A Repair Mechanism for Fault-Tolerance for Tree-Structured Peer-to-Peer Systems

Cédric Tedeschi[1]
Eddy Caron[1], Frédéric Desprez[1], Charles Fourdrignier[2], Franck Petit[2]

1 - LIP laboratory
2 - LaRIA laboratory

HiPC'2006 - December 20, 2006

Outline

Outline

Context

- Resource discovery
- Target platforms
  - large scale
  - no central infrastructure
  - dynamic joins and leaves of nodes
- P2P systems
  - Purely decentralized algorithms
  - Scalable algorithms to retrieve objects
  - Fault-tolerance

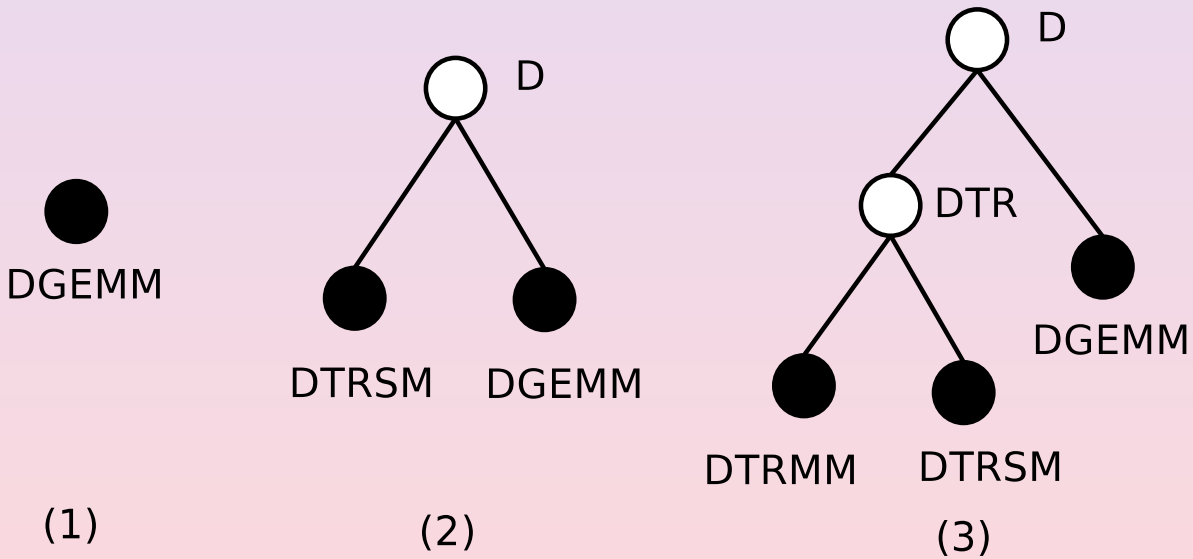
Trie-based lookup

- Range queries (logarithmic latency)
- Approaches
  - Skip Graphs (Aspnes and Shah – 2003)
  - PHT (Ramabhadran, Ratnasamy, Hellerstein, Shenker – 2004)
  - Nodewiz (Basu, Banerjee, Sharma, Lee – 2005)
  - P-Grid (Datta, Hauswirth, John, Schmidt, Aberer – 2005)
  - DLPT (Caron, Desprez, Tedeschi – 2006)

Outline

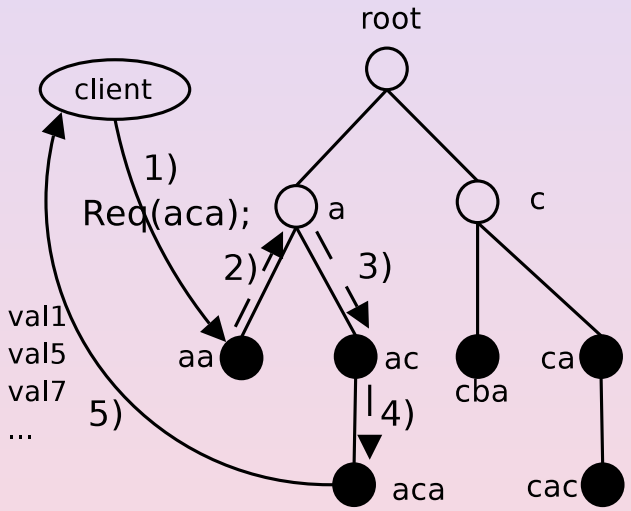
DLPT

- Distributed Lexicographic Placement Table
- Based on a greatest common prefix tree (GCP Tree)
- Distributed and dynamic building and mapping



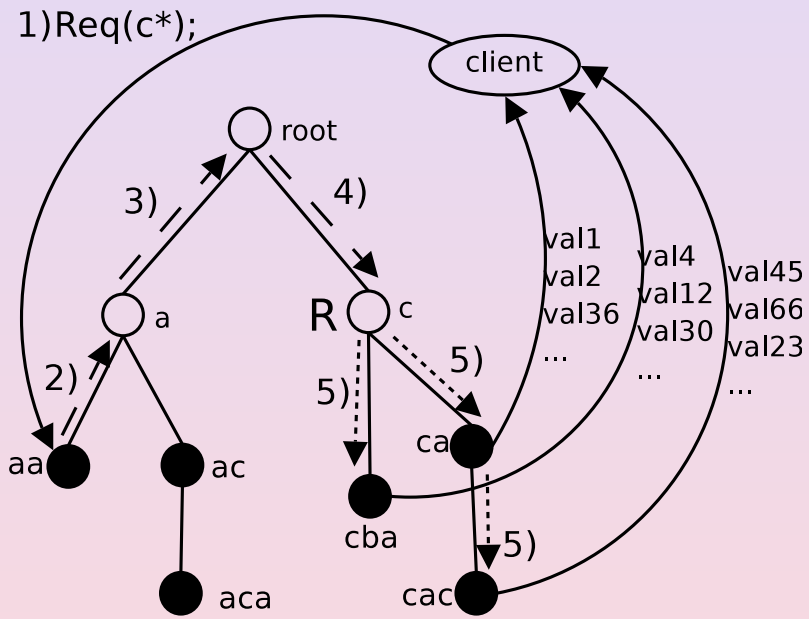
Querying the GCP tree

- Exact match query
- Range query (automatic completion)
- Multicriteria lookup



Querying the GCP tree

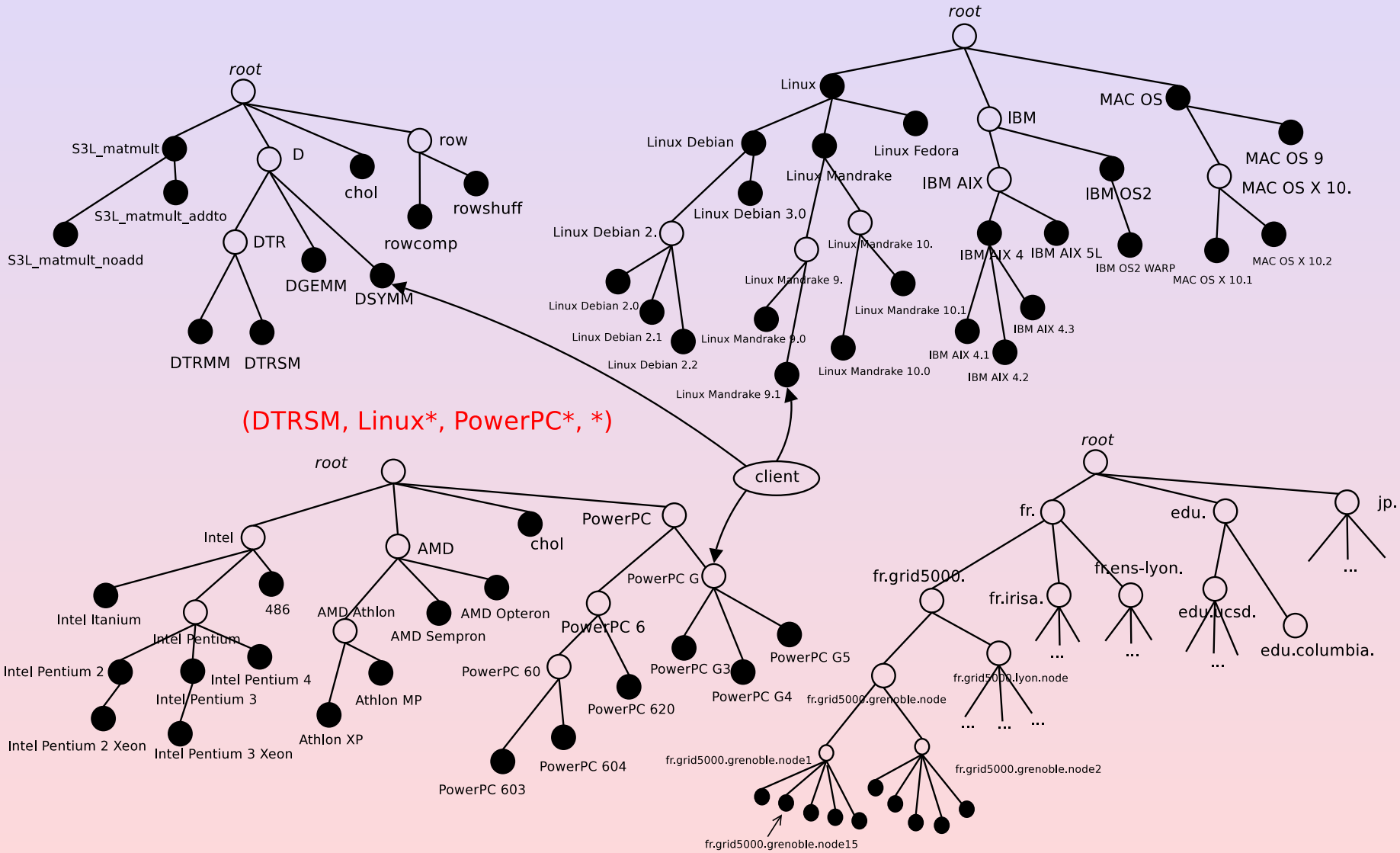
- Exact match query
- Range query (automatic completion)
- Multicriteria lookup



Querying the GCP tree

- Exact match query
- Range query (automatic completion)
- Multicriteria lookup

Querying the GCP tree



Replicating or repairing ?

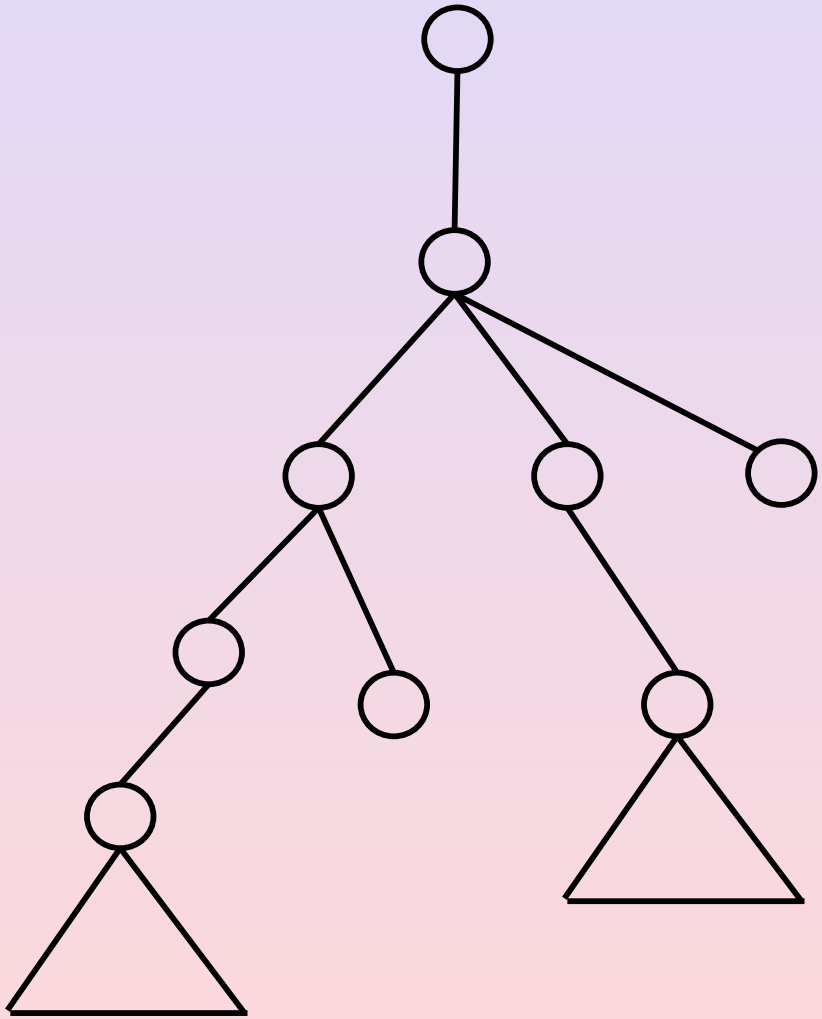
- Replication of the tree
  - preventive approach
  - how to tune the replication factor ?
  - costly to maintain (resources/local state)
- Repair
  - let the tree split into a forest
  - *a posteriori* reconnection and reordering of nodes

Outline

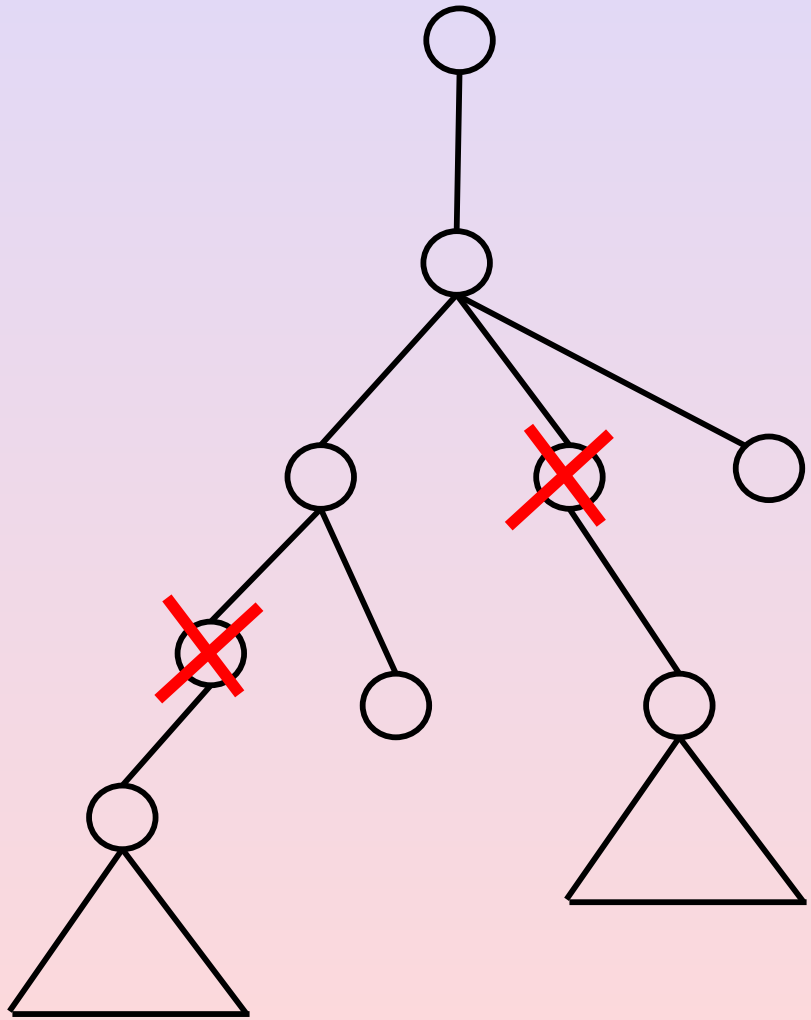
Two phases

- Tree recovery (forest $\rightarrow$ tree)
- Tree reorganization (tree $\rightarrow$ GCP tree)

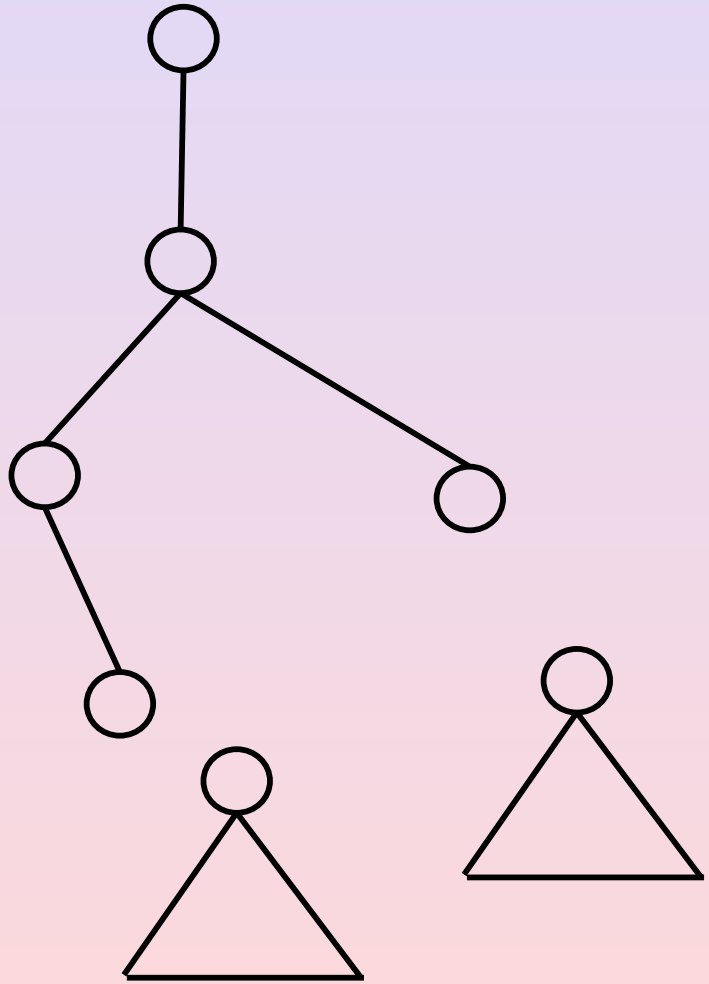
Tree reconnection



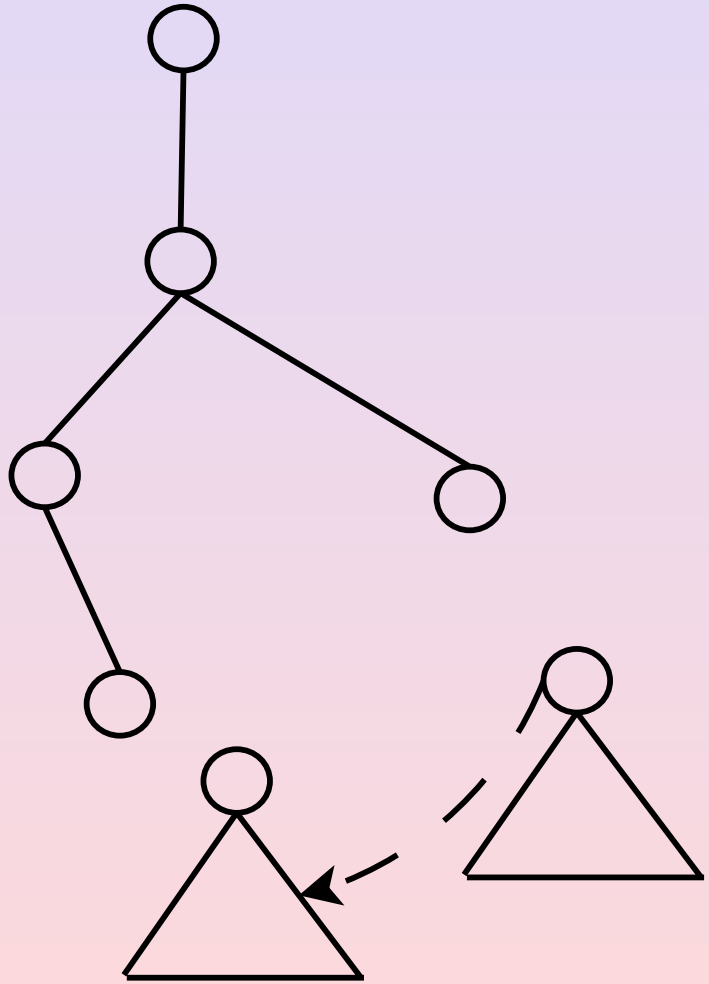
Tree reconnection



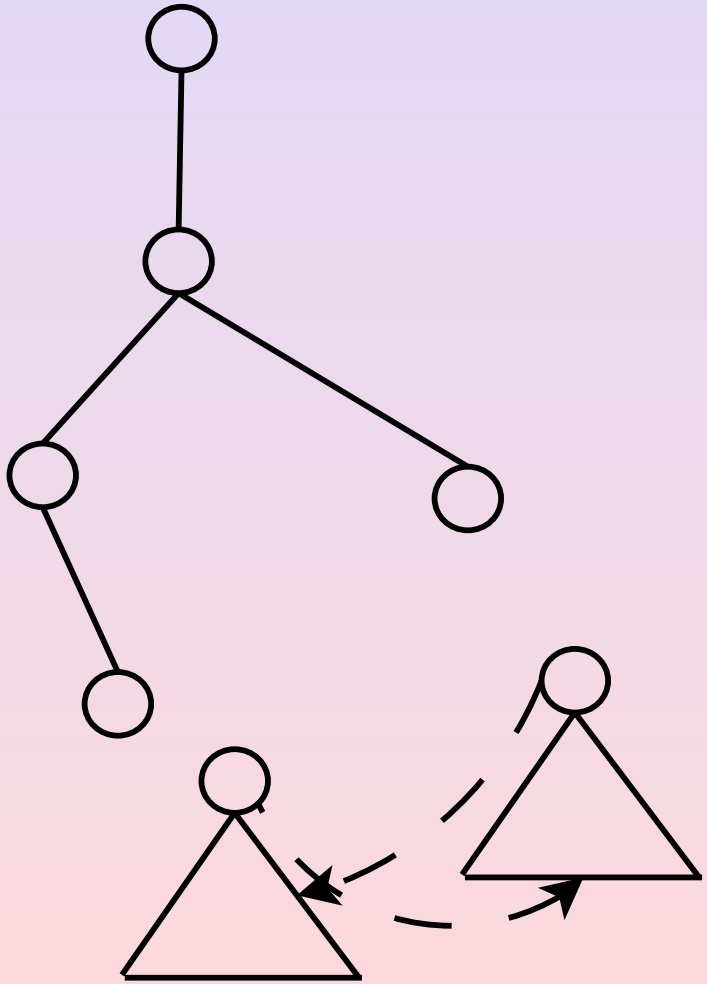
Tree reconnection



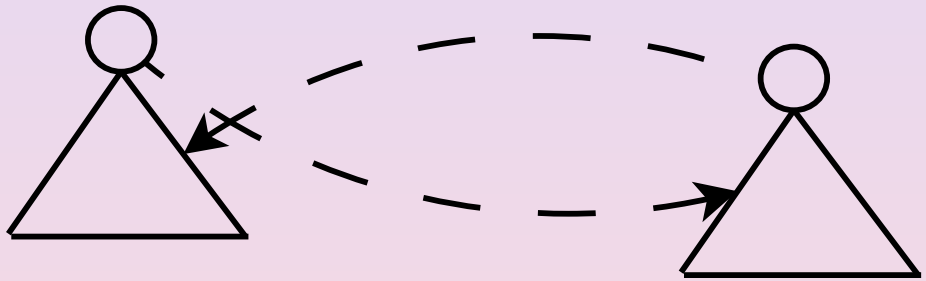
Tree reconnection



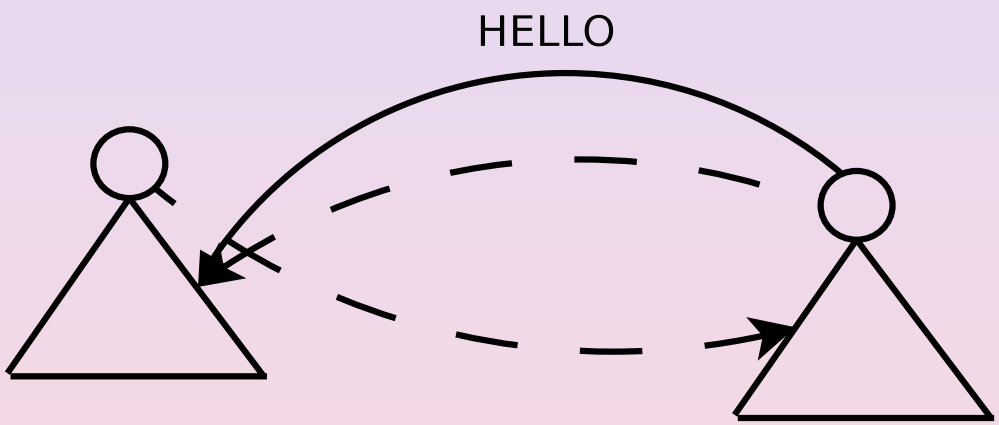
Tree reconnection



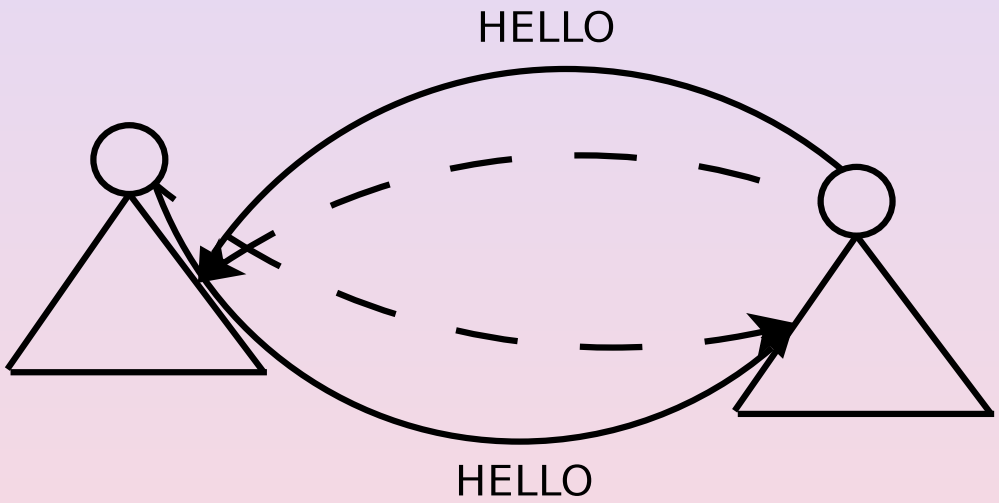
Tree reconnection



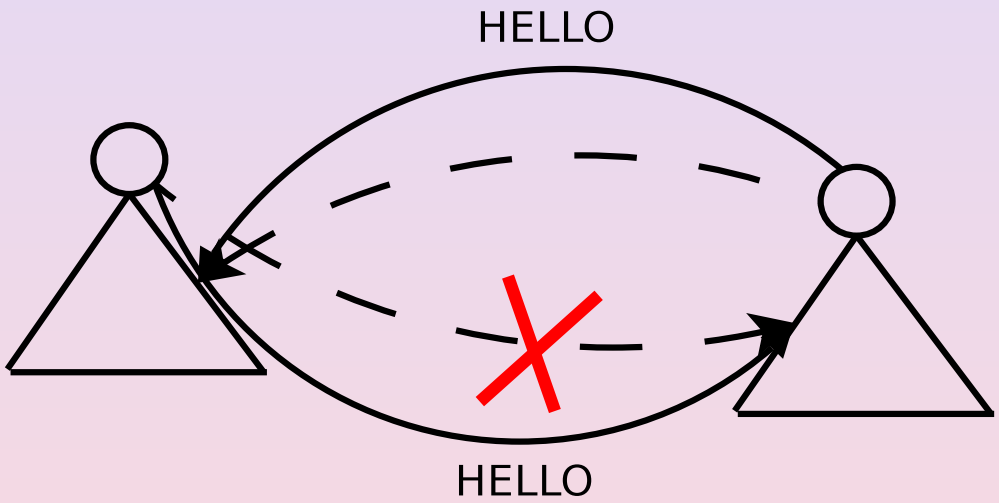
Tree reconnection



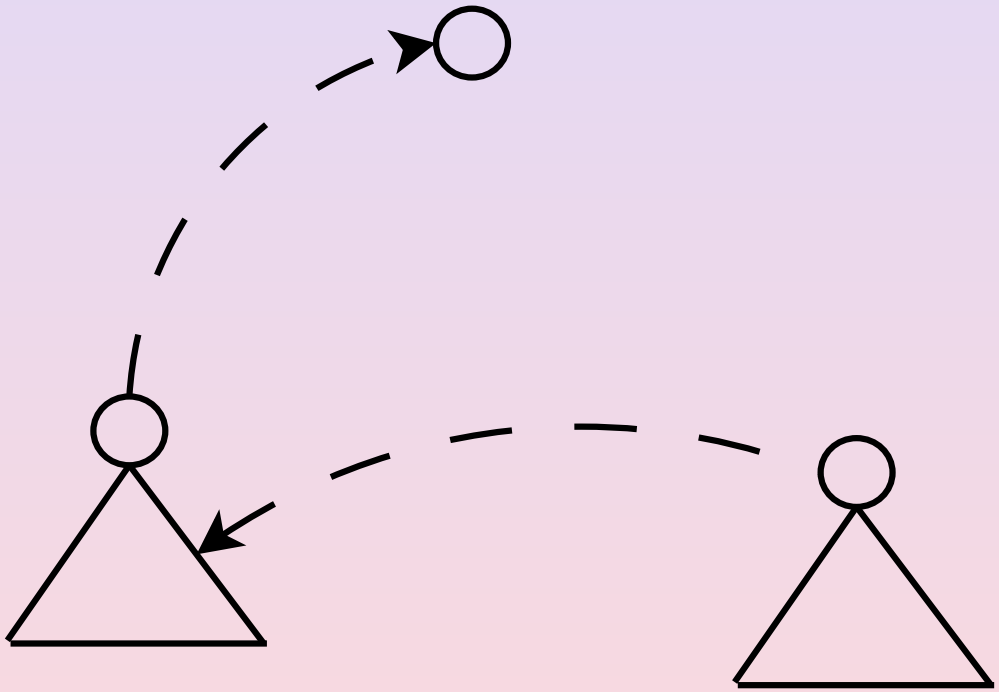
Tree reconnection



Tree reconnection



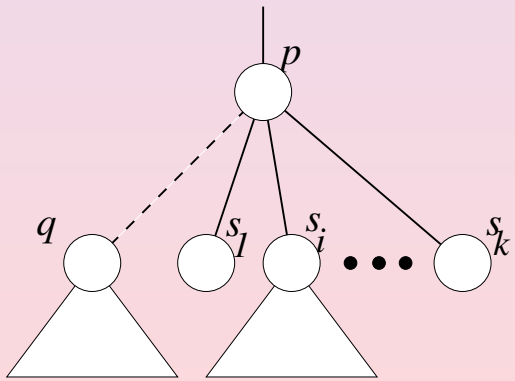
Tree reconnection



Routing the false sons (1/2)

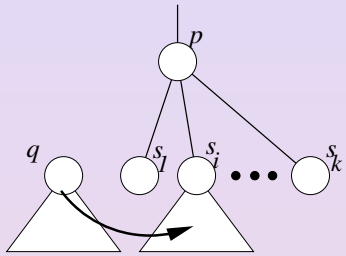
Each node p having a false son q initiates the routing of q
Two cases :

- $p = q$, `MERGE` between q subtree and p subtree
- No relation of prefixing between p and q
  - p moves q to its father, if exists
  - Creation of a common father otherwise
- $q = prefix(p)$, p moves q to its father
- $p = prefix(q)$, four cases.

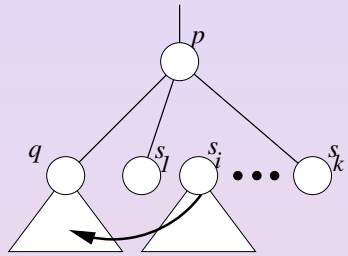


(i) $p.val = prefix(q)$ and $p.val = GCP(s_1, \ldots, s_k)$.

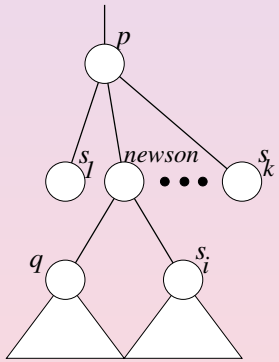
Routing of the false sons (2/2)



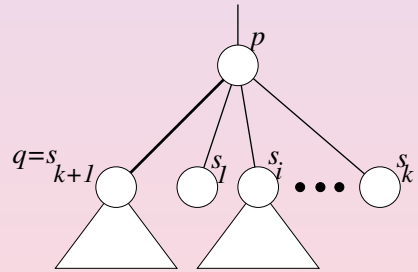
(*a*) There exists s_i such that $s_i.val = prefix(q.val)$



(*b*) There exists s_i such that $q.val = prefix(s_i.val)$

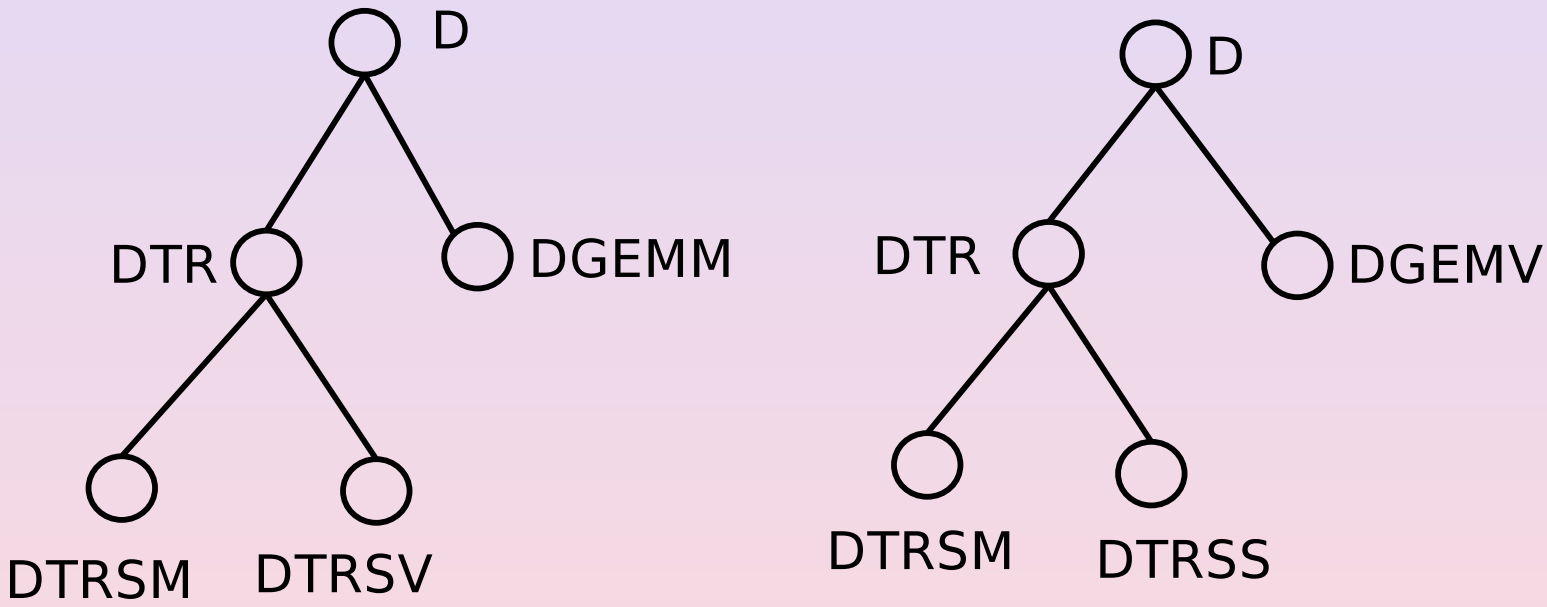


(*c*) There exists s_i such that $GCP(q.val, s_i.val) > p.val$

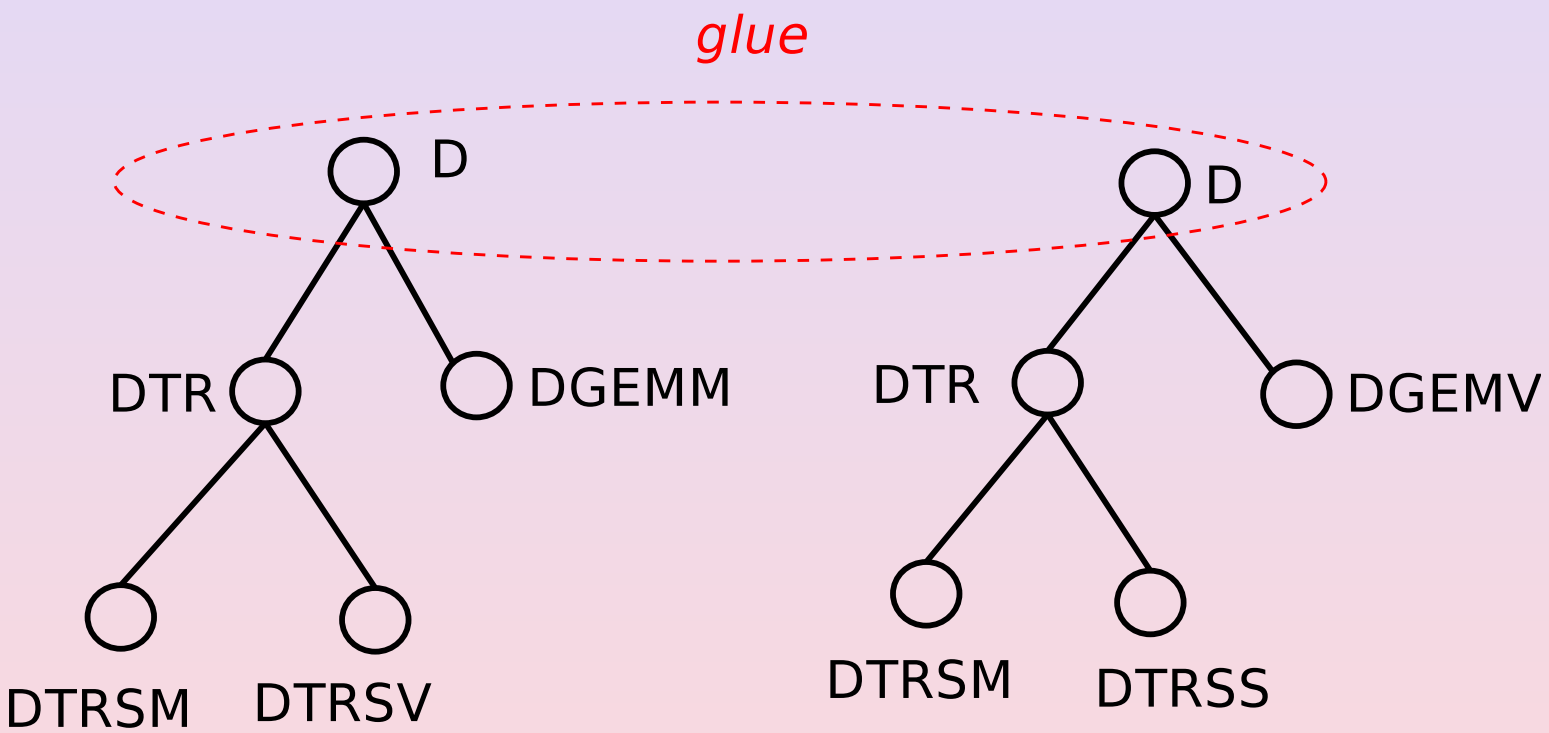


(*d*) $p.val = prefix(q.val)$

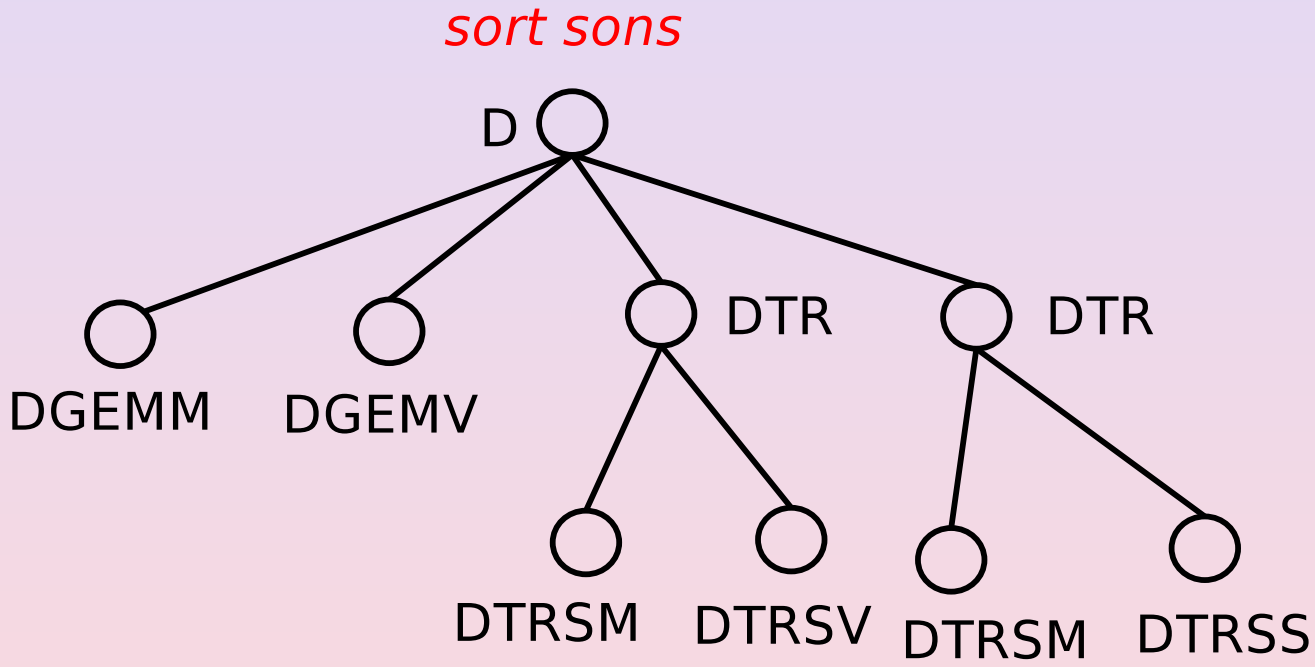
MERGE of two GCP trees



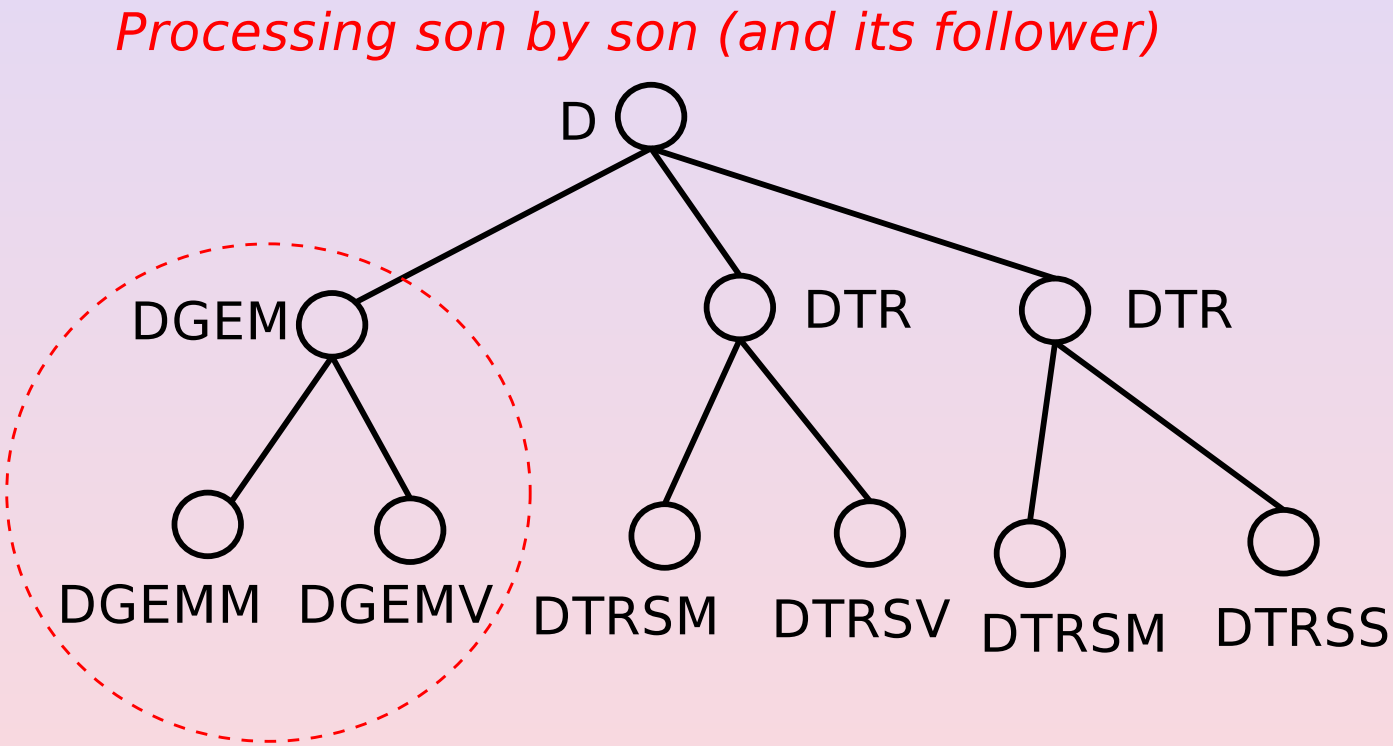
MERGE of two GCP trees



MERGE of two GCP trees

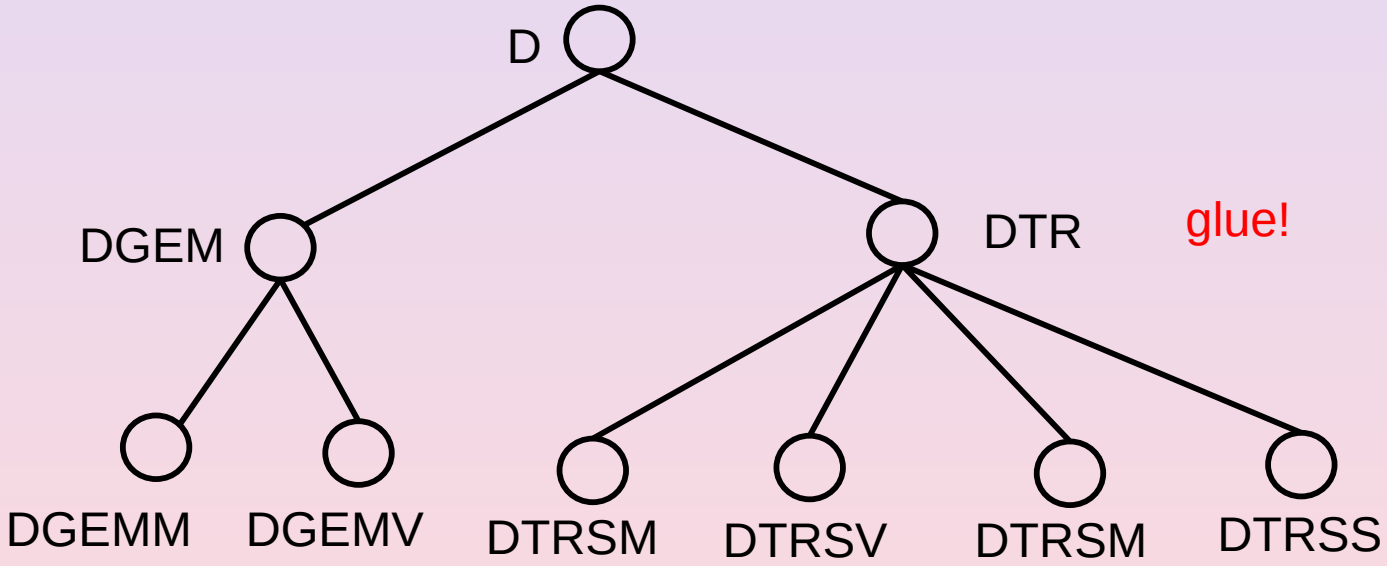


MERGE of two GCP trees



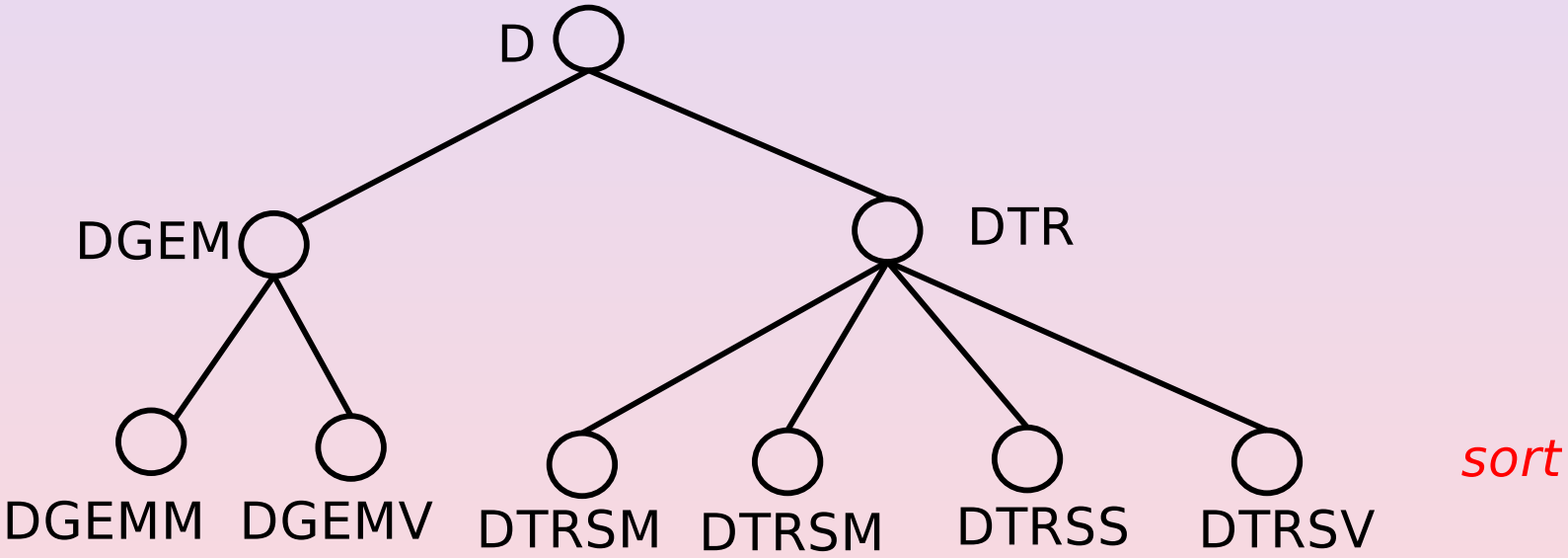
MERGE of two GCP trees

Processing son by son (and its follower)
and launching MERGE recursively



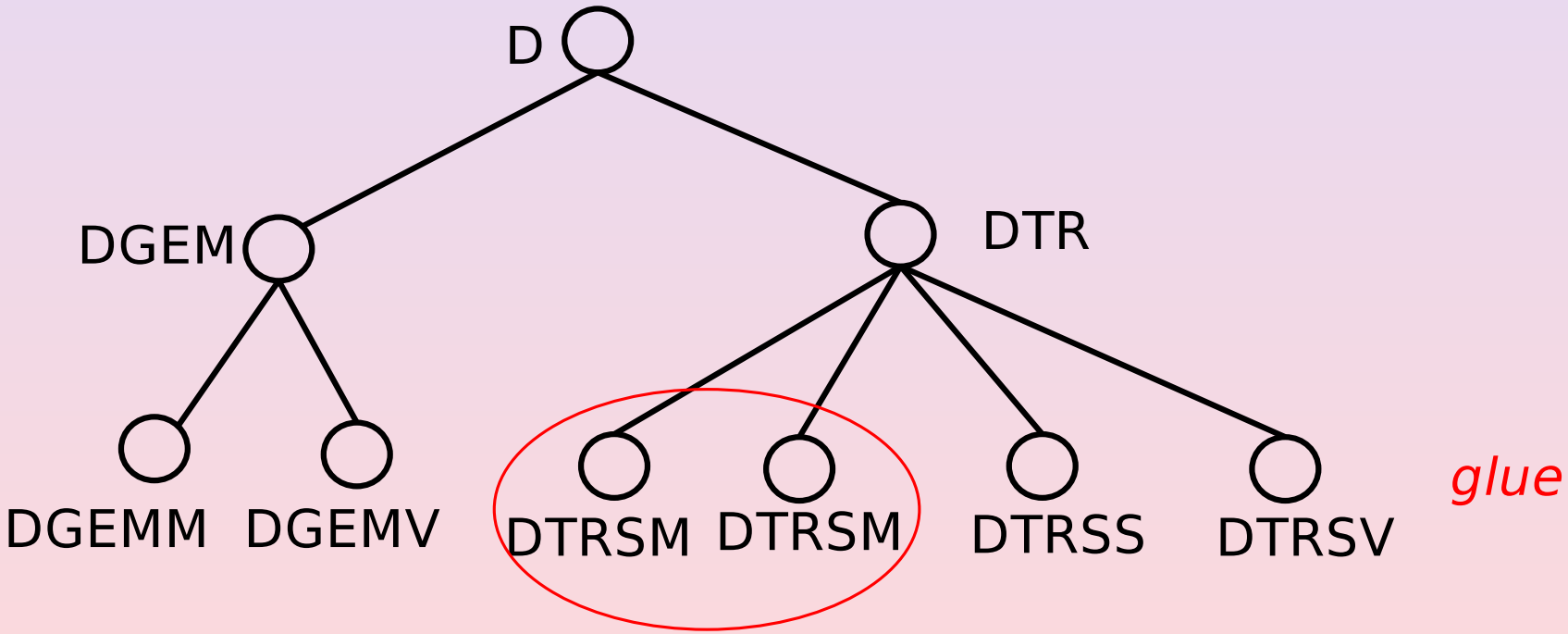
MERGE of two GCP trees

Processing son by son (and its follower) and launching MERGE recursively

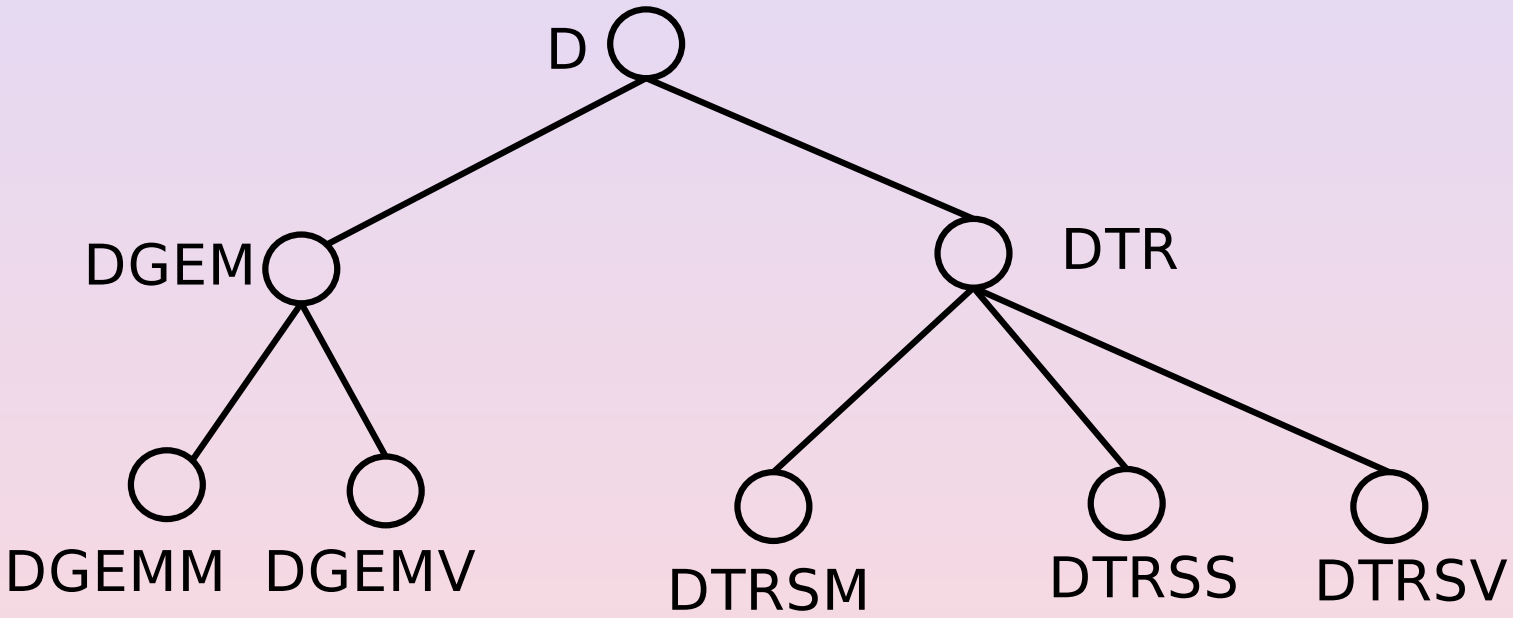


MERGE of two GCP trees

Processing son by son (and its follower) and launching MERGE recursively



MERGE of two GCP trees



Outline

Conclusion

- Repair protocol facing node crashes in a GCP Tree
  - Reconnection and reorganization of subtrees
  - GCP tree recovery after a finite time (proofs in the paper)
  - Avoid/reduce the cost of a replication strategy
- Future Work
  - Connecting replication and repair mechanisms to minimize the cost of fault-tolerance
  - Develop and validate a prototype on the Grid'5000 platform