Online Distributed Trace Synchronization

Masoume Jabbarifar

Michel Dagenais

December 6, 2012

École Polytechnique, Montreal

Outline

Streaming Mode Incremental Clock Synchronization

Reference Node Selection in Dynamic Tree

Minimum Spanning Tree Maintenance in Dynamic Tree

- Key problem
- Methodology
- Experimental results
- Conclution

Key problem

Problems and solutions

- **Accuracy:** The generated events are at the nanosecond scale.
- **Performance:** The analysis speed must be the same as, or higher than, the data rate.
- **Data buffering:** The gathered data from a computer cluster is huge but the required data is smaller.
- **Delay:** Little delay must be added by analysis for tracing to diagnose problems and attacks in real time

- **Convex-hull algorithm:** This algorithm guarantees the best accuracy.
- **Incremental algorithm:** It refreshes synchronization as it receives accurate data.
- **Layered improvements:** Proposed online synchronization methods improve performance in different layers;

  1) Individual connection

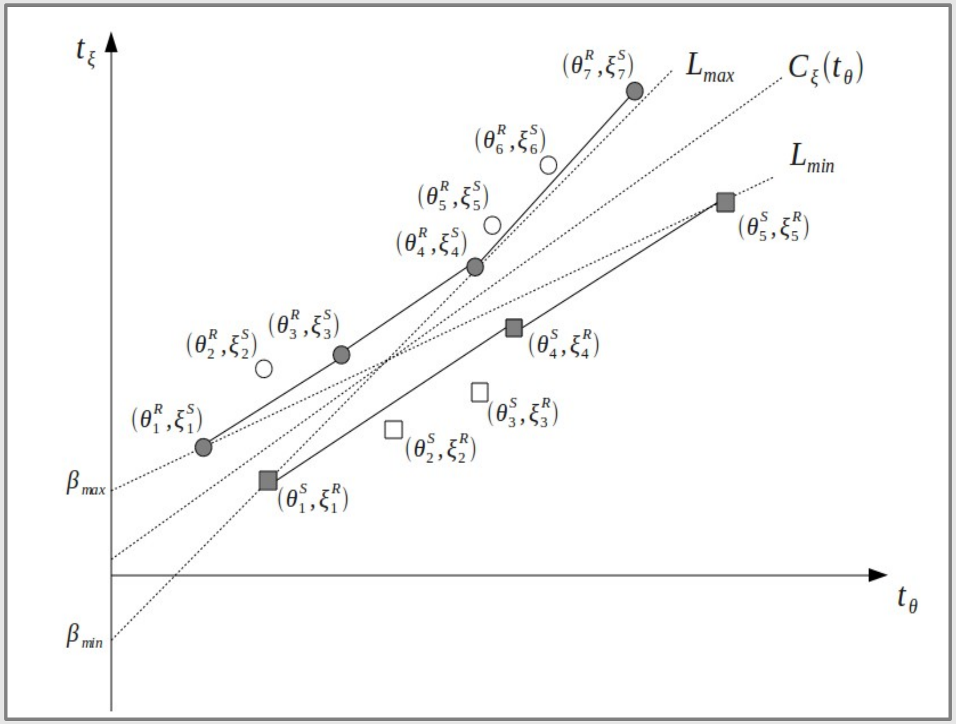
  2) In a computer network

  3) Time reference updates

“Streaming Mode Incremental Clock Synchronization“

Two Clocks Synchronization

Convex-Hull



- Sent and Received messages sets
- Guarantees no message inversion
- Two lines with Max & Min slope

$$L_{max} = \alpha_{max}\theta + \beta_{min}$$

$$L_{min} = \alpha_{min}\theta + \beta_{max}$$

$$Accuracy = \alpha_{max} - \alpha_{min}$$

- The bisector of the angle formed by these two lines

$$C_A(t) = \alpha C_B(t) + \beta$$

Time Interval based approaches (1)

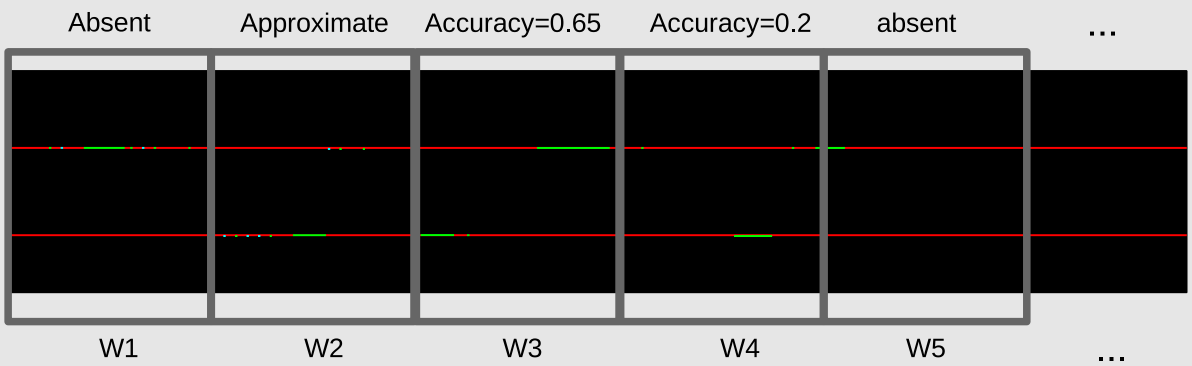
Independent Window

Advantages

- Performance
- No buffering
- Simple to implement

Disadvantages

- Accuracy

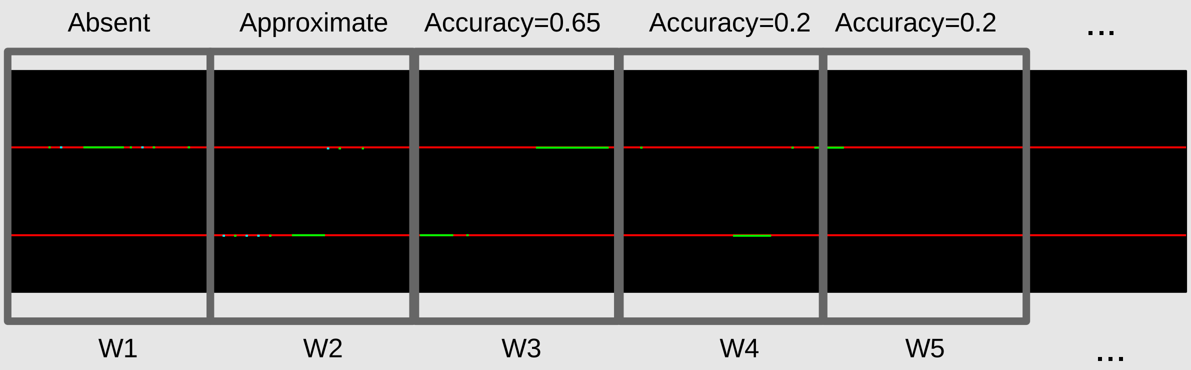


Time Interval based approaches (2)

Replacement approach

Advantages

- Performance
- Simple to implement

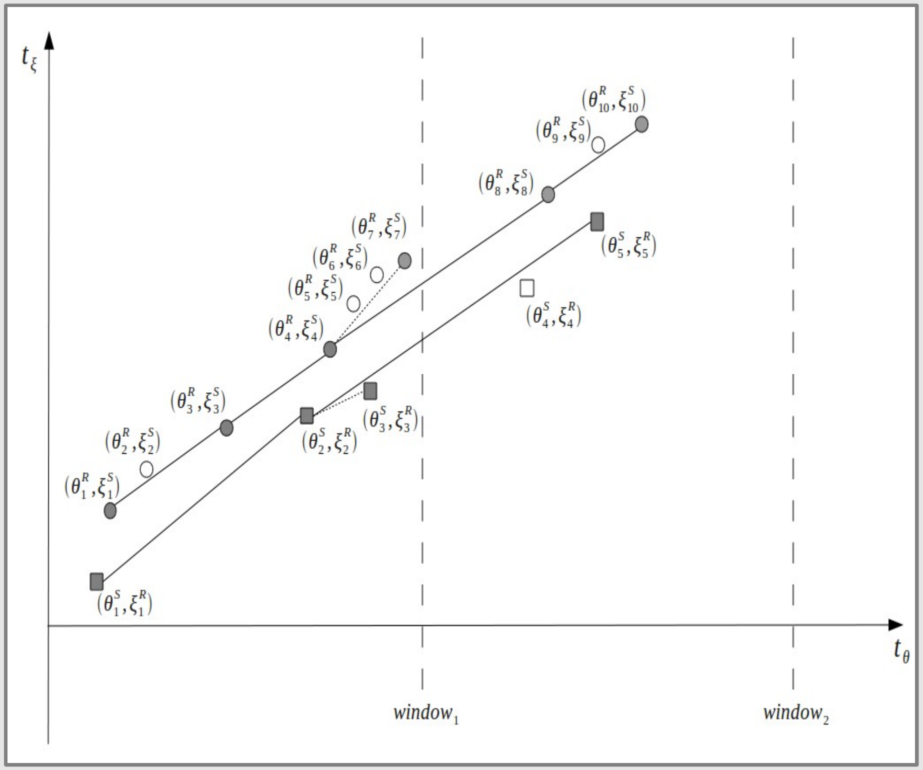


Disadvantages

- High level accuracy is unobtainable

Time Interval based approaches (3)

Correlated approach



Advantages

- The highest level of accuracy
- No buffering

Disadvantages

- Processing is postponed to the end of each window

Fully incremental Approach (1)

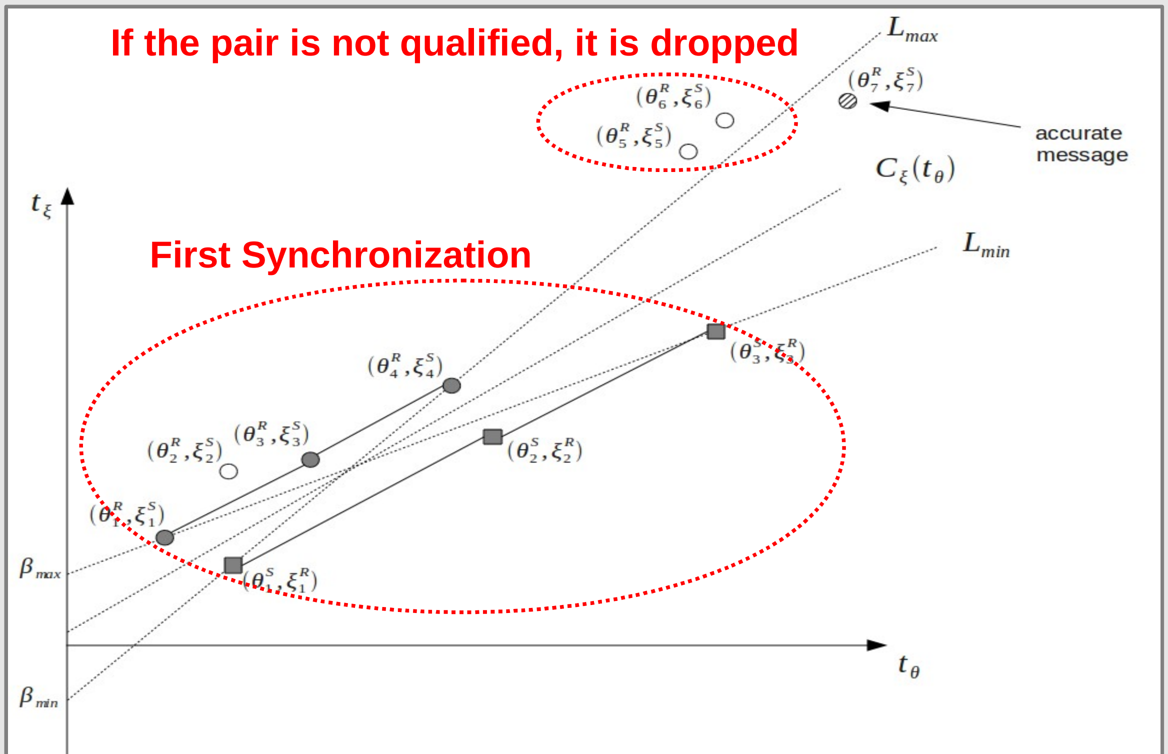
No window concept

- The highest level of accuracy
- No buffering
- No delay

Fully Incremental Approach (2)

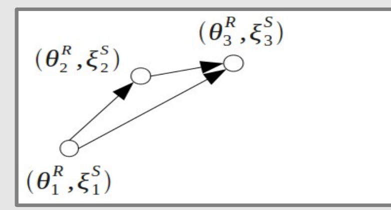
Step1: wait to establish at least three messages in each direction

Step2: Look for accurate messages

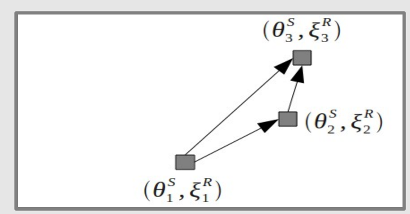


Detection method:

Lower bound



Upper bound

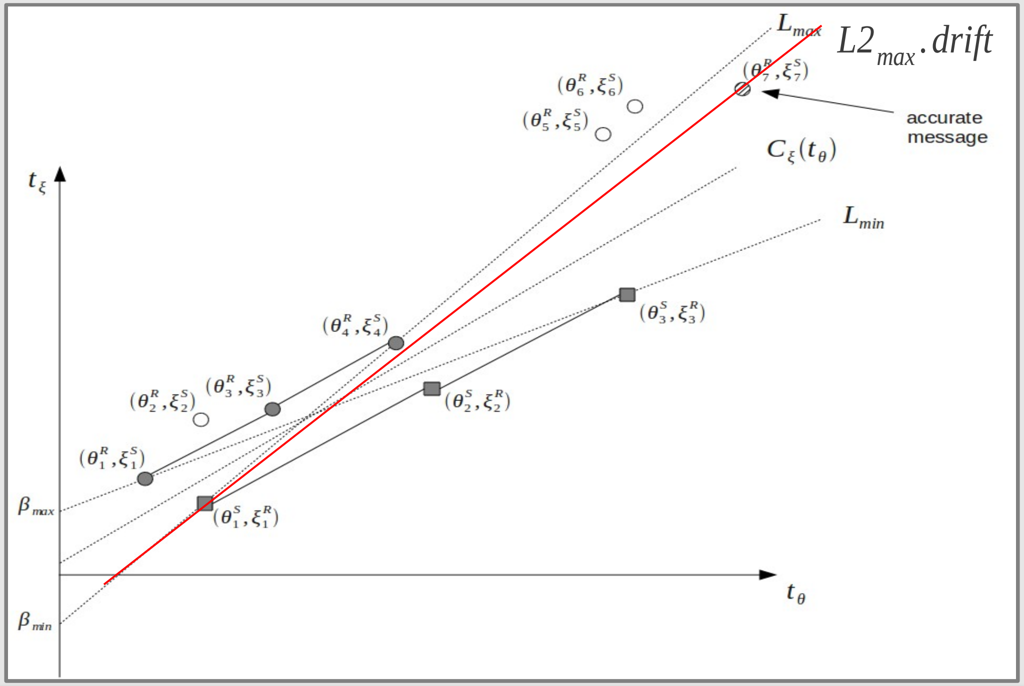


Fully Incremental Approach (3)

Accurate Message Improves Sycnhronization Accuracy

- accuracy is the difference between the minimum and maximum possible drifts between the two clocks

$$Accuracy1 = L1_{max}.drift - L1_{min}.drift$$



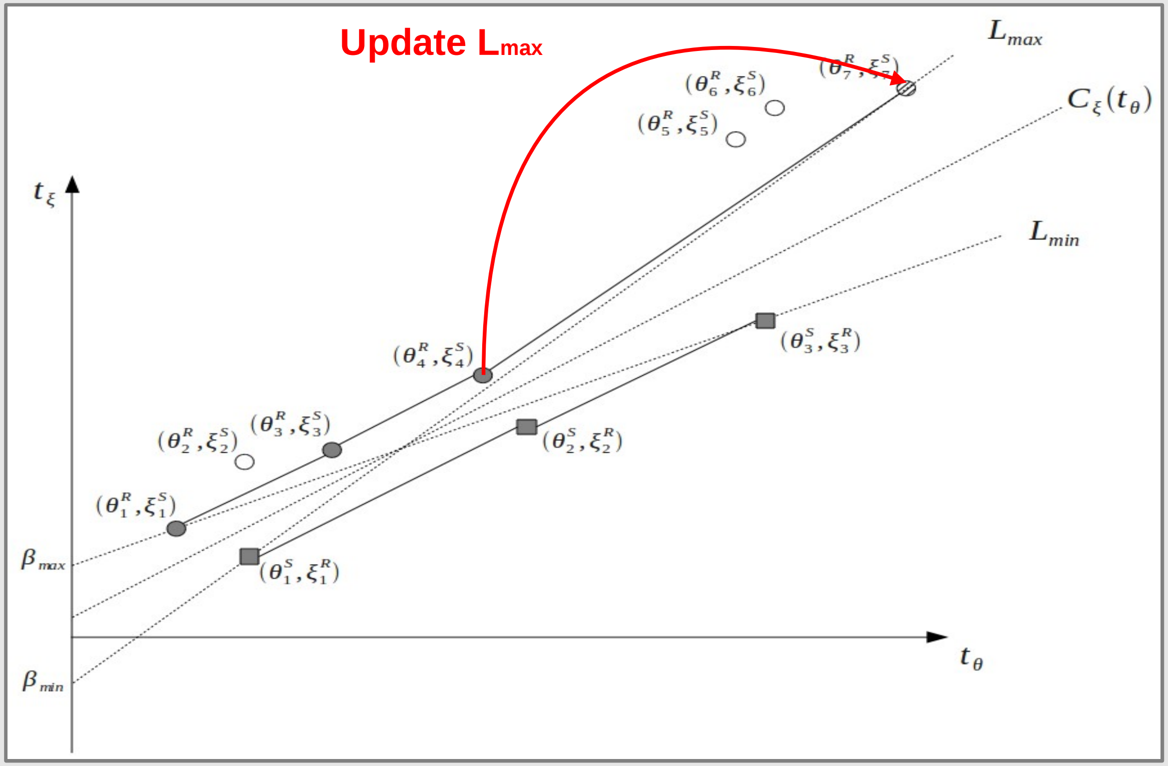
$$L1_{max}.drift > L2_{max}.drift$$

$$Accuracy1 > Accuracy2$$

Fully Incremental Approach (4)

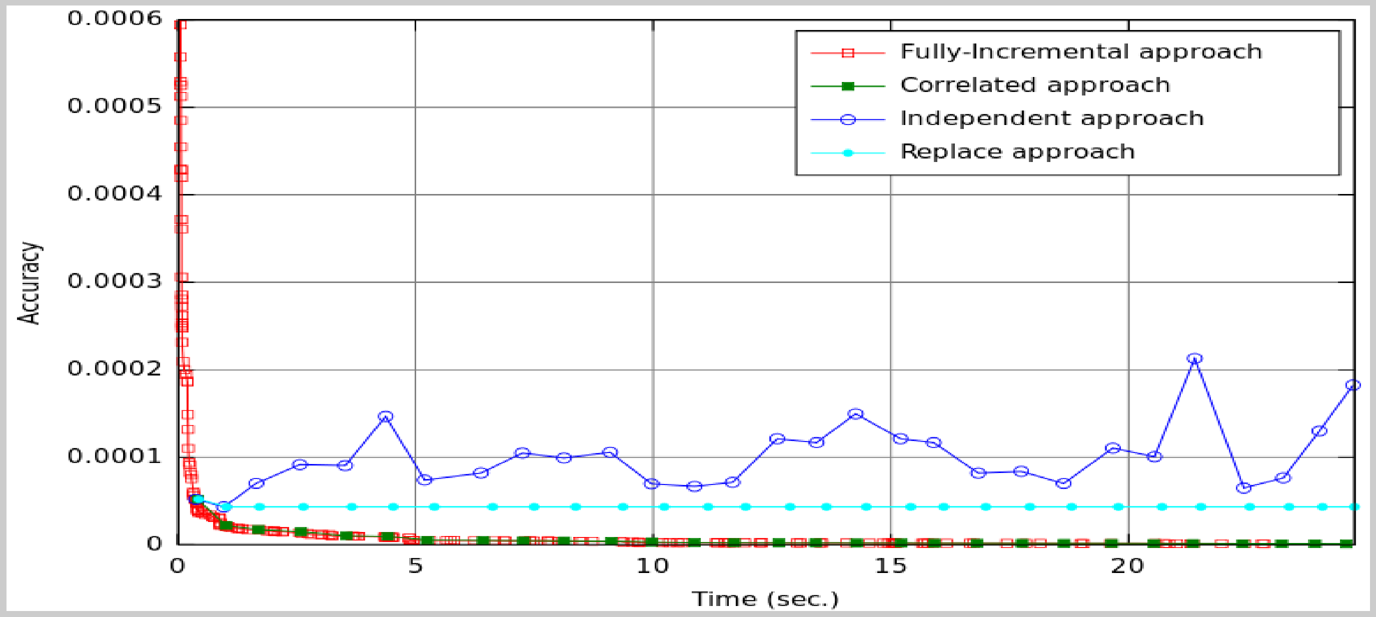
Step3: adjust-bounds

- Only ***Lmax*** or ***Lmin*** is changed when an accurate pair is received

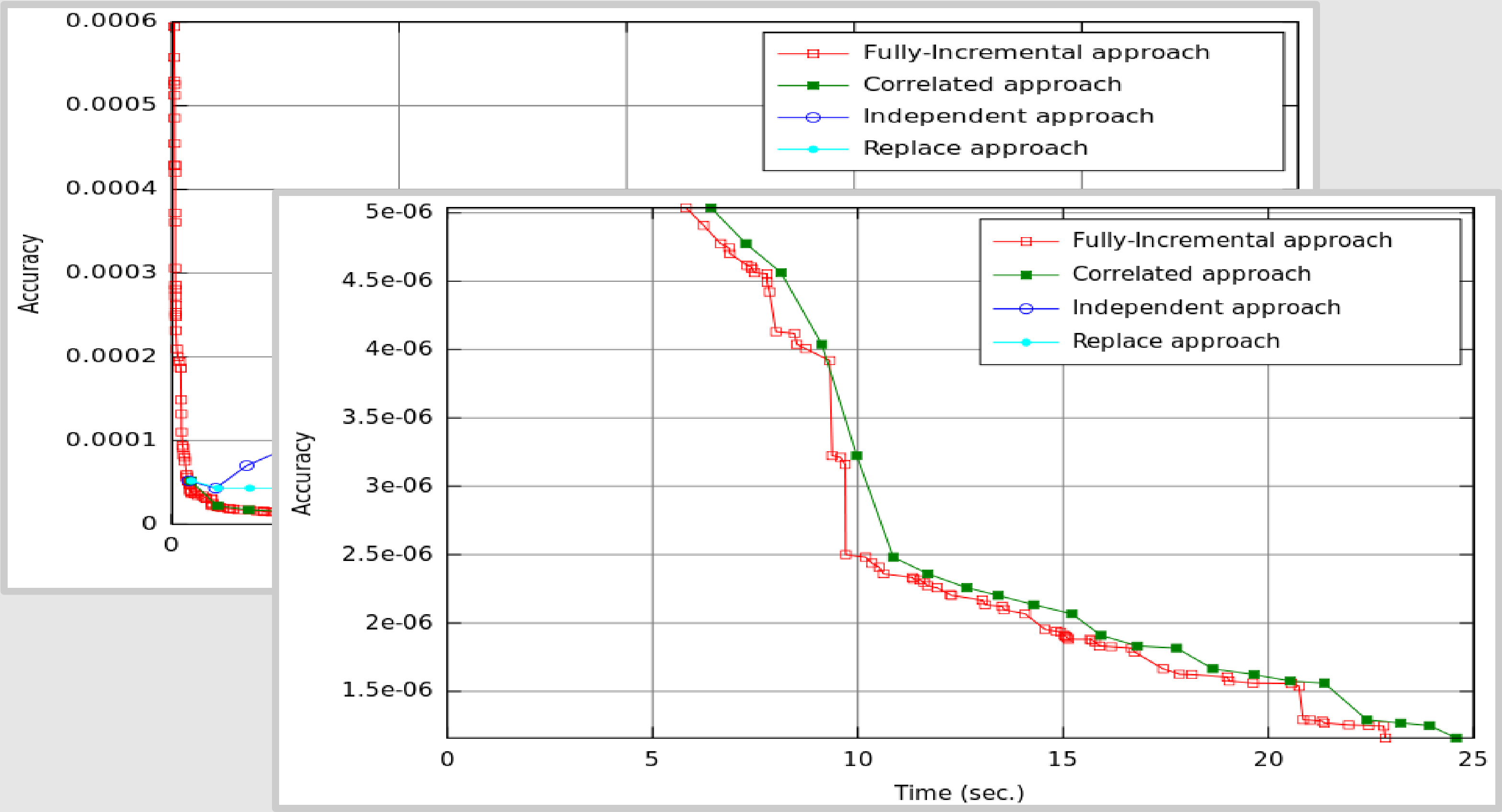


Each synchronization in the Fully Incremental approach requires O(1) time, on average

Results (1)


0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0
Accuracy
0
5
10
15
20
Time (sec.)
Fully-Incremental approach
Correlated approach
Independent approach
Replace approach

Results (2)



Average delay =0.91 sec

Fully incremental Approach (5)

No window consideration

Advantages

- Performance: Each synchronization requires O(1) time, on average.
- High level time accuracy
- No delay
- No buffering

Features

- **Analysis:** Appropriate data are filtered prior to synchronization computations.

“Reference Node Selection in Dynamic Tree“

Key problem

Refrence Node Selection

1) For any node in Spanning Tree, the shortest paths to every other node are computed.
2) The best time reference node (RN) is the node which has the smallest paths to all.
3) All nodes in the network synchronize their time with the reference node through those paths.

- It takes O(n^2) time to find the RN.
- A fixed RN?
  - Costs synchronization accuracy.
  - A single point of failure.

Reference Node Position

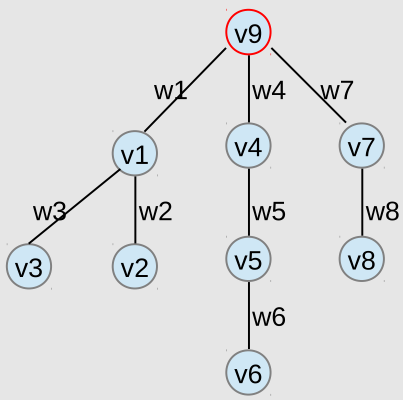
Total accuracy

- The position of RN is critical to decrease the total time conversion error through all paths.

$$T: < v_1, v_2, \ldots, v_n >$$

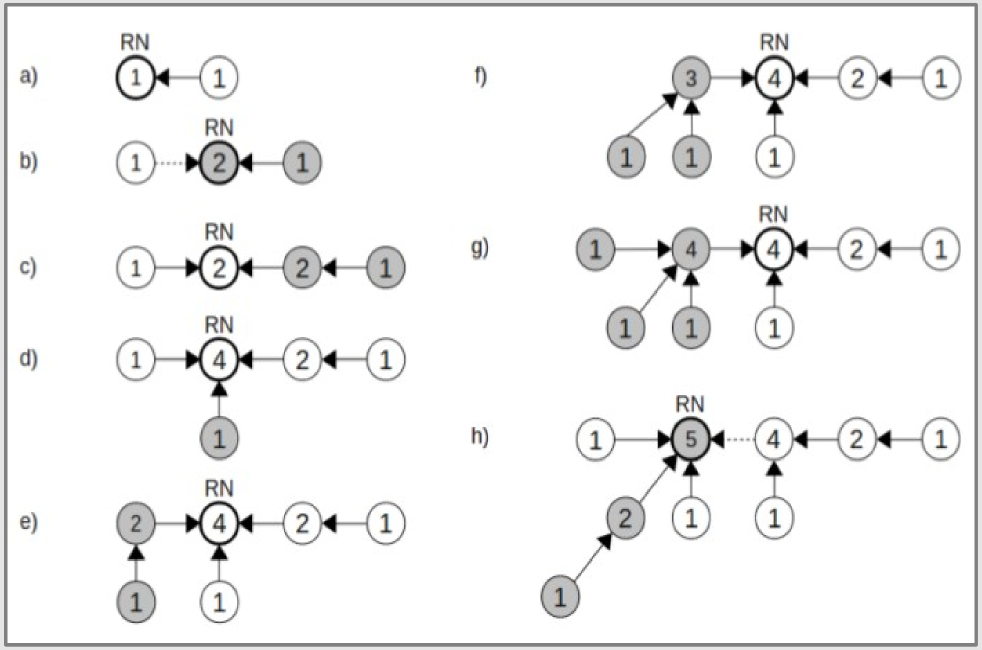
$$P_{v_i} < v_i, v_{i+1}, \ldots, RN >$$

$$\text{Total Synchronization Error}_T = \sum_{i=1}^{n} \sum_{j=1}^{l} weight_{e_j} \; on \; P_{v_i}$$



Dynamic Reference Node (1)

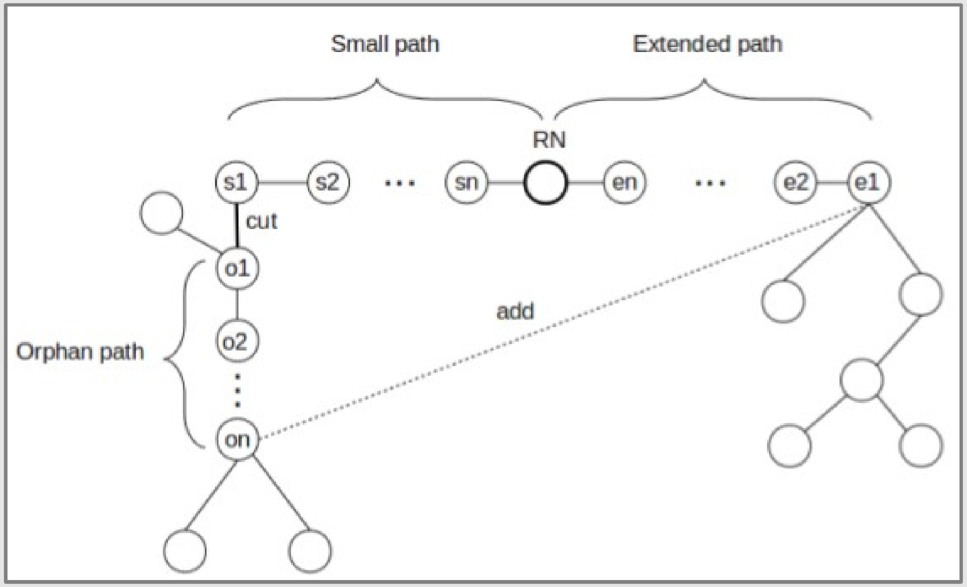
New Node Connection



- The number inside each vertex shows the Descendant Size
- Propagation along the path from the parent of “v” to the RN
- Comparison between two nodes

Dynamic Reference Node (2)

Cycle/ Cut and Add (1)

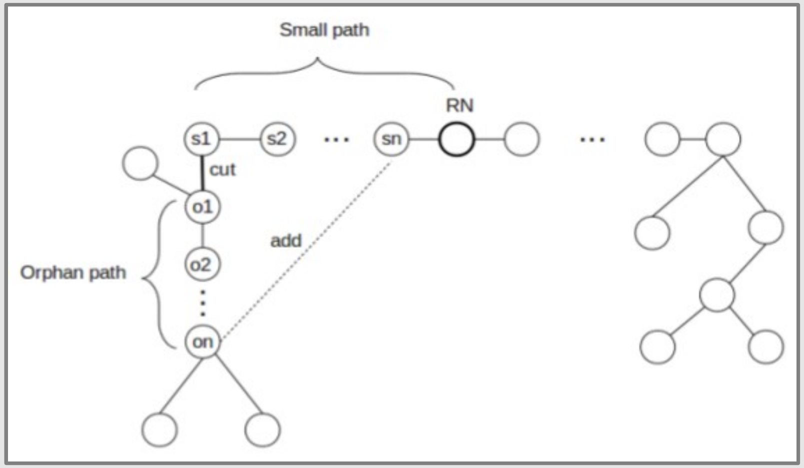


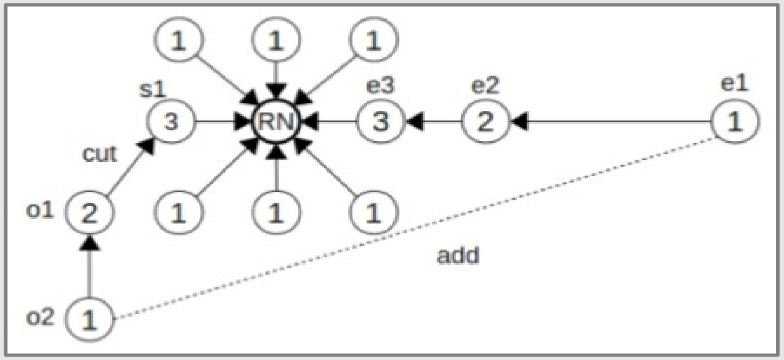
RN is in Extended path

- Orphan path: < o2 , ..., on−1 >
- Small path: < s1 , s2 , ..., sn >
- Extended path: < e1 , e2 , ..., en >

Dynamic Reference Node (3)

Cycle/ Cut and Add (2)





- RN does not change when:
  - Cut and add are in one side of RN
  - RN has many branches

$$reverse(e_1)$$

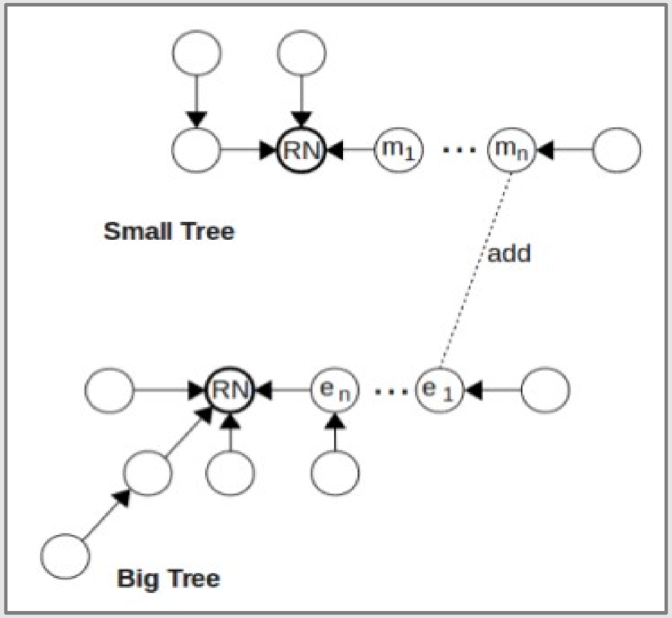
$$\triangle = \underbrace{\xi(o_1)}_{\text{cut impact}}$$

$$\xi(\chi) = \overbrace{treeSize(\chi) - [\underbrace{\xi(parent(\chi))}_{e_n} + \triangle]}^{\text{the new DescendantSize for previous reference node}}$$

$$RN(\tau_i) = \begin{cases} \chi & \text{if } \xi(\chi) > \underbrace{treeSize(\chi)/2}_{\text{tree balance value}} \\ \text{search in extendedPath} & \text{otherwise} \end{cases}$$

Dynamic Reference Node (4)

Joining Two Trees



1) Edge (RN_{ST}, RN_{BT}) → RN_{BT}

2) Edge (RN_{ST}, e_i) →
   Reverse path <e_i, e_{i+1}, …, RN_{BT}>

3) Edge (m_i, RN_{BT}) → RN_{BT}
   Reverse path < RN_{ST}, m_1, …, m_n>

4) Edge (m_{ST}, e_{BT}) → RN_{BT}
   Reverse path < RN_{ST}, m_1, …, m_n>
   Candidate RN Є <e_i, e_{i+1}, …, RN_{BT}>

Data set (1)

Number of each operation, from one million operations

	Nodes	Insertion	Join	Cycle: Stay[1]	Cycle: Remove[2]	updateEdge
$Dataset_1$	10000	4991	2503	45449	946879	178
$Dataset_2$	20000	9892	5052	76404	908556	96
$Dataset_3$	30000	15005	7496	102672	874761	66
$Dataset_4$	40000	19955	10021	125650	844322	52
$Dataset_5$	50000	24959	12519	145733	816753	36
$Dataset_6$	60000	29953	15022	164104	790885	36

[1] the new connection stay in loop and other edge in cycle is removed by MST algorithm.
[2] the new connection has the highest weight in cycle and is removed by MST algorithm

RN changes

	Insertion					Join					Cycle				
	Number[1]	win_{RN}[2]	%	$lose_{RN}$[3]	%	Number[4]	win_{RN}	%	$lose_{RN}$	%	Number[5]	win_{RN}	%	$lose_{RN}$	%
$Dataset_1$	4991	732	15%	4259	85%	2503	1464	58%	1039	42%	45449	1186	3%	44263	97%
$Dataset_2$	9892	1435	14%	8457	86%	5052	2972	70%	2080	30%	76404	1259	2%	75145	98%
$Dataset_3$	15005	2270	15%	12735	85%	7496	4439	59%	3057	41%	102672	1508	2%	101164	98%
$Dataset_4$	19955	2986	15%	16969	85%	10021	5847	58%	4174	42%	125650	1365	1%	124285	99%
$Dataset_5$	24959	3675	15%	21284	85%	12519	7298	58%	5221	42%	145733	1432	1%	144301	99%
$Dataset_6$	29953	4373	15%	25580	85%	15022	8839	59%	6183	41%	164104	1776	1%	162328	99%

[1] The total number of cases where a vertex add to existent tree. Note that other cases belong to two new vertices connections which makes a new tree. In recent case one of vertices is selected as RN and there is no computation to find RN.
[2] The number of cases that RN changes.
[3] The number of cases that RN does not change.
[4] The number of cases that two trees merge in the forest.
[5] The number of cases that an edge makes cycle in one of trees in the forest.

Data sets (2)

Merge positions

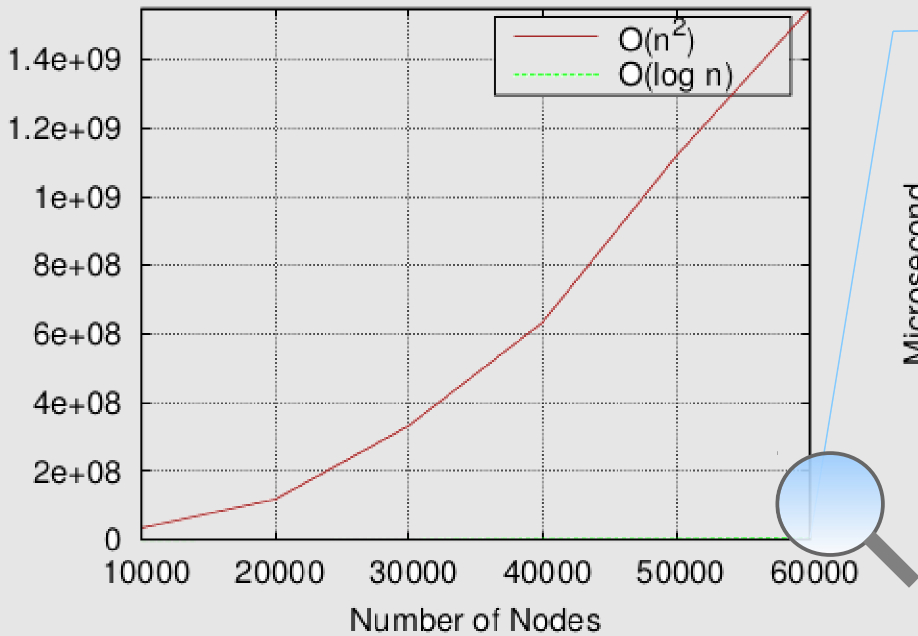
	$RN_s - RN_b$	$m_i - RN_b$	$RN_s - e_i$	$m_i - e_i$
$Dataset_1$	0	190	80	2233
$Dataset_2$	0	440	161	4451
$Dataset_3$	0	595	232	6669
$Dataset_4$	0	819	339	8863
$Dataset_5$	0	1035	420	11064
$Dataset_6$	0	1238	474	13310

Number of descendant Size update in each operation

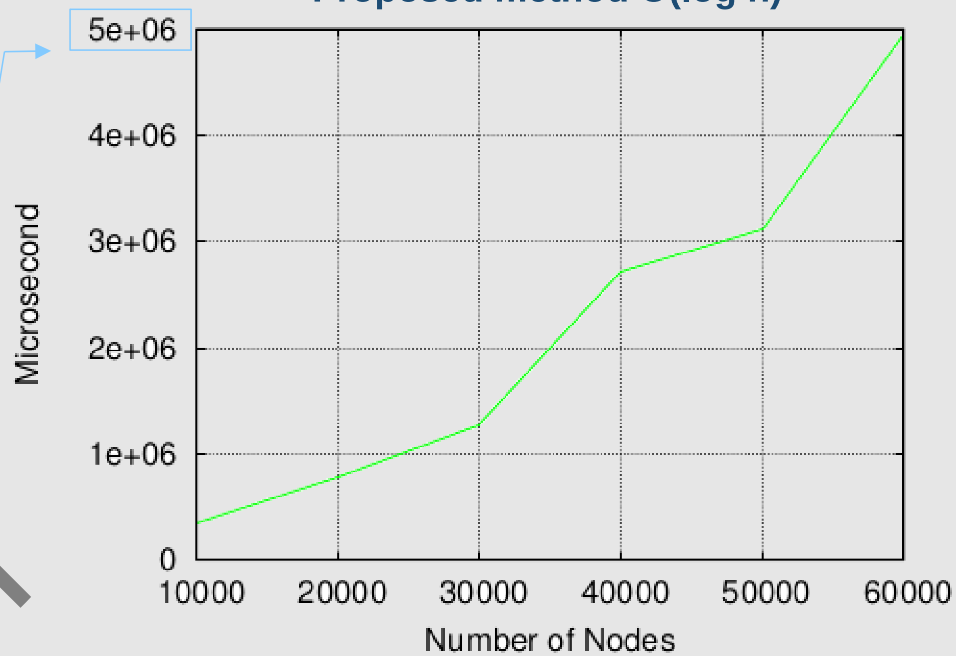
	Insertion			Join			Cycle		
	Number	Update	Avg. Distance to RN	Number	Update	Avg. Distance to RN	Number	Update	Avg. Distance to RN
$Dataset_1$	4991	89985	18	2503	50930	20	45449	3431773	75.5
$Dataset_2$	9892	246827	25	5052	128818	25.5	76404	7614996	100
$Dataset_3$	15005	366967	24	7496	191811	26	102672	12250493	119
$Dataset_4$	19955	513153	26	10021	275419	27.5	125650	15740773	127
$Dataset_5$	24959	762460	31	12519	398419	32	145733	18804027	129
$Dataset_6$	29953	1018792	34	15022	540992	36	164104	23822448	145

Result

Updating the reference node in a dynamic network with one million operations



Proposed method O(log n)



10000 nodes= 32.53 sec.

60000 nodes= 1549.92 sec.

10000 nodes= 0.33 sec.

60000 nodes= 4.92 sec.

“Minimum Spanning Tree Maintenance in Dynamic Tree“

Kruskal's Algorithm for Minimum Spanning Tree

- Running Time = O(m log n)   m = edges, n = nodes
- The steps are:
  1) The edges are placed in a priority queue
  2) Until we have added n-1 edges
     i. Extract the lowest edge from the queue
     ii. If it forms a cycle, eliminate it
     iii. Else add it to the tree

Challenges in Dynamic Minimum Spanning Tree

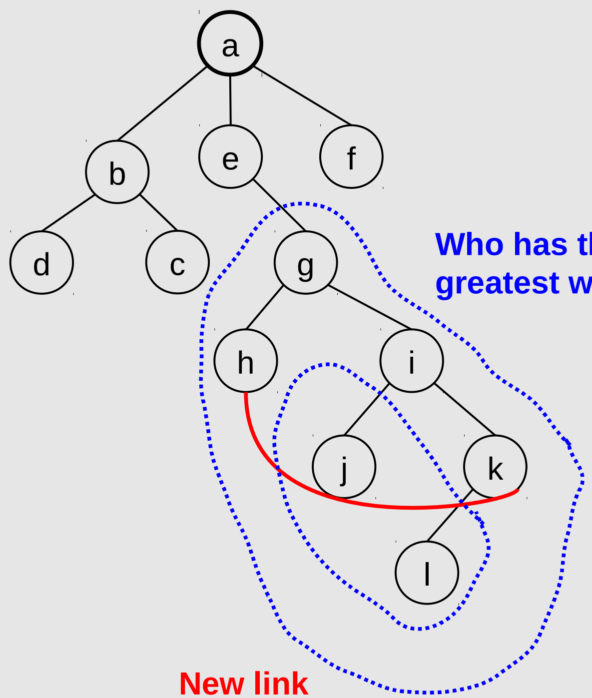
- Two cases can occur:
  1) Edge is in Tree and new weight is less than previous (no effect)
     - It is the characteristic of Fully incremental Approach

$$Accuracy_{new} < Accuracy_{previous}$$

  1) Edge is not in Tree
     - Add without cycle (no effect)
     - Add with cycle (effect)

Dynamic MST

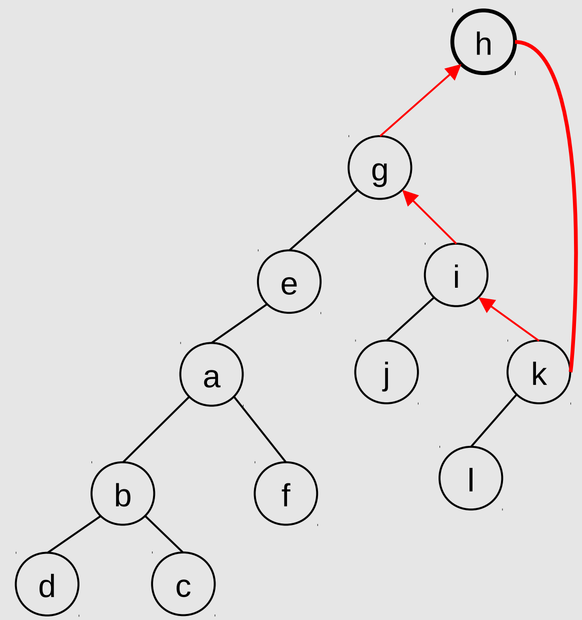
Loop in MST



Evert (h)

parent(g)= h
parent(e)= g
parent(a)= e

Make vertex *h* the root of its tree



Data set

Number of each operation, from one million operations

	Nodes	Added link[1]	Removed link[2]	Total
Dataset1	10000	45449	946879	992328
Dataset2	20000	76404	908556	984960
Dataset3	30000	102672	874761	977433
Dataset4	40000	125650	844322	969972
Dataset5	50000	145733	816753	962486
Dataset6	60000	164104	790885	954989

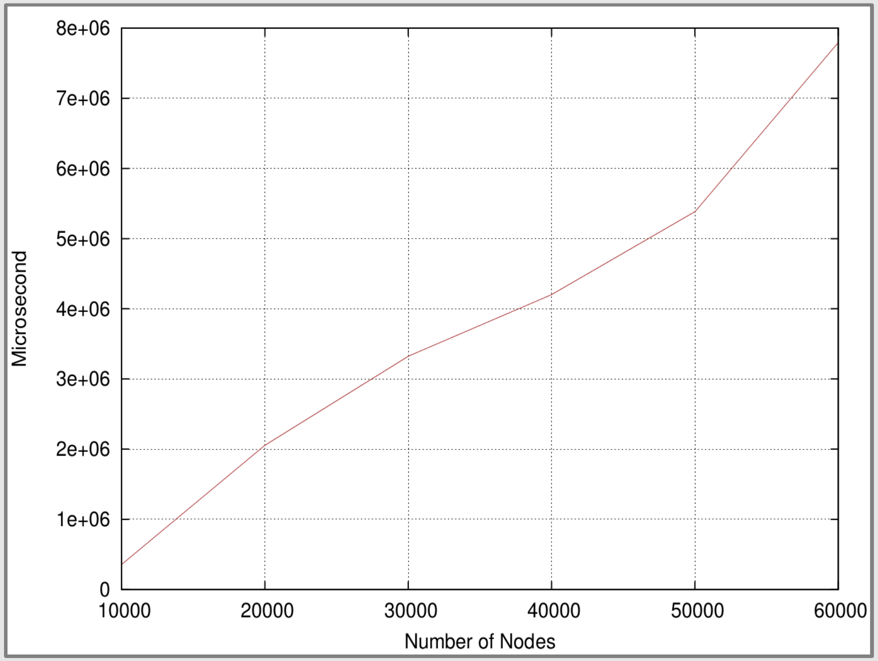
1) The new connection is added to the MST and one of other edges in the cycle is removed by MST algorithm

2) The new connection has the highest weight in cycle and is removed by MST algorithm

Result

- Old approach requires 1.440216 sec. to find MST <u>once</u> in a cluster with 10000 simulated nodes.
- For 992328 changes in MST: 992328* 1.440216 = 1429166.662848 sec. ~ 396 hours = 16.5 days!!

Proposed method O(log n)



10000 nodes= 0.36 sec.

60000 nodes= 7.79 sec.

Conclusion

- Resynchronization rate per edge
  - Performance: Each synchronization requires O(1) time, on average.
  - High level time accuracy
  - No delay
  - No buffering

- Resynchronization rate per network
  - Reference node selection
  - Synchronization path update
  - Performance: Each update requires O(log n) time, on average.

- Error reduction and continuity
- Scalability
- Robustness

Thank you

DORSAL

Online Distributed Trace Synchronization

www.lttng.org

E-Mail:

masoume.jabbarifar@polymtl.ca

michel.dagenais@polymtl.ca