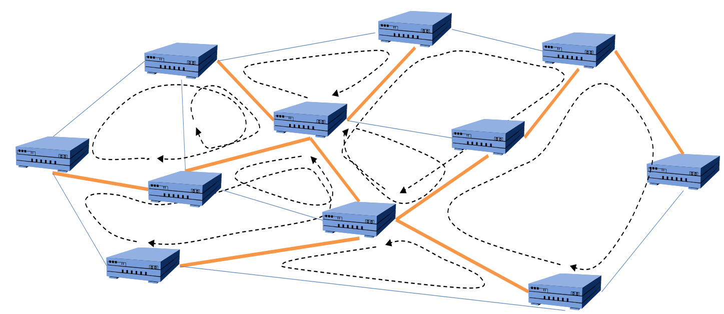
Tie-set Based Fault Tolerance (TBFT)

Kiyoshi Nakayama, Kyle Benson,
Yi-Hsi Huang, Vahe Avagyan, Lubomir Bic,
Michael Dillencourt

Comp. Science, University of California, Irvine, USA

Research Target

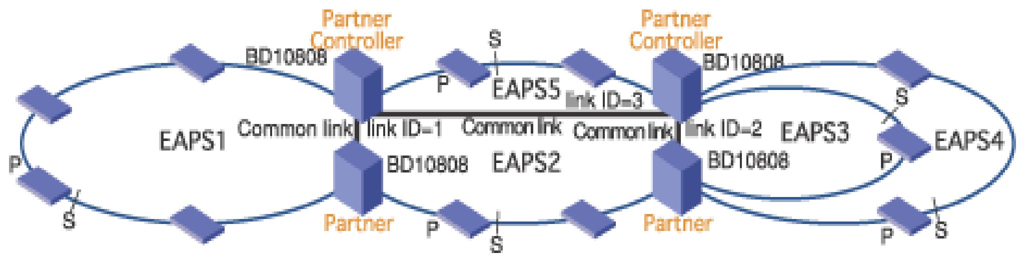
- Background
  - Networks are becoming Large-Scale, Complicated (Mesh Topology, Intricately-Intertwined)
- Loop (Ring) Network Management
  - Ring Protection of Failure Recovery
  - Loop Management in Power Grid



Aiming at Versatile Autonomous Distributed Local Control for Various Network Problems based on Loop Structure

Existing Protocols Focusing on Rings

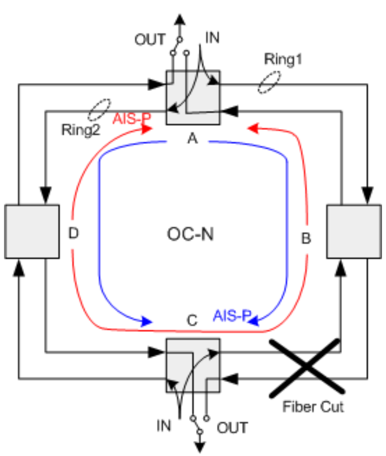
EAPS (Ethernet Automatic Protection Switching)

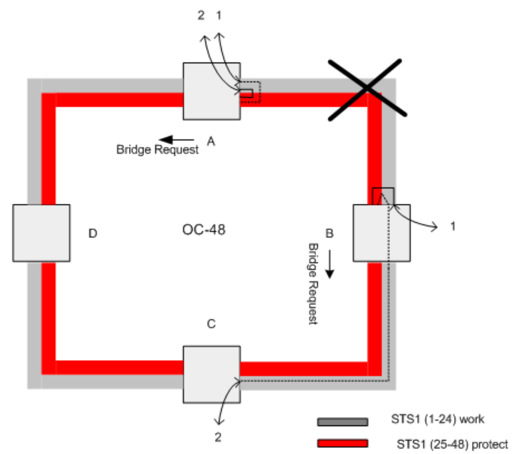


- Features
  - Easy to conduct stable distributed control
  - Performs better than a star topology under heavy network load
  - Does not require network server to manage the connectivity between the computers
  - Able to create much larger network using Token Ring
  - Fast in fail-over

SONET Rings

UPSR
(Unidirectional Path Switched Ring)





BLSR
(Bidirectional Line Switched Ring)

These protocols are NOT applicable to mesh networks

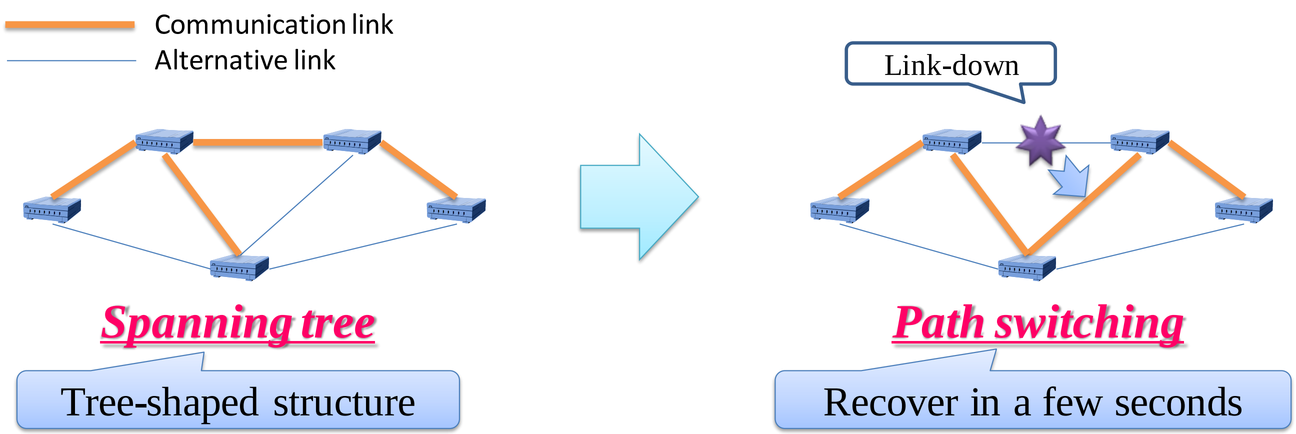
Existing Protocols Applicable to Mesh Networks

STP (Spanning Tree Protocol)

- Uses Spanning Tree in setting communication paths
- Even if a link failure occurs, a network is kept connected by shifting to an alternative link.
- Takes 50 seconds to complete failure recovery

RSTP (Rapid-Spanning Tree Protocol)

- Enhanced version of STP
- Takes a few seconds to complete failure recovery



Route Switching based on Tie-Sets

Taking advantage of ring-based restoration

- Succeeded fast and reliable recovery of ring-based switching
- Local and distributed control within a loop (tie-set)

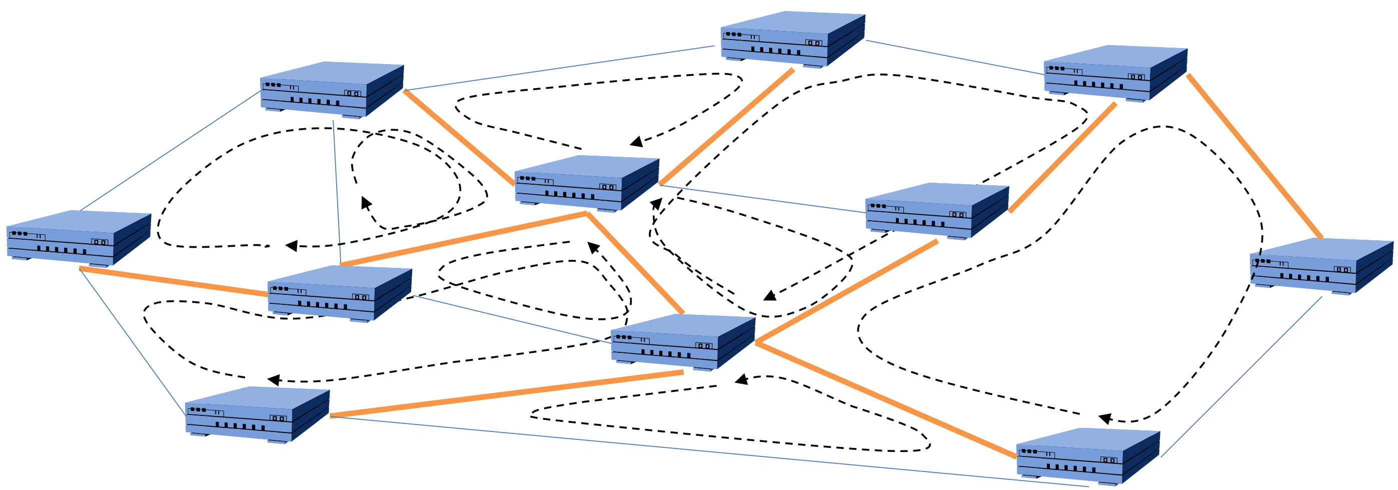
Applicable to mesh topological networks

- Logically created ring structure with a concept of a fundamental system of tie-sets
- More local control and lower-load communication than RSTP

(With RSTP, some switching and many messages affect an entire network)

Our Research Target

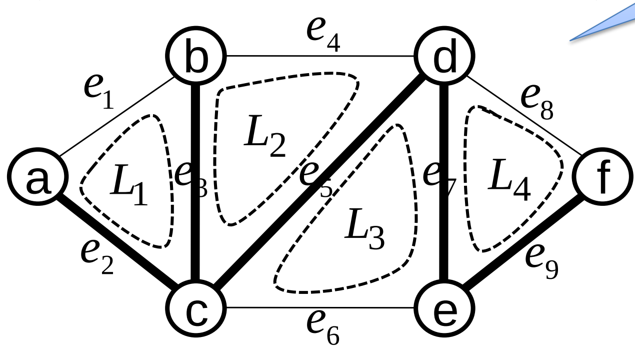
Realization of Distributed Control for Configuration, Failure, and Performance Management based on Loop Structure in Information Networks



Cover all the nodes and links with independent loops

Tie-Set Graph Theory

Original Graph $G=(V, E)$
(Bi-connected and undirected)



$\{L_1, L_2, L_3, L_4\}$ *is a fundamental system of tie-sets*

Tree T
Cotree $\overline{T} = E - T$

Rank : $\rho = \rho(G) = |T|$
Nullity : $\mu = \mu(G) = |\overline{T}|$

- **Tie-set** $L_i = \{e_1^i, e_2^i, ..., e_k^i\}$ **:** A set of edges which constitutes a loop in G.
- **Fundamental Tie-set :** A tie-set which contains just one edge of cotree $\overline{T}$.

  Ex: $L_1=\{e_1,e_2,e_3\}$ *is a fundamental tie-set, where* e_1 *is the edge of cotree.*
- **Fundamental System of Tie-sets :** All fundamental tie-sets in G. In other words, a string of tie-sets created by a tree T.
  - μ fundamental and independent tie-sets exist in *G*
  - All the vertices and edges are covered

Distributed Control based on Tie-Sets

What is the Tie-Set?

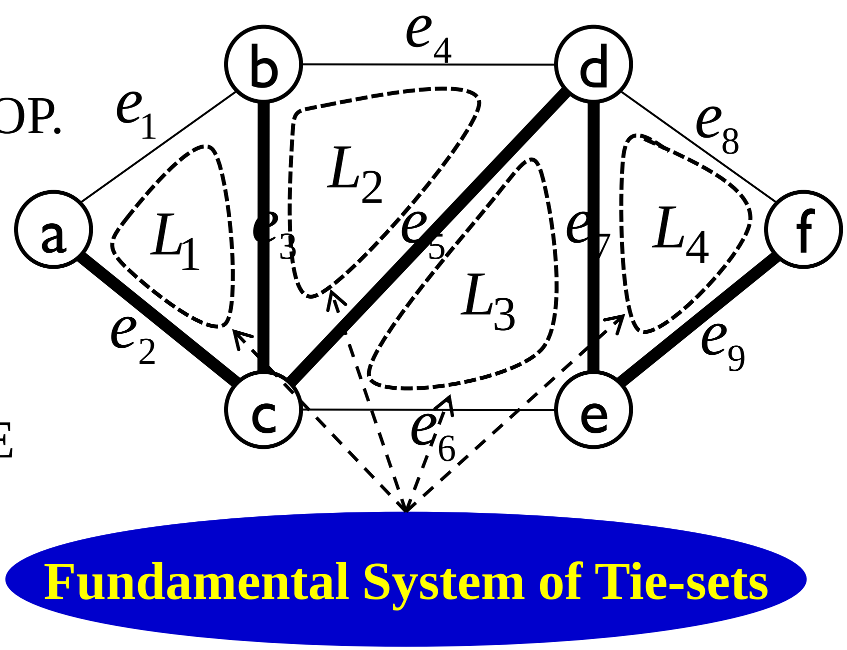
- A Set of Links (Edges) that constitutes a LOOP. (Cycle or Ring).

Ex: Tie-set $L_1=\{e_1, e_2, e_3\}$

Fundamental System of Tie-sets

- Tie-Sets (Loop Structure) Defined by a TREE

Ex: $\{L_1, L_2, L_3, L_4\}$



Advantages

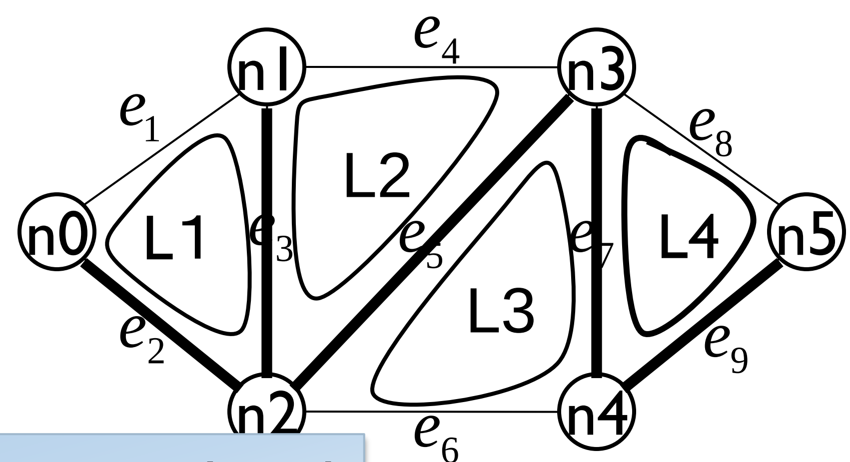
Cover an ENTIRE Network with LOOPS !

- *Communications become easy as Messages are exchanged within a Tie-Set*
- *Independent Solution by Autonomous Distributed Control in a Tie-Set even in a Mesh Network*

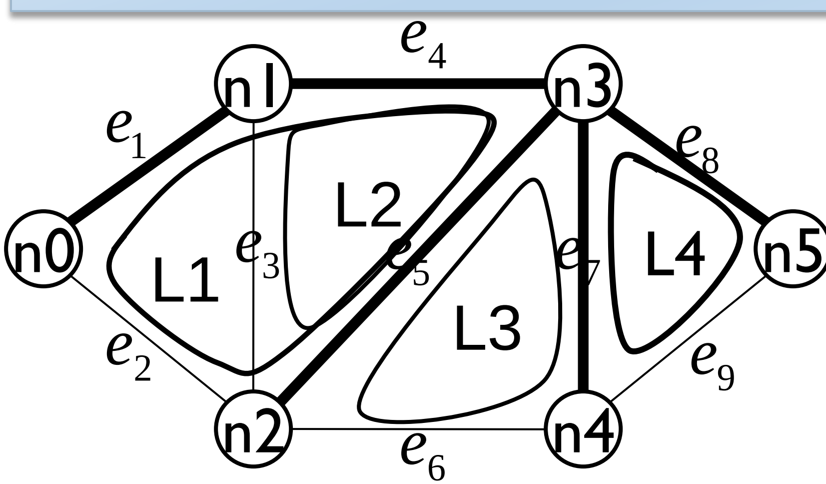
Various Tie-Sets by Trees

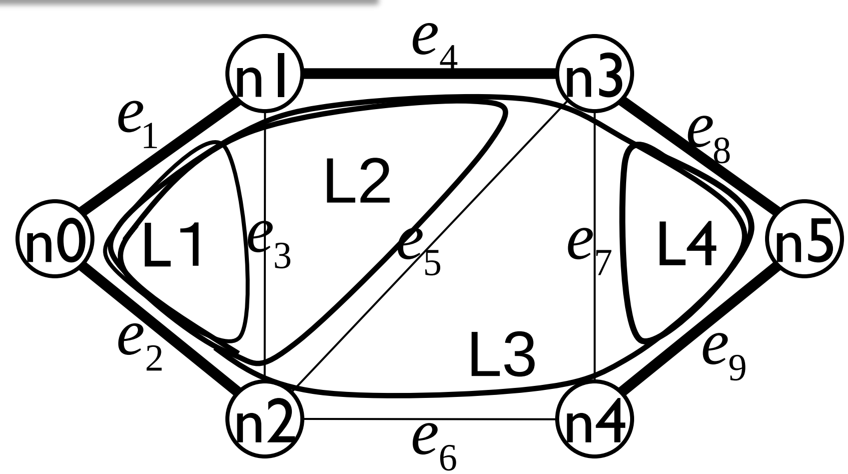
Segmentation using tie-set theory

Tie-set distribution is different by formulation of tree



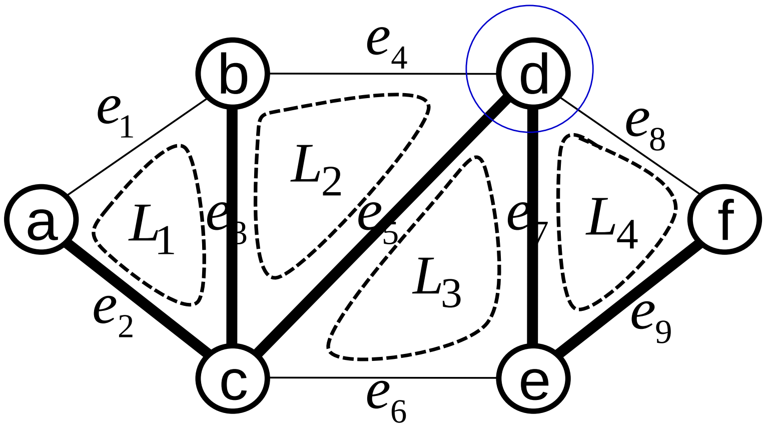
Creation of Optimal Tree should be considered





State Information of a Node

Each node has Tie-set Information to which the node belong



- **Ex: State Information in Node *d***
  - Incident Links : $\{e_4, e_5, e_7, e_8\}$
  - Adjacent Nodes : $\{b, c, e, f\}$
  - Tie-set Information : $\{L_2, L_3, L_4\}$

State Information of Node n

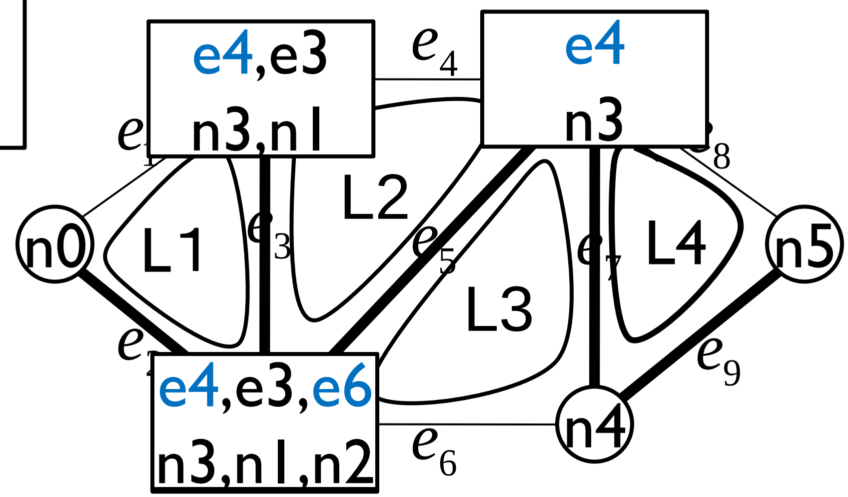
- **Incident Links :** Information of links connected to n.
- **Adjacent Nodes :** Information of nodes connected through incident links of n.
- **Tie-set Information :** Information of fundamental tie-sets to which n belongs.

DATIC

DATIC: Distributed Algorithm for Tie-set Information Configuration

FTM: Find Tie-set Message
Message to find Tie-set Information

- Only nodes connected to Co-Tree conduct DATIC.
- Two nodes connected a cotree link negotiate (let ni do DATIC).
- ni sends FTM to nj on cotree with info of ni and e(ni, nj).
- FTM is copied and sent out on T.
- FTM comes back to ni.
- ni sends FTM on a tie-set to notify its info.



Ex. Tie-set L2 whose Co-Tree is e4

- n1 and n3 connected e4 negotiate.
- n3 conducts DATIC

Other Tie-set Info are constructed in the same way simultaneously

Communiations $O(\mu|V|)$ / *Time* $O(D)$

Single Link Failure Recovery

Distributed Control for Link Failure

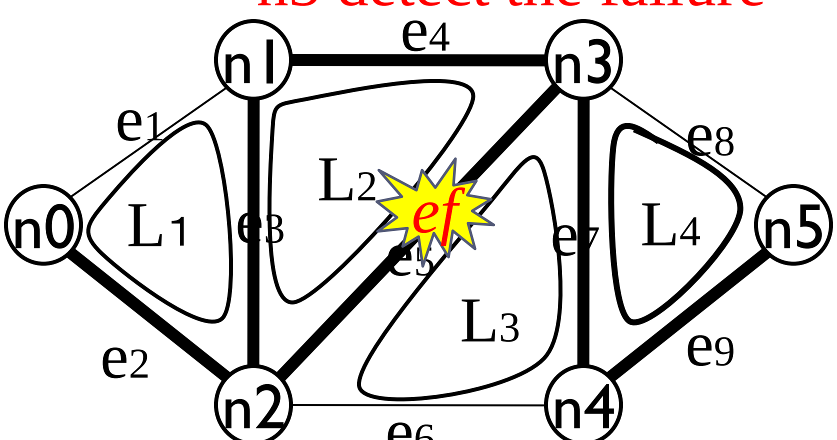
Tie-Set Information of n3

$$\mathbf{L} = \{L_2, L_3, L_4\}$$

$$L_2 = \{e_3, e_4, e_5\}$$

$$L_3 = \{e_5, e_6, e_7\}$$

n3 detect the failure



Procedure in a Node

① Blocking physical ports connected to *ef*

➡ Send "ClosePort" Message to n2

② Finding a tie-set L_i from Tie-set Information, where $e_f \in L_i$

➡ L2 and L3 are chosen

③ Determining a tie-set L_r to conduct route switching

➡ ComRoute is shifted in L2 (ef →e4)

④ Opening physical ports connected to the cotree link

➡ Send "OpenPort" Message to n1

Procedures after Failure

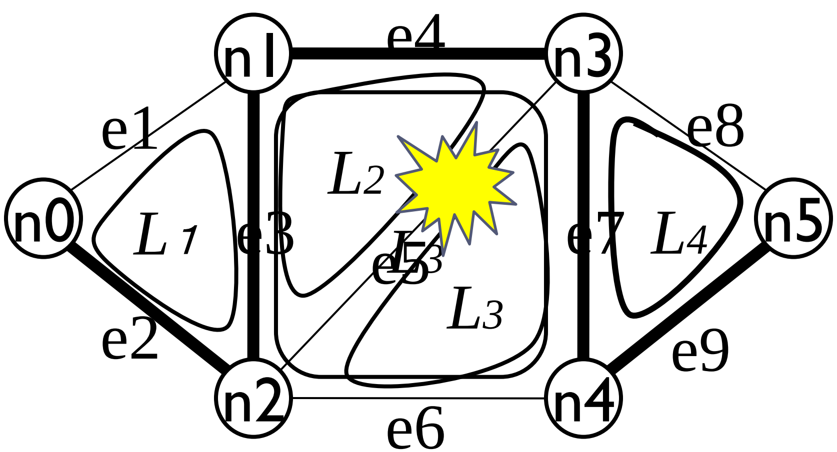
Tie-Set Info at n3

$$\mathbf{L} = \{L_2, L_3, L_4\}$$

$$L_2 = \{e_3, e_4, e_5\}$$

$$L_3 = \{e_5, e_6, e_7\}$$

$$L_3 = \{e_3, e_4, e_6, e_7\}$$



L-Transformation

The definition of $\oplus$ for a set A and a set B is defined as follows:

$$\begin{aligned} A \oplus B &= (A - B) \cup (B - A) \\ &= (A \cup B) - (A \cap B) \end{aligned}$$

Procedure in a Node

⑤ Apply L-Transformation

$$L_3 \leftarrow L_2 \oplus L_3$$

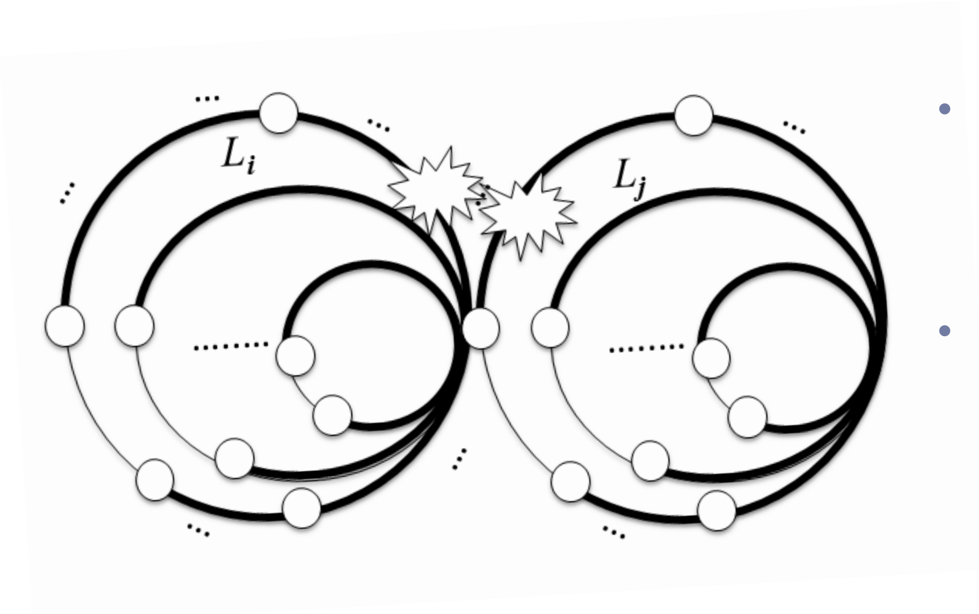
Get rid of Failed link e5 with L_2

Double-Link Failure Recovery

Classification of Double-Link Failure

Two major classes:

Class 1: Double-Link Failure is independent: $\mathbf{L}^{e_i^f} \cap \mathbf{L}^{e_j^f} = \emptyset$



- Essentially the same problem with Single link failure.
- The double-link failure is handled by different tie-sets *Li* and *Lj* individually and autonomously.

Class 2: Double-Link Failure is Dependent : $\mathbf{L}^{e_i^f} \cap \mathbf{L}^{e_j^f} \neq \emptyset$

Classification of Double-Link Failure

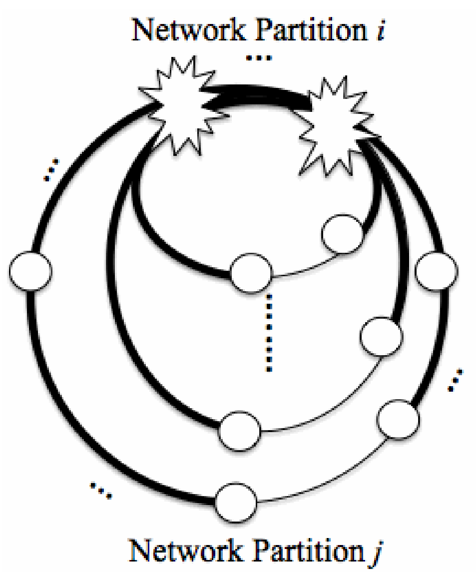
Class 2: Double-Link Failure is Dependent : $\mathbf{L}^{e_i^f} \cap \mathbf{L}^{e_j^f} \neq \emptyset$

- $L'_{E_{i,j}} = \mathbf{L}^{e_i^f} \oplus \mathbf{L}^{e_j^f}$
  - A class of tie-sets share exactly one failed link e_i^f or e_j^f
- $\left|L'_{E_{i,j}}\right| = 0$
  - Impossible to recover double-link failure
- $\left|L'_{E_{i,j}}\right| \geq 1$
  - Possible to recover double-link failure

Class 2

Sub-classes for *Class 2*:

- *Sub-class 1:* All the tie-sets with failed links share the same double-link failure, $|L'_{E_{i,j}}|$=0 (there exists no tie-set that share exactly one failed link with the tie-set having two failed links).

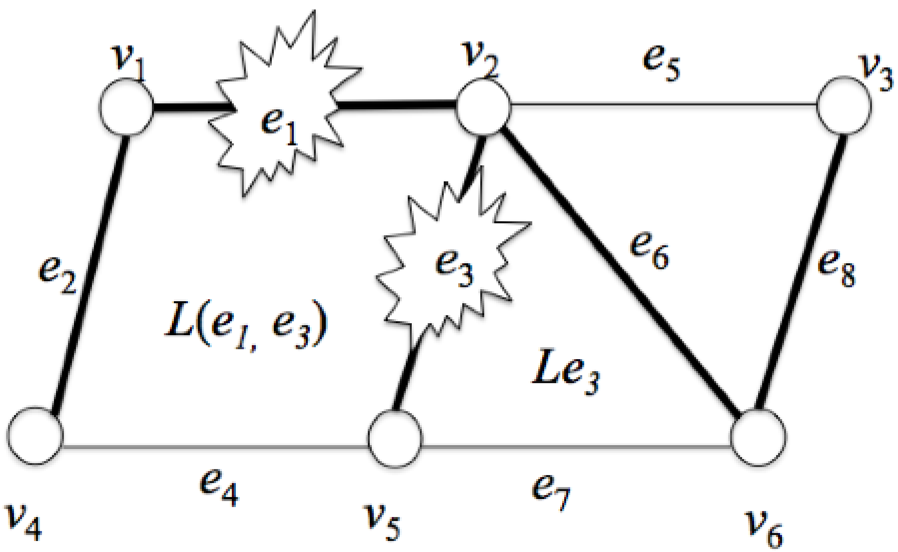


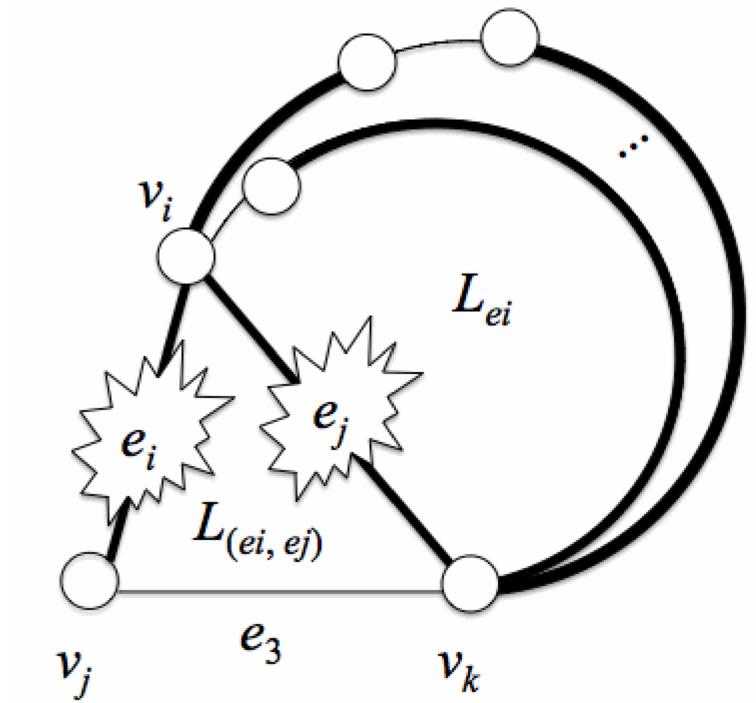
- The system is divided into two network partitions. There is no way to redirect the messages between separate network partitions.
- Such a distributed system can not be restored with either RSTP or TBFT.

Class 2

- *Sub-class 2*: there is one tie-set sharing exactly one failed link with the tie-sets that have two failed links, $|L'_{E_{i,j}}|$=1.

The tie-set with only one failed link should be responsible to recover its failed link.

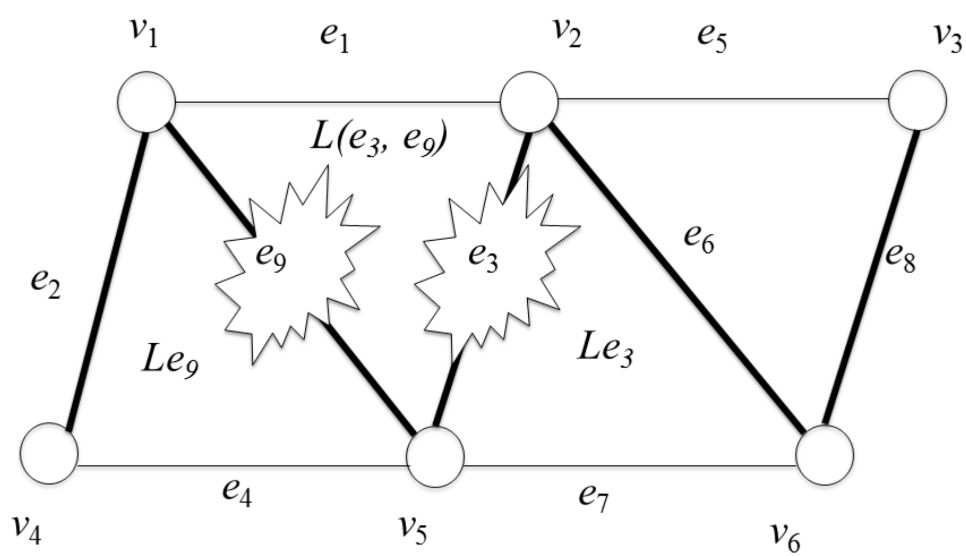




Class 2

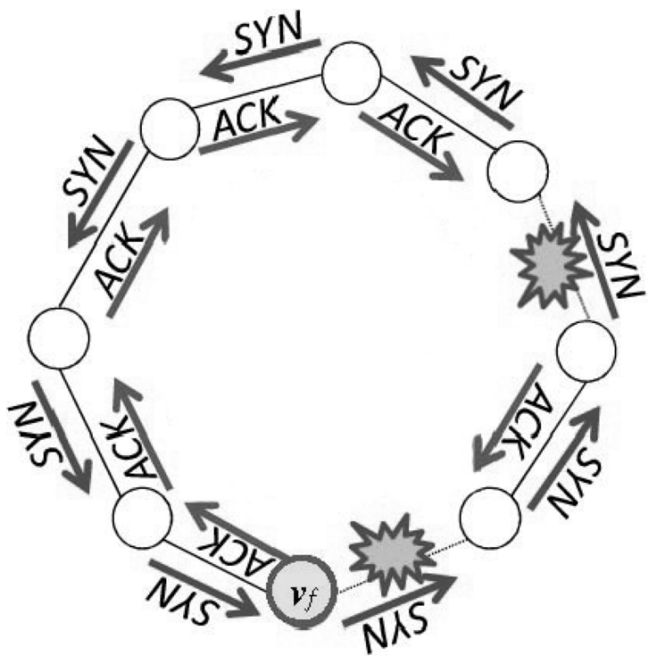
- *Sub-class 2*: there are at least two tie-sets that share exactly one of the failed links with the tie-set which has two failed links, $|L'_{E_{i,j}}| \geq 2$.

Tie-sets with only one failed link should be responsible to recover the failed link they have.



Failure detection mechanism.

- Each node v_i sends SYN message to an adjacent node v_a (periodically within some time period t_{syn}).
- Each node v_i receives ACK response message from v_a (during a time period t_{ack}).



The direction may be decided by the order of edges in Tie-set Information table, such as *EdgeTable* or *NodeTable*.

Failure recovery algorithm.

Algorithm for v_f that detects failure on e_f :

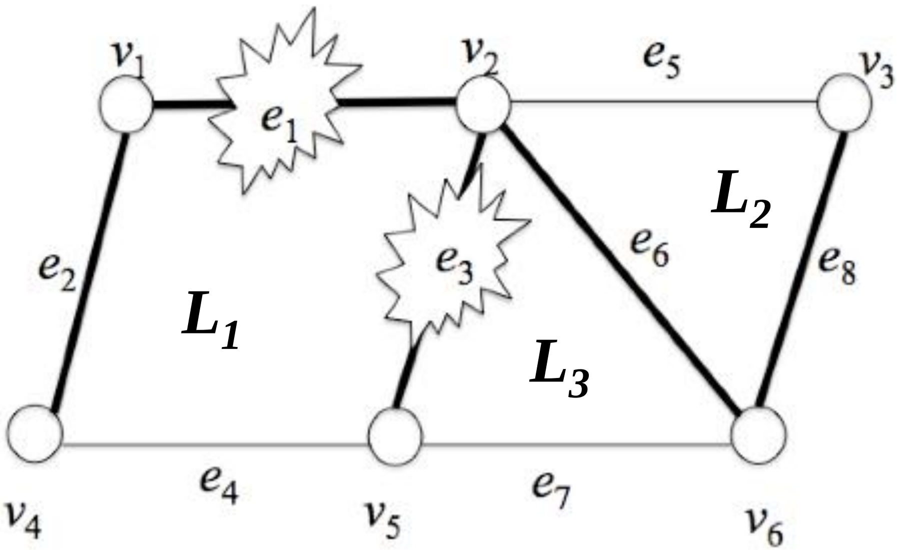
- *Step 1*: Blocking physical ports connected to e_f.
- *Step 2*: Selecting from *Tie-set Information* a set of tie-sets that include e_f.
- *Step 3*: Sharing information about the failure e_f within nodes of selected set of tie-sets.

Algorithm for v_i :

- *Step 4*: Receiving a message about failure and storing the failure information, sending a copy of the message to the next node in the tie-set.
- *Step 5*: Analyzing failure information and making a decision.
- *Step 6*: Opening physical ports connected to the cotree link.

Failure recovery algorithm.

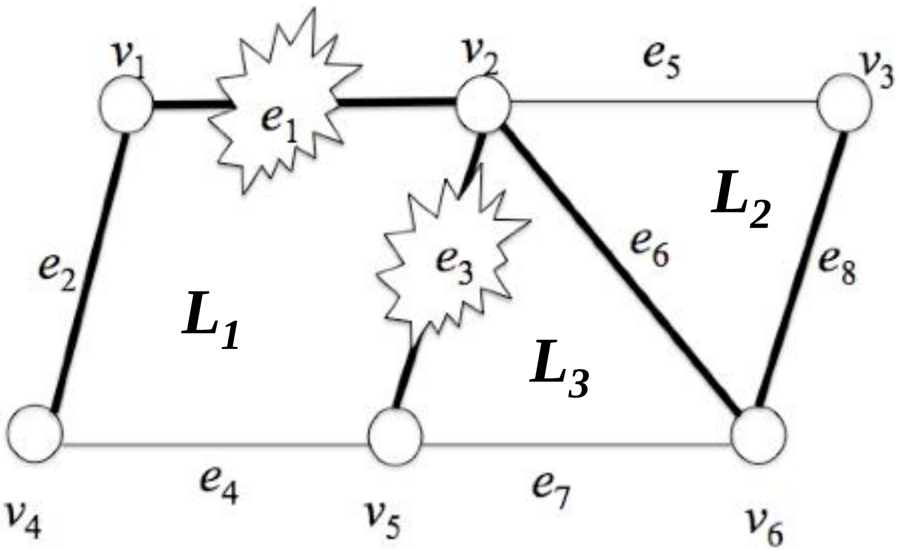
- *Step 1*: Blocking physical ports connected to e_f.



v_1 and v_2 block their ports to e_1.
v_2 and v_5 block their ports to e_3.

Failure recovery algorithm.

- *Step 2*: Selecting from *Tie-set Information* a set of tie-sets that include e_f.



Tie-Set Information of v_2

$$\mathbf{L}=\{L_1, L_2, L_3\}$$

$$L_1=\{e_1, e_2, e_3, e_4\}$$

$$L_2=\{e_5, e_6, e_8\}$$

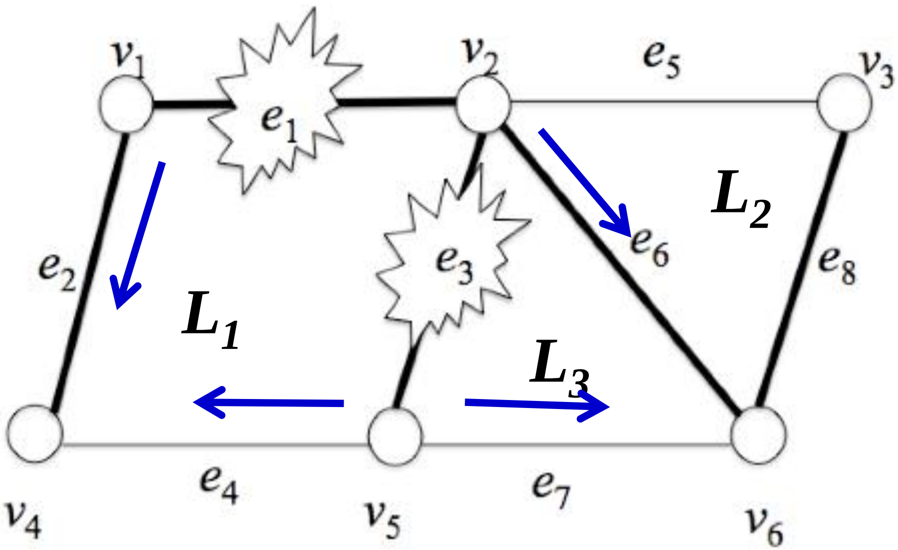
$$L_3=\{e_3, e_6, e_7\}$$

$$e_1 \in \mathbf{L}_{e_1}=\{L_1\}$$

$$e_3 \in \mathbf{L}_{e_3}=\{L_1, L_3\}$$

Failure recovery algorithm.

- *Step 3*: Sharing information about the failure e_f within nodes of selected set of tie-sets.

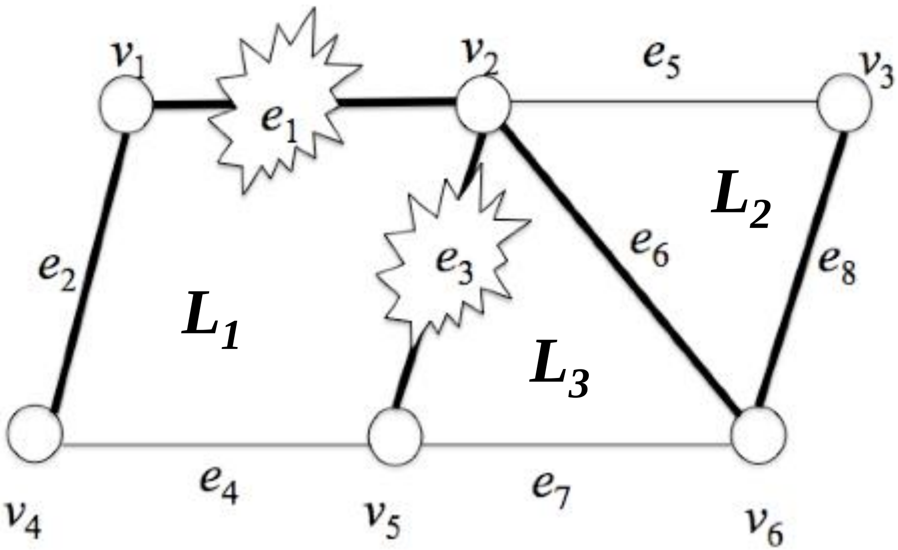


Failure on $e_3 \in L_1, L_3$

$v_2 \longrightarrow v_6$

Failure recovery algorithm.

- *Step 4*: Receiving a message about failure and storing the failure information, sending a copy of the message to the next node in the tie-set.



Failure information on v_6

$e_3 \in L_1, L_3$

Failure on $e_1 \in L_1$

$v_4 \longrightarrow v_5$

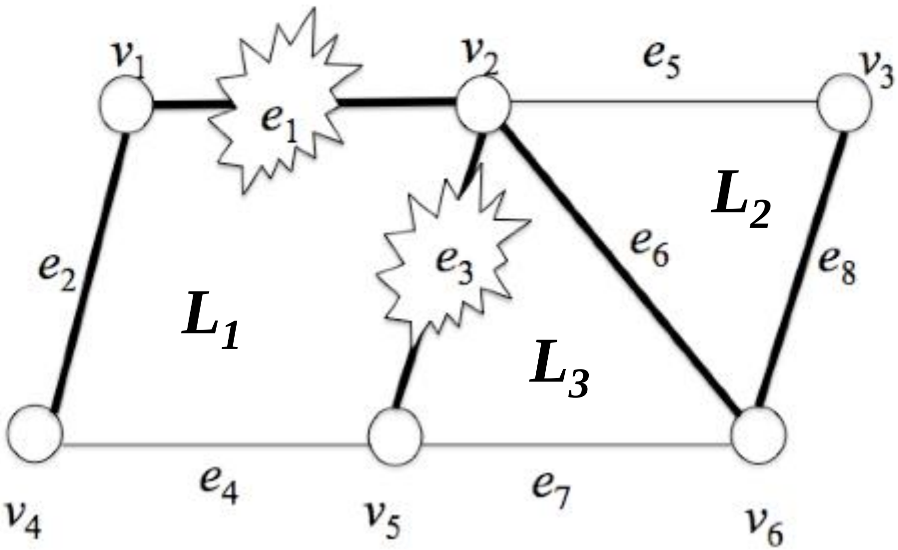
Failure information on v_5

$e_1 \in L_1$

$e_3 \in L_1, L_3$

Failure recovery algorithm.

- *Step 5*: Analyzing failure information and making a decision.



Analysis on v_6:
Connected to cotree e_7
$\{e_3\} \in L_3$ => recover e_3

Analysis on v_5:
Connected to cotree e_4,e_7

$\{e_1 , \cancel{e_3}\} \in L_1$ => recover e_1
~~$\{e_3\} \in L_3$~~

- *Step 6*: Opening physical ports connected to the cotree link.

Evaluation & Comparison with RSTP

Evaluation

- Theoretical Analysis with Distributed Algorithms
  - Communication Complexity and Time Complexity

Experiments

- The Number of Route Switched Points
  - Throughput
  - Delay in Failure Recovery
- The Number of Hops (from Node that detects a link failure to Node that completes the restoration)
  - Estimation of Recovery Time
- The Number of Influenced Nodes (that change the Communications Port States)
  - Communications Reliability and Quality

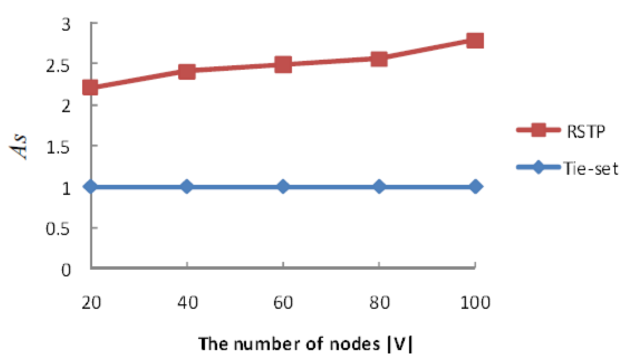
Superior than RSTP in terms above

Simulation and Experiments against RSTP

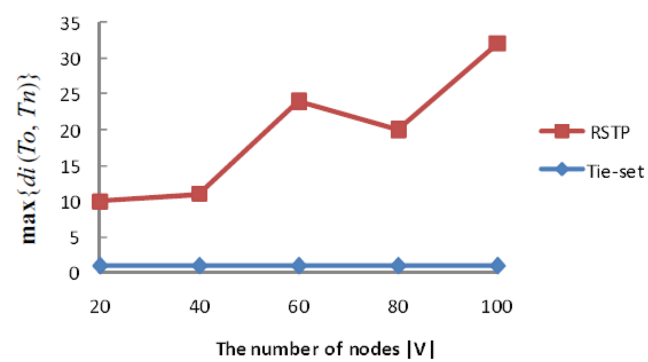
Route Switching Points

- T_o : Tree before route switching
- T_n : Tree after route switching
- $d(T_o, T_n)=|T_o - T_n|$: Distance between T_o and T_n
- $d_i(T_o, T_n)$: Distance between T_o and T_n when link failure occurs on a tree link $e_i(\in T)$

$$A_s = \frac{\sum_{i=1}^{\rho} d_i(T_o, T_n)}{\rho}, (i = 1, 2, \ldots, \rho(= |T|))$$



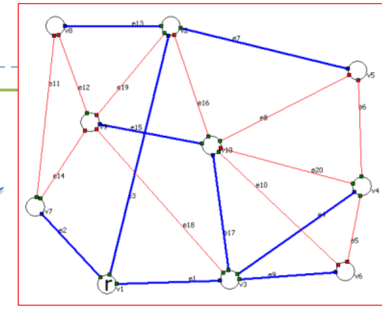
The average times of route switching (A_s)



The maximum times of route switching

Tie-set-based recovery requires only one switching
⇔ RSTP requires more than one switching

Experiments against RSTP



Influenced Nodes

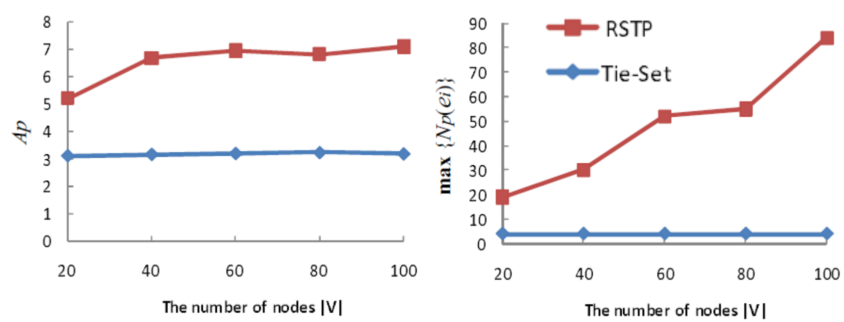
Simulation networks are designed to be redundant

Ex: Network configuration consisting of 10 nodes and 20 links

Nodes that Change Physical Port States

$N_p(e_i)$: The number of nodes that change their physical port states when link failure occurs on a tree link $e_i(\in T)$

$$A_p = \frac{\sum_{i=1}^{\rho} N_p(e_i)}{\rho}, (i = 1, 2, \ldots, \rho(= |T|))$$

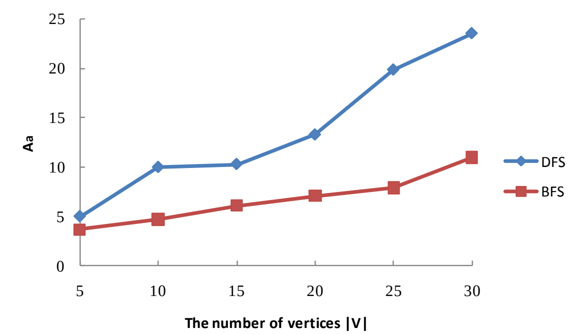


The average number of nodes changing port states (A_p)

RSTP influences a network to a greater degree than the Tie-set-based restoration

Nodes that Change State Information

$N_a(e_i)$: The number of nodes that change their state information by advertisement when link failure occurs on a tree link $e_i(\in T)$



The average number of nodes changing state information (A_a)

BFS is more suitable than DFS since BFS can reduce the number of nodes that change their state information in comparison with DFS