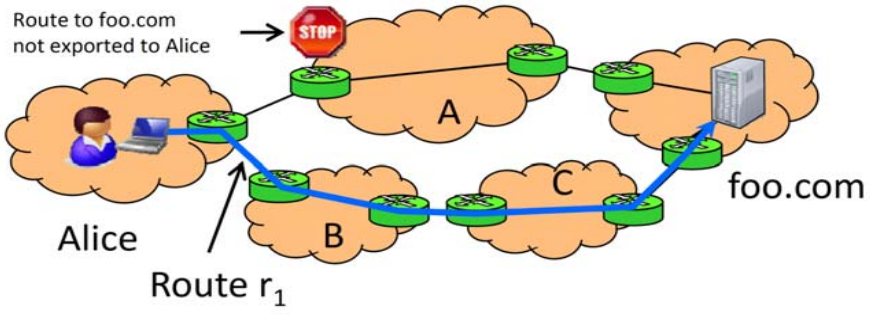
Time-aware Provenance for Distributed Systems

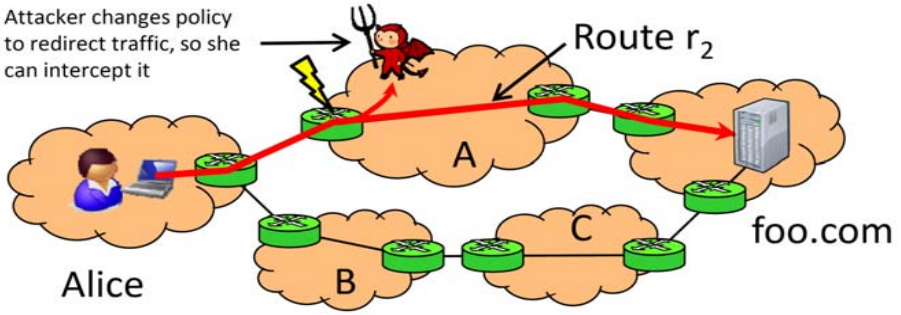
Wenchao Zhou, Ling Ding, Andreas Haeberlen, Zachary Ives, Boon Thau Loo

University of Pennsylvania

Provenance for Distributed Systems

Goal: Develop capability to answer diagnostic questions

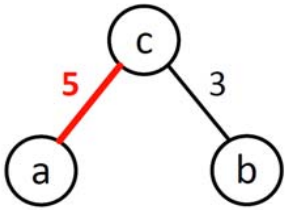




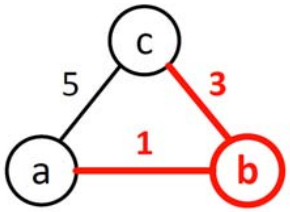
We need to tackle additional challenges…

- Provenance in transient and inconsistent state
- Explanation for state changes
- Security without trusted nodes
  - Nodes may be compromised by the attacker

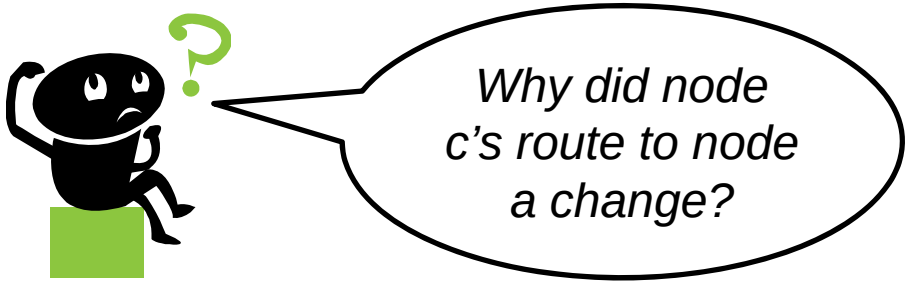
Provenance in Dynamic Environments



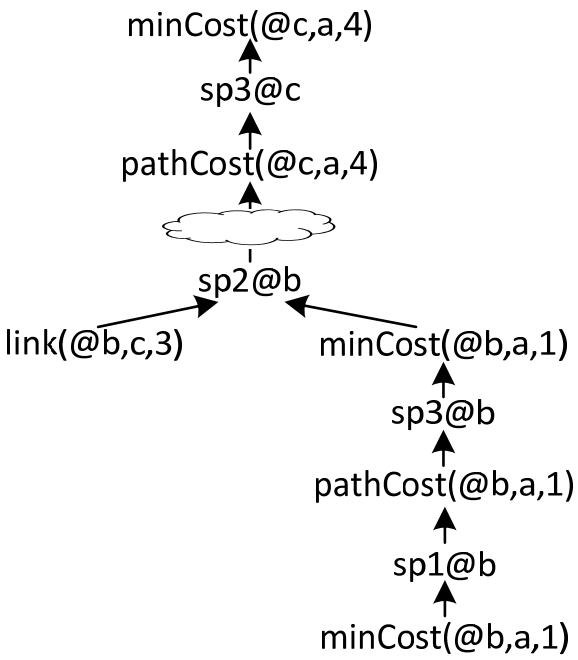
at time t1



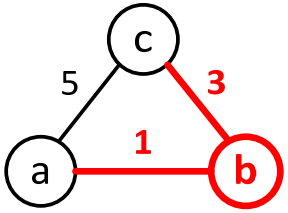
at time t2 > t1



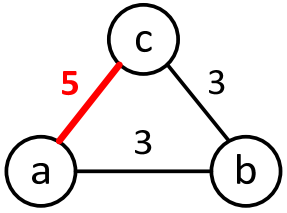
- **Reason - insertion of link(a,b,1)**
- **Provenance for system state**
  - Not track dependency between changes
  - Possible solution: differencing the current provenance with a previous version.
  - *But,* what about a deletion? No current version to compare…



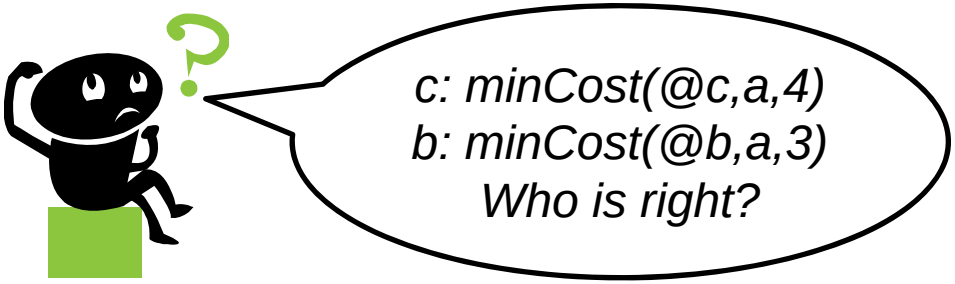
Provenance in Dynamic Environments



at time t2 > t1



at time t3 > t2



- **Explicitly capture time**
  - Handle question asked when the system is in transient state
  - Consistent view of the provenance graph

Time-aware Provenance

- **Explicitly capture causalities between state changes**
  - Explain the INSERT / DELETE of tuples
  - Event-based execution triggered by state changes

sp2: pathCost(@Z,D,C1+C2) :- link(@S,Z,C1), minCost(@S,D,C2).

sp2a: ΔpathCost(@Z,D,C1+C2) :- link(@S,Z,C1), ΔminCost(@S,D,C2).

sp2b: ΔpathCost(@Z,D,C1+C2) :- Δlink(@S,Z,C1), minCost(@S,D,C2).

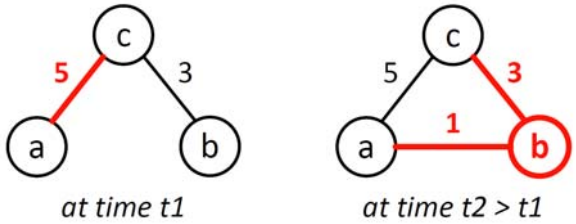
Time-aware Provenance

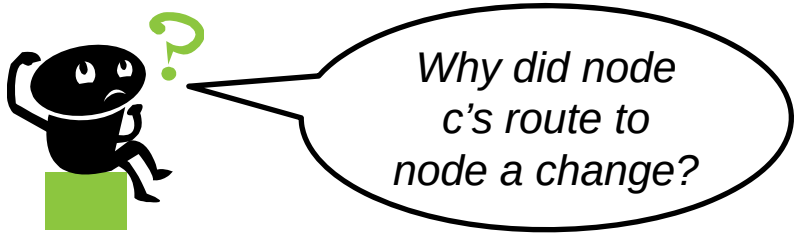
- **Explicitly capture causalities between state changes**
  - Explain the INSERT / DELETE of tuples
  - Event-based execution triggered by state changes
  - Update due to constraints (primary keys, aggregation)

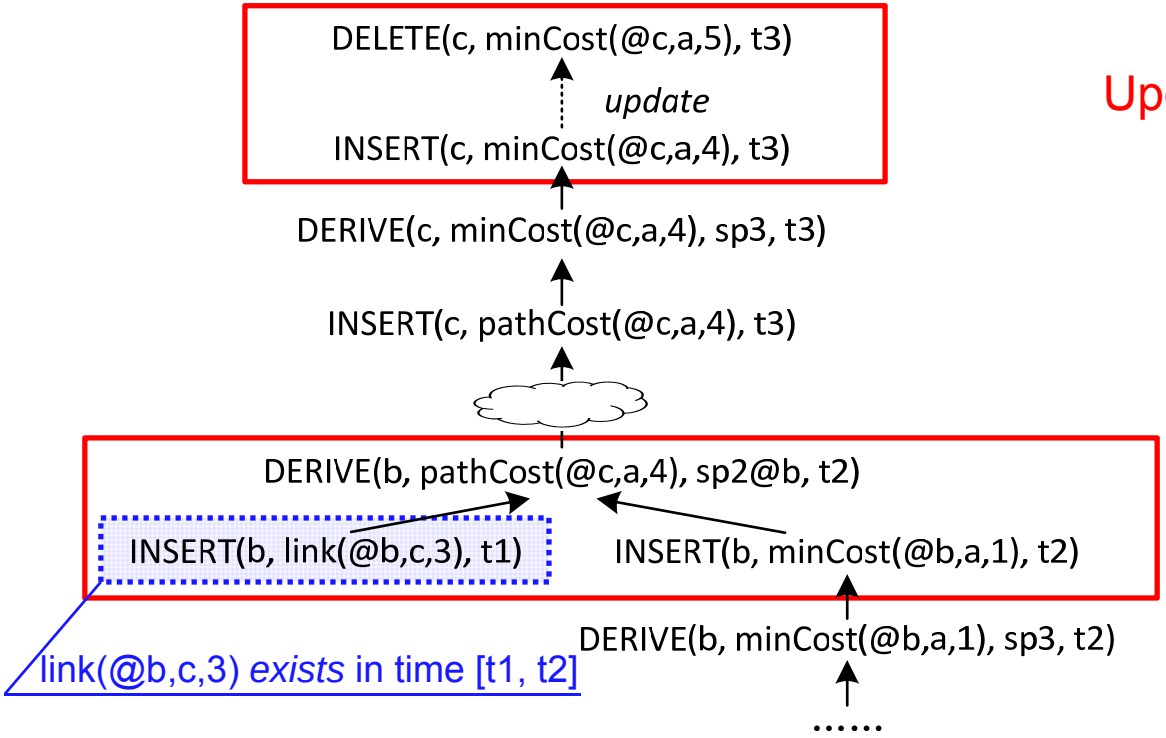
sp3: minCost(@S,D,MIN<C>) :- pathCost(@S,D,C).

insertion of minCost(@c,a,4) caused deletion of minCost(@c,a,5)

TAP Provenance Model





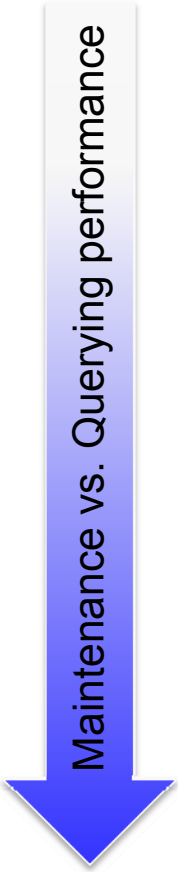


Update due to constraints

Rule triggering

Provenance Maintenance

- **Provenance with temporal dimension**
  - Versions of provenance
  - Expensive – provenance explosion
- **Active maintenance**
  - Provenance deltas – deltas between adjacent versions
  - Incrementally applied in querying
- **Reactive maintenance**
  - Input logs – communications and update of base tuples
  - Reconstruct provenance by deterministic replay
  - Long-running systems? Periodic snapshots



Secure Provenance Querying

- **Byzantine adversaries**
  - May have compromised an arbitrary subset of the nodes
  - May have complete control over the nodes – arbitrary behavior
- **Guarantees**
  - Idealism: Always get correct forensics results (not possible!)
  - Practicality: The conservative model requires compromises
    - May be incomplete, but, it will identify at least one faulty node

Thank You …