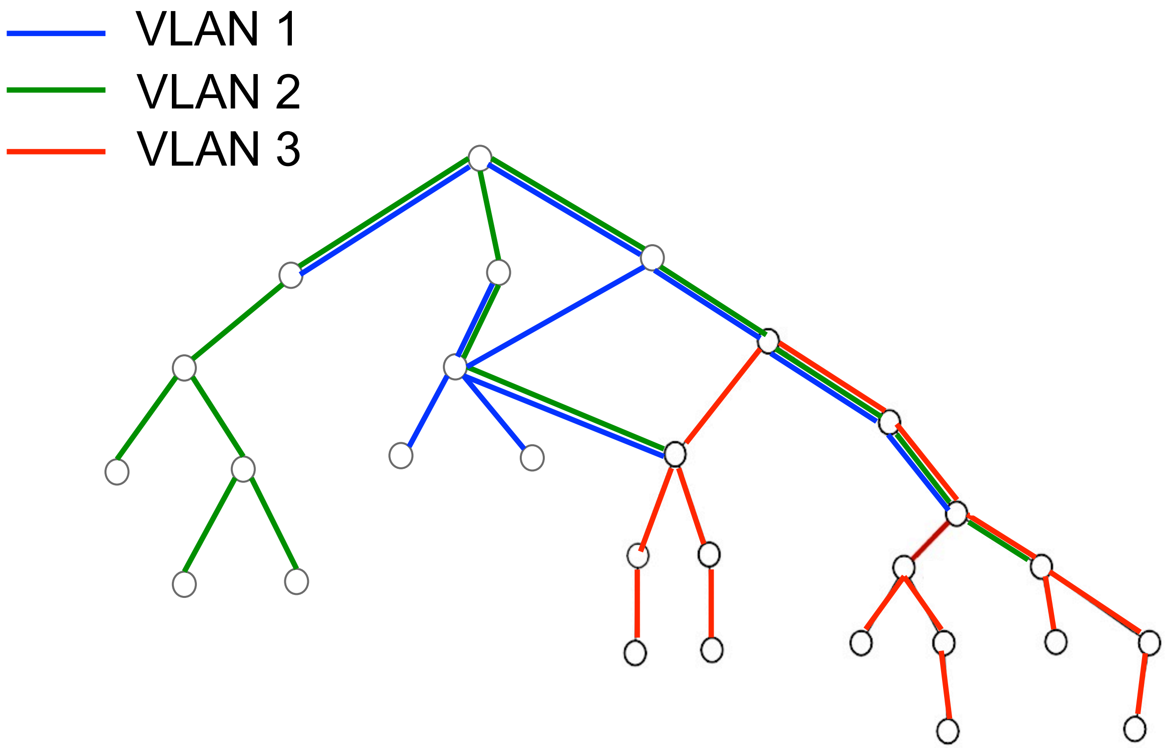
- Root Selection
  - In each spanning tree (VLAN), any node can be chosen as root, without impact on results
  - A root can be chosen arbitrarily
  - Our choice: configure the switch with highest capacity (ports x bandwidth) as the root

- k Initial Spanning Trees
  - Shortest path tree based on initial link cost matrices W_1, W_2, …, W_k (simulate 802.1d standard)

LSA4MSTP: Heuristics to find a good neighbor

Select VLAN to do the edge replacement

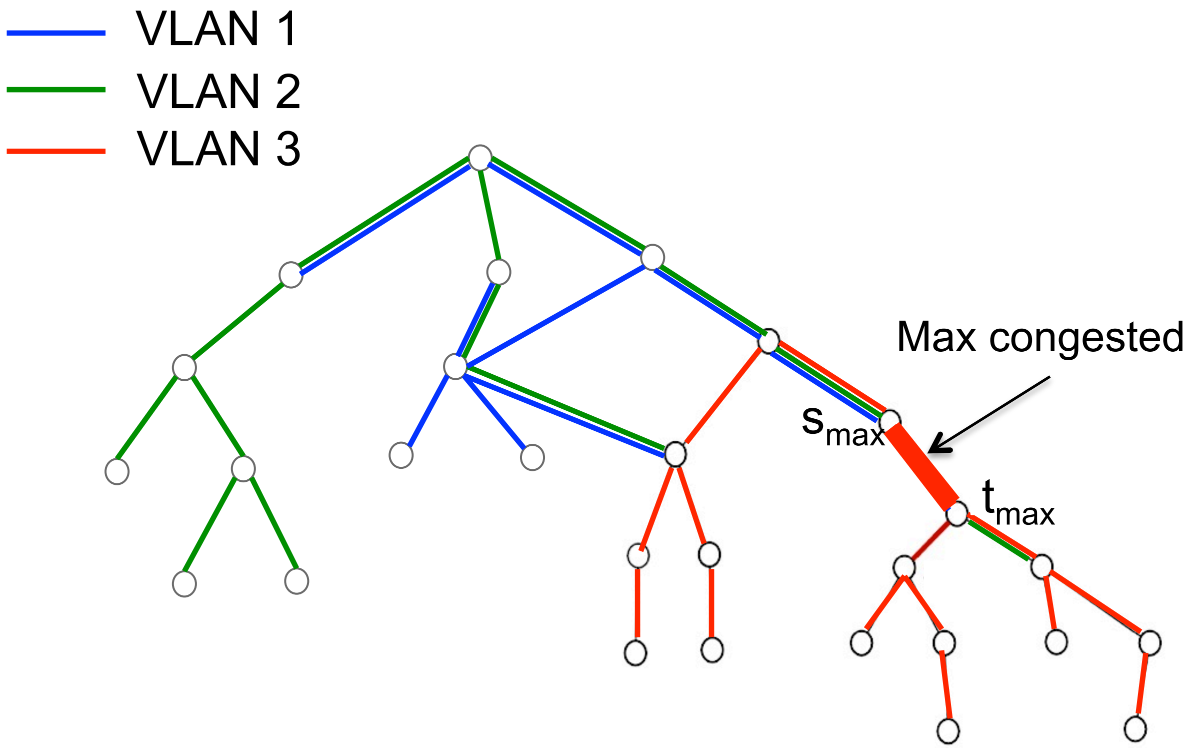
- Find the most congested oriented link (s_{max},t_{max}): $U_{max}=U[s_{max},t_{max}]$
- Compute the load rate of each VLAN on (s_{max},t_{max})
- VLAN is selected based on its rate on (s_{max},t_{max})



LSA4MSTP: Heuristics to find a good neighbor

Select VLAN to do the edge replacement

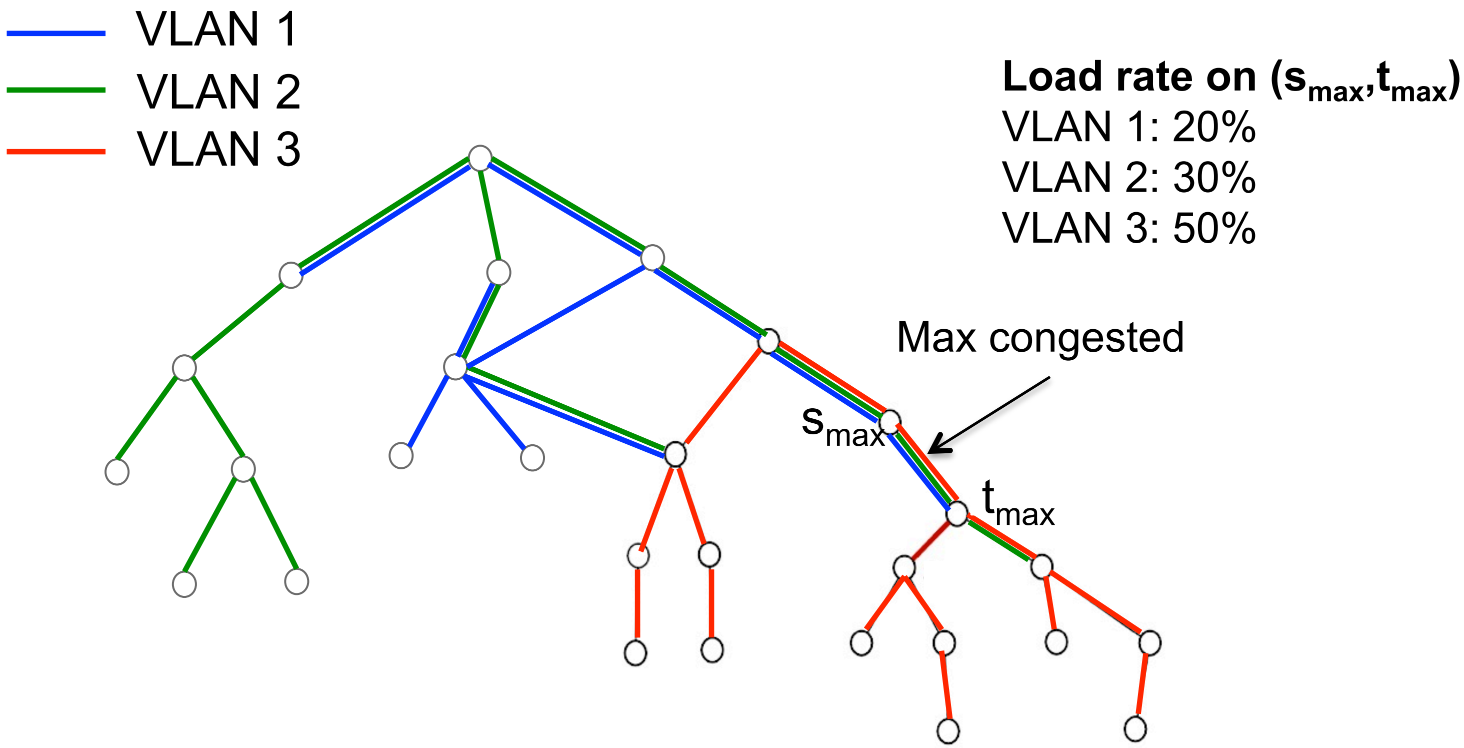
- Find the most congested oriented link (s_{max},t_{max}): $U_{max}=U[s_{max},t_{max}]$
- Compute the load rate of each VLAN on (s_{max},t_{max})
- VLAN is selected based on its rate on (s_{max},t_{max})



LSA4MSTP: Heuristics to find a good neighbor

Select VLAN to do the edge replacement

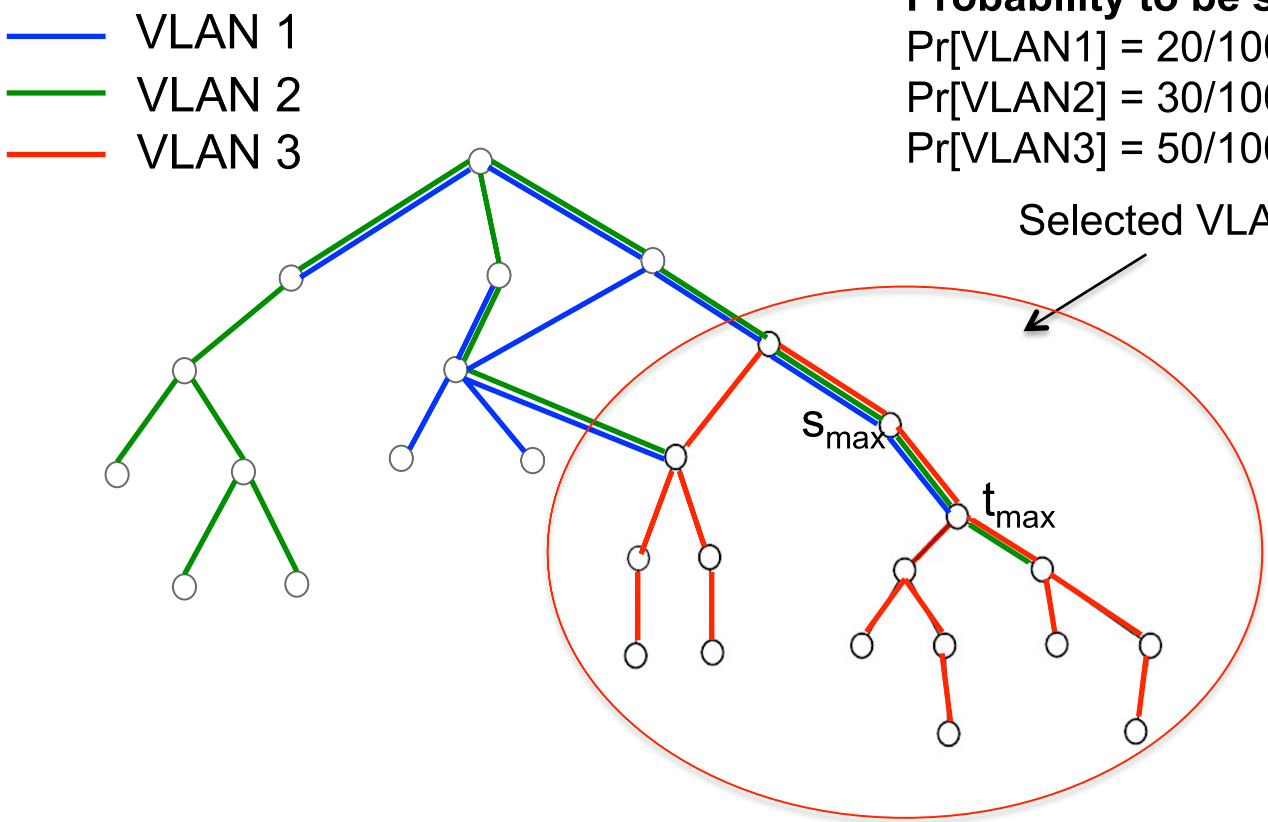
- Find the most congested oriented link (s_{max},t_{max}): $U_{max}=U[s_{max},t_{max}]$
- Compute the load rate of each VLAN on (s_{max},t_{max})
- VLAN is selected based on its rate on (s_{max},t_{max})



LSA4MSTP: Heuristics to find a good neighbor

Select VLAN to do the edge replacement

- Find the most congested oriented link (s_{max},t_{max}): $U_{max}=U[s_{max},t_{max}]$
- Compute the load rate of each VLAN on (s_{max},t_{max})
- VLAN is selected based on its rate on (s_{max},t_{max})

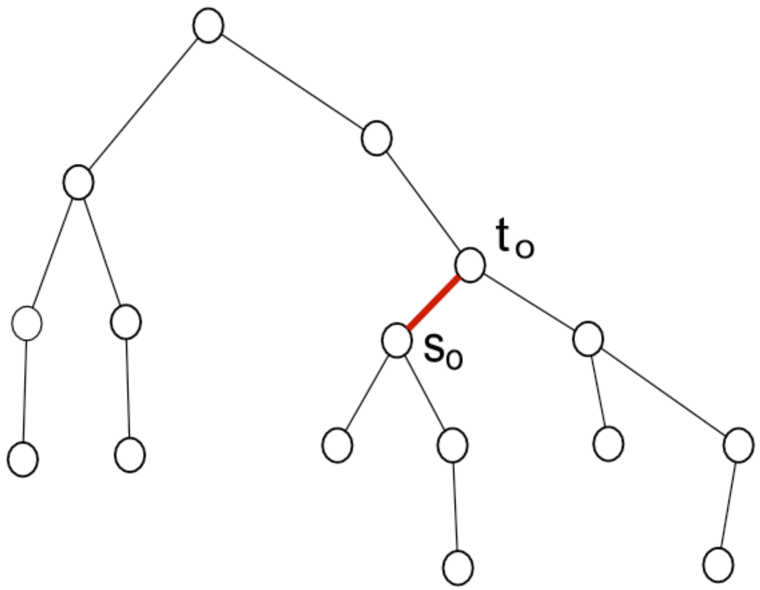


Probability to be selected
Pr[VLAN1] = 20/100 = 0.2
Pr[VLAN2] = 30/100 = 0.3
Pr[VLAN3] = 50/100 = 0.5

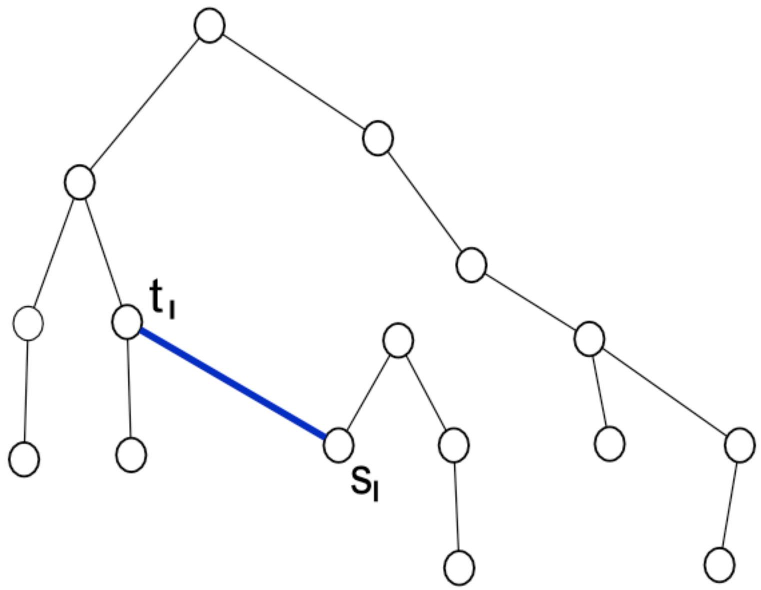
LSA4MSTP: Heuristics to find a good neighbor

- From the selected VLAN
  - Choosing an edge to remove
  - Adding a new edge

Remove an edge

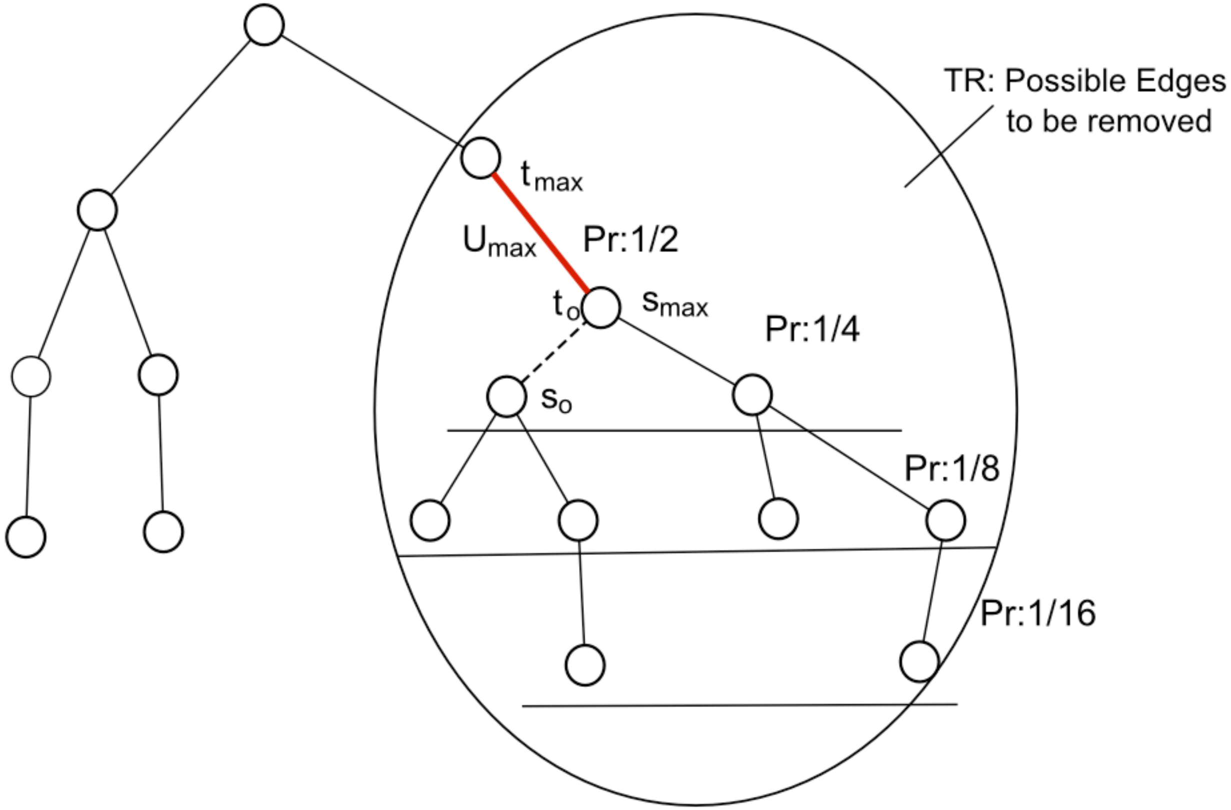


Add an another edge



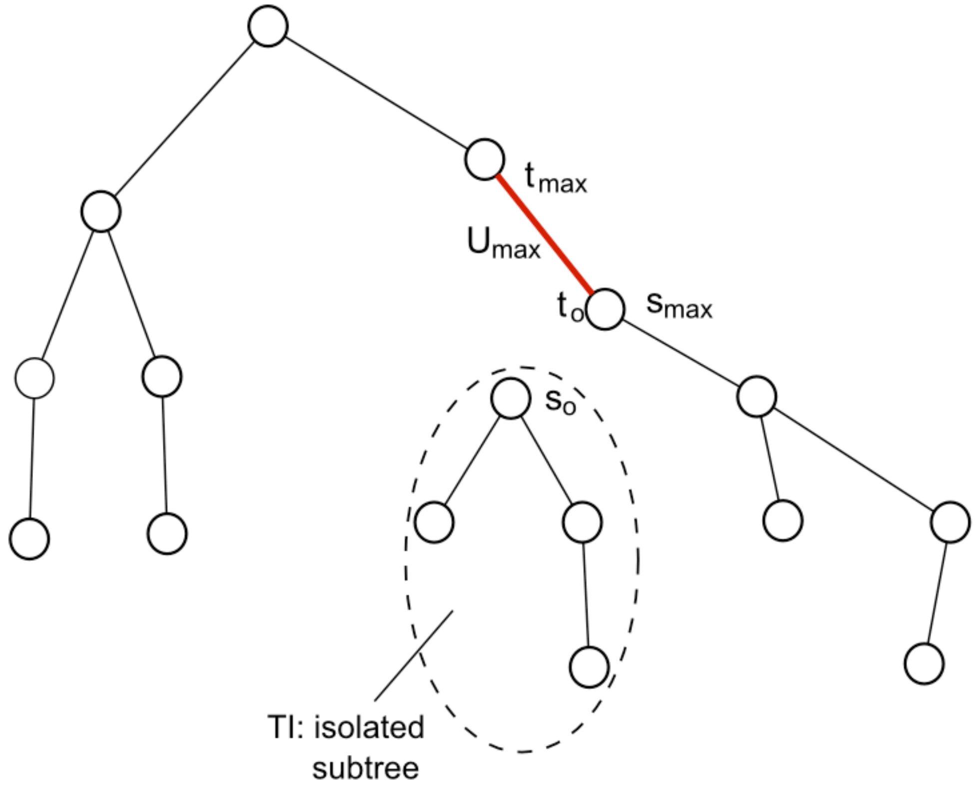
LSA4MSTP : Heuristics to find a good neighbor

- Removing an edge
  - Most congested oriented link (s_{max},t_{max}): U_{max}=U[s_{max},t_{max}]
  - SR: set of candidate edges to be removed (all edges in TR)
  - Assign to each edge in SR a probability to be selected
  - Select (s_O, t_O) to be removed



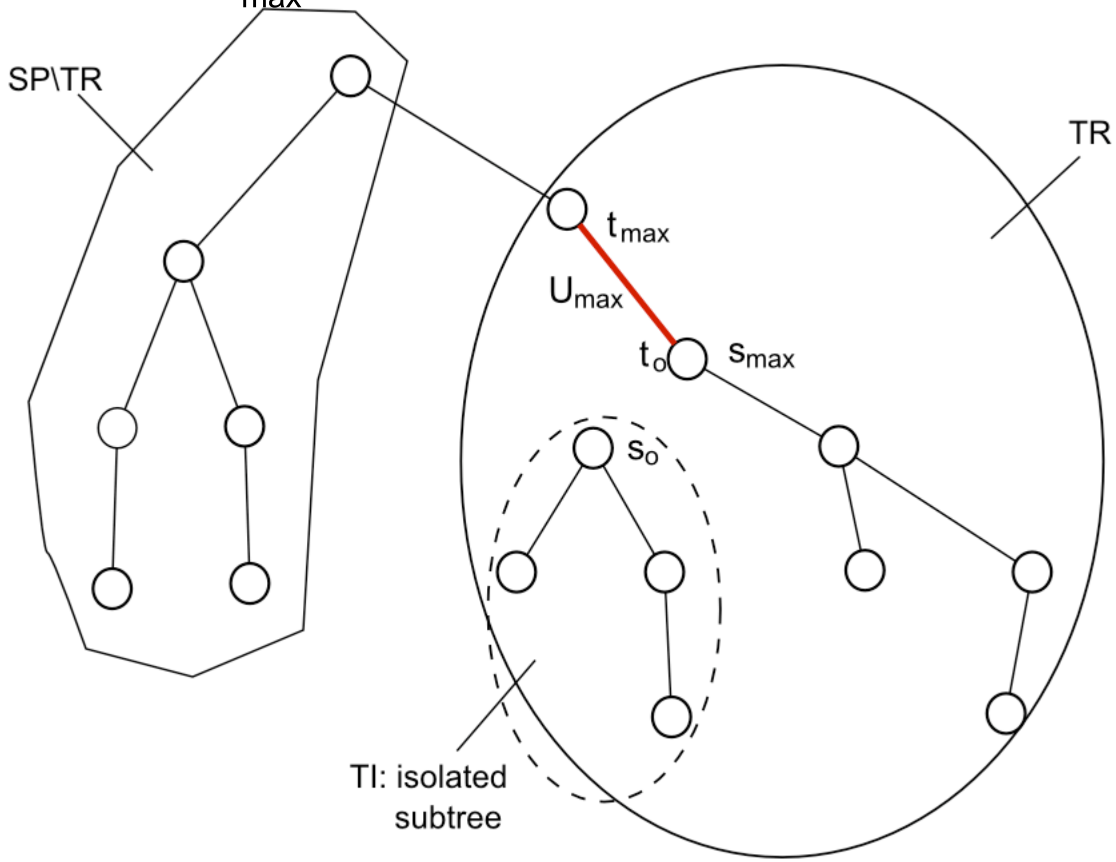
LSA4MSTP : Heuristics to find a good neighbor

- Adding an edge
  - TI: isolated subtree - unconnected to the root.
  - SA contains all the edges that join SP\TR and TI.
  - Select (s_I, t_I) to be added from k highest rest bandwidth edges in SA which leads to min U_{max}



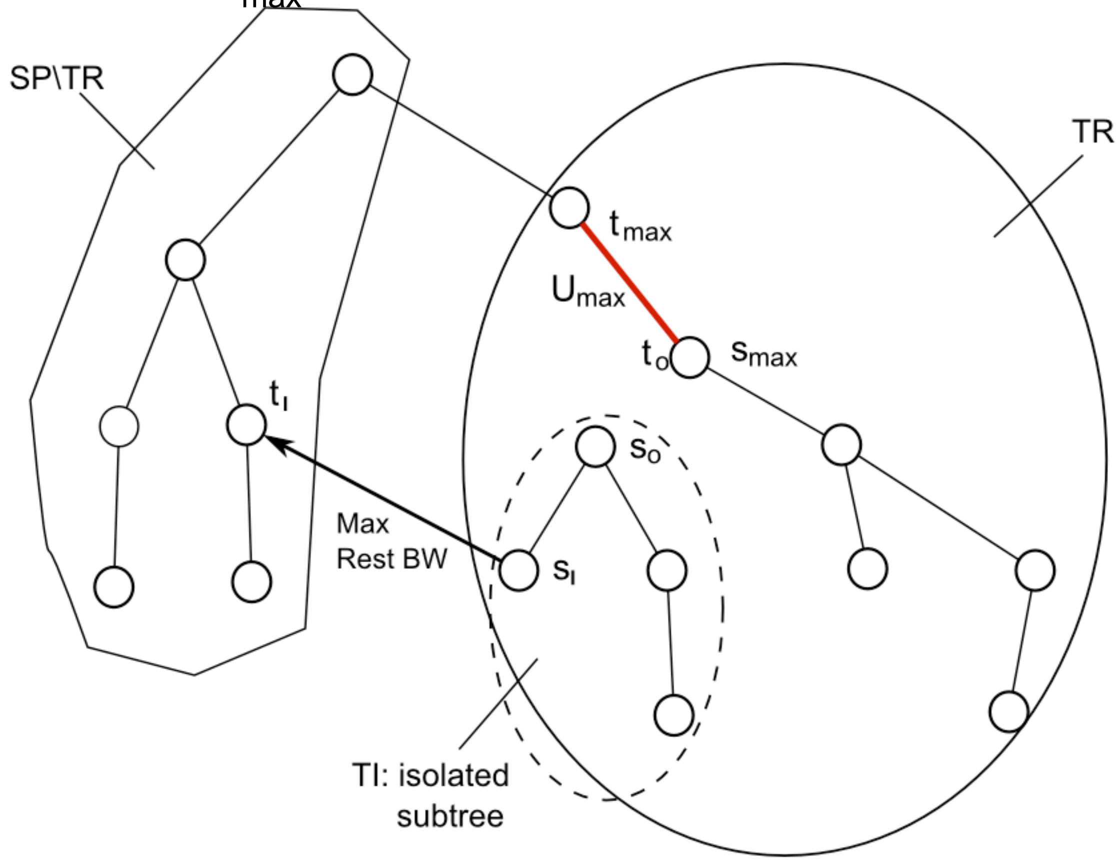
LSA4MSTP : Heuristics to find a good neighbor

- Adding an edge
  - TI: isolated subtree - unconnected to the root.
  - SA contains all the edges that join SP\TR and TI.
  - Select (s_l, t_l) to be added from k highest rest bandwidth edges in SA which leads to min U_{max}



LSA4MSTP : Heuristics to find a good neighbor

- Adding an edge
  - TI: isolated subtree - unconnected to the root.
  - SA contains all the edges that join SP\TR and TI.
  - Select (s_I, t_I) to be added from k highest rest bandwidth edges in SA which leads to min U_{max}



LSA4MSTP: Metaheuristics

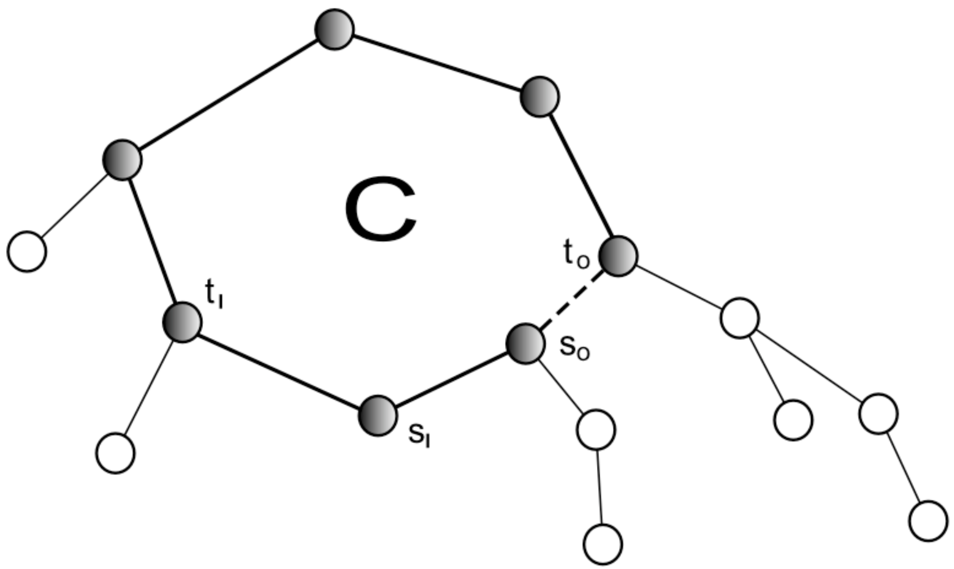
- Termination criteria
  - Time window: 15 minutes

- Tabu List
  - Tabu: forbids the repetitive replacement of a couple of edges in successive iterations.
  - Tabu list: inserts only the added edge at each search iteration – freeze for next x iterations

LSA4MSTP: Technical issues

- Incremental link load computation (20% speed up)
  - Load changes only on the links on the cycle C created by removing (s_O, t_O) and adding (s_I, t_I)
  - ➔ Avoid recomputing all pair paths

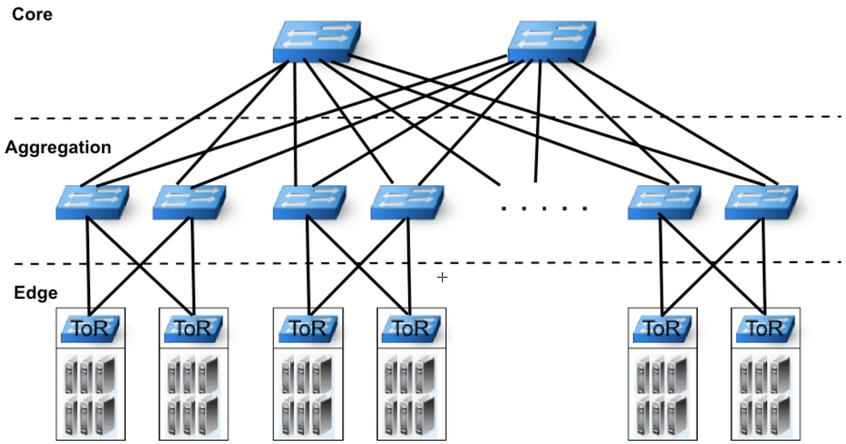
- Use of LS (Graph & Tree) framework
  - Graph and trees are objects available in the LS algorithm



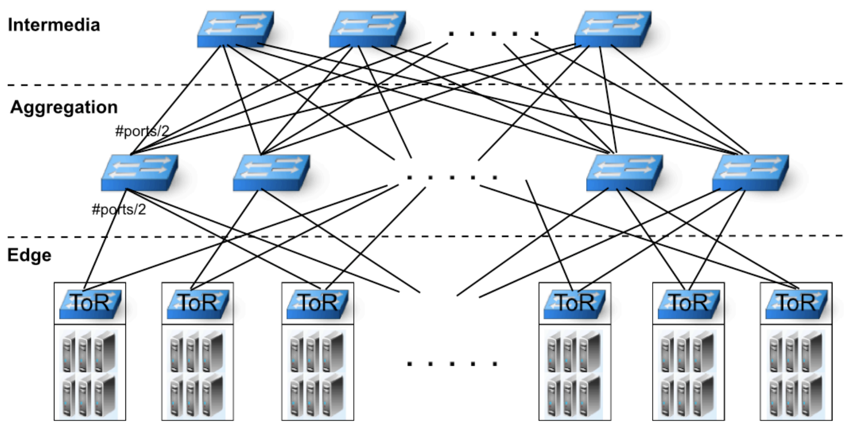
Plan

- Introduction
- Problem description
- Local Search Algorithm for Multiple Spanning Tree Protocol problem – LSA4MSTP
- Experimental Results
- Conclusion

Experimental Results



Private Enterprise

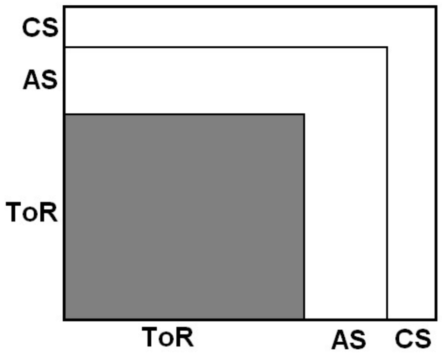


Cloud

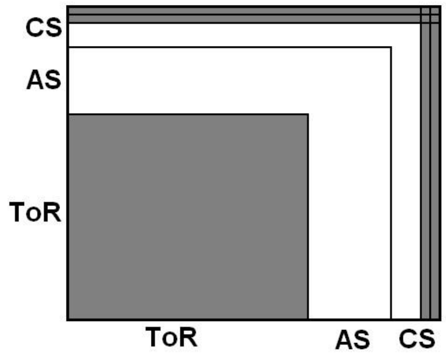
We consider 2 topology types

- Private Enterprise DC (3-Tier Cisco): few hundred → few thousand servers
  - In our tests, 4K servers, 20 servers/rack, 200 ToRs, 40 Aggregation SWs, 2 Core SWs
- Cloud DC (VL2): few thoudsand → more than 10K servers
  - Improvement of 3-Tier Cisco: Core → Intermedia,
  - In our tests, 10K servers, 20 servers/rack, 500 ToRs, 32 Aggregation SWs, 32 intermedia SWs

Experimental Results



Internal TM



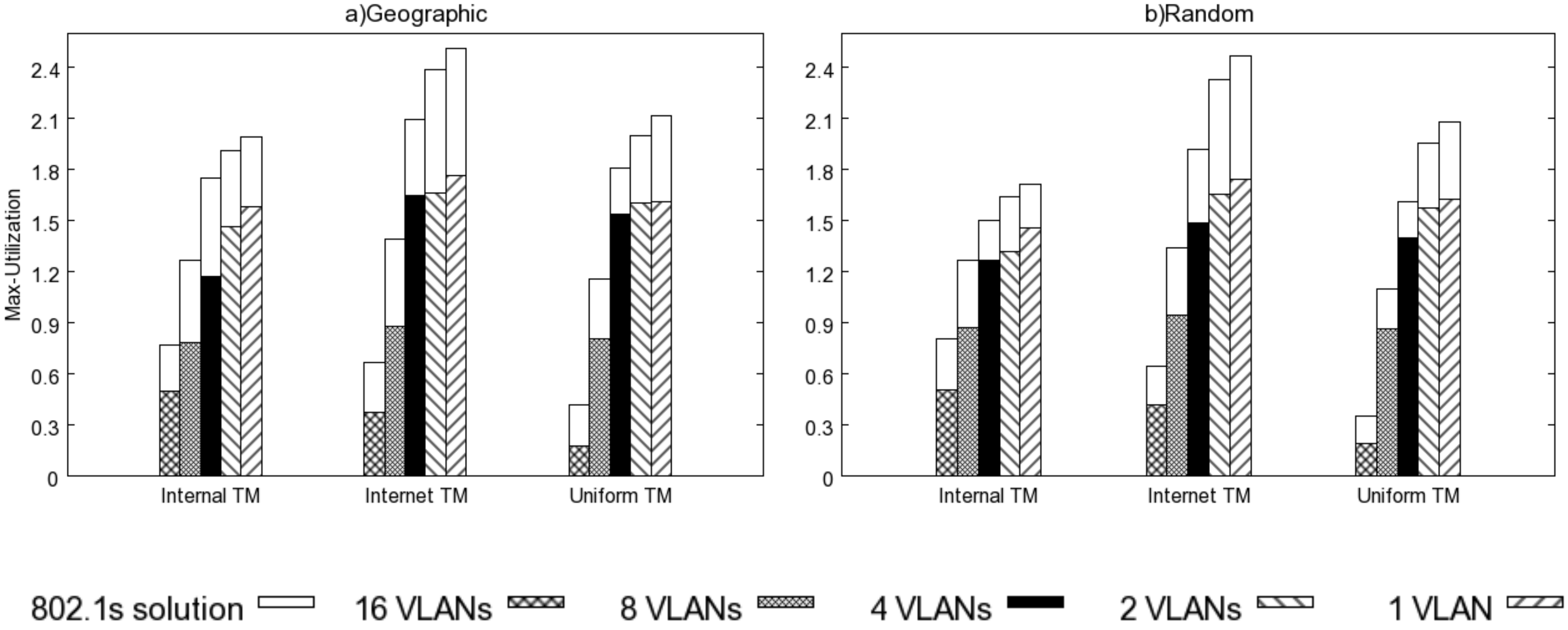
Internet TM

Uniform TM

- 16 VLANs for each topology, 2 approaches to generate VLAN
  - Geographic: each VLAN groups a set of neighboring racks
  - Random
- Merge VLANs 2 by 2 → for each topology: 16, 8, 4, 2, 1 VLANs
- Analyse of a private enterprise SNMP data with 53 nodes, we consider 3 traffic demand matrix types
  - Internal TM: all traffic stays within VLAN – discussions across racks
  - Internet TM: traffic within VLAN 80%, traffic across VLANs 20%
  - Uniform TM: uniform distribution between all pairs of SWs inside VLAN

Experimental Results

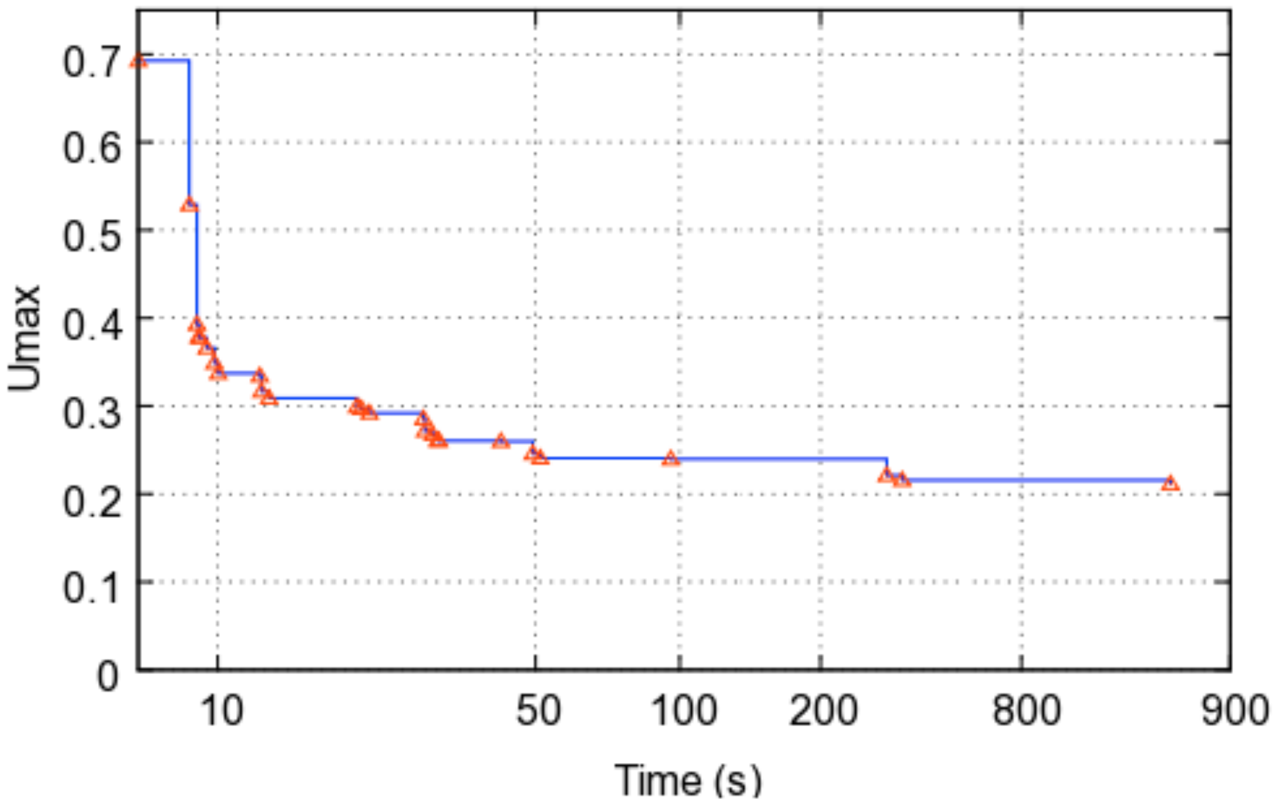
Cloud data centers



- U_{max}[LSA4MSTP]:
  - ~ 50% of U_{max}[802.1s] for 16 VLANs
  - ~ 60% of U_{max}[802.1s] for 8 VLANs
  - ~70 – 80% U_{max}[802.1s] for 4, 2, 1 VLAN

Experimental Results

Improvement of U_{max} over execution time



- Time window 15'
- Very good results are obtained quickly
  - U_{max} reduces to ~50% in the first 10s

Experimental Results

#used links across tiers for Cloud

Links	Available	1 VLAN		2 VLANs		4 VLANs		8 VLANs		16 VLANs	
		802.1s	LSA	802.1s	LSA	802.1s	LSA	802.1s	LSA	802.1s	LSA
Int-AS	1024	63	63	66	69	73	78	75	81	91	98
AS-ToR	1000	500	500	521	532	536	573	598	648	656	712
Total	2024	563	563	587	601	609	651	673	729	747	810

- Links Int-As always < 100 links
- #used links AS-ToR grows quicky with #VLANs

- LSA4MSTP uses more links than 802.1s
- With 63 more links for 16 VLANs, LSA4MSTP reduces 50% U_{max}

Experimental Results

#used links across tiers for Cloud

Links	Available	1 VLAN 802.1s	1 VLAN LSA	2 VLANs 802.1s	2 VLANs LSA	4 VLANs 802.1s	4 VLANs LSA	8 VLANs 802.1s	8 VLANs LSA	16 VLANs 802.1s	16 VLANs LSA
Int-AS	1024	63	63	66	69	73	78	75	81	91	98
AS-ToR	1000	500	500	521	532	536	573	598	648	656	712
Total	2024	563	563	587	601	609	651	673	729	747	810

- Links Int-As always < 100 links
- #used links AS-ToR grows quicky with #VLANs

- LSA4MSTP uses more links than 802.1s
- With 63 more links for 16 VLANs, LSA4MSTP reduces 50% U_{max}

Plan

- Introduction
- Problem description
- Local Search Algorithm for Multiple Spanning Tree Protocol problem – LSA4MSTP
- Experimental Results
- Conclusion

Conclusion

- New TE technique based on local search
  - Extended from LSA4STP for single switched Ethernet network → adapting heuristics for Large Data Centers deploying Multiple Spanning Tree Protocol 802.1s
- Consider current modern topologies (up to 10K servers) with studied traffic demand matrices in our experiments
- Give good performance with large instances of network topology
  - LSA4STP: Grid, Cube, Expanded Tree, Fat Tree, PortLand
  - LSA4MSTP: 3-Tier Cisco, Cloud Data center architecture
- Local search heuristic has been implemented in the Comet language and the simulations show promising results.
- Further work: extend framework to take in to account delay, sum load and fault tolerant aspect

Thanks for your attention!

- Question?