Recursive Computation of Regions and Connectivity in Networks

Mengmeng Liu, Nicholas E. Taylor, Wenchao Zhou, Zachary G. Ives, Boon Thau Loo
Computer and Information Science Department, University of Pennsylvania

Motivation

• Declarative queries are revisited in both declarative networking and sensor networks scenarios.
• How to maintain a view over dynamic network state where the view is frequently distributed, recursive and may contain aggregations?
• Problem: Incremental Recursive View Maintenance in the presence of distributed streams of update tuples (insertions and deletions).

Example Queries

Q1 (Network reachability query)

(r1) reachable(x,y) :- link(x,y).
(r2) reachable(x,y) :- link(x,z), reachable(z,y).

It finds all pairs of nodes that can reach each other.

Q2 (Sensing contiguous regions)

(r1) activeRegion(rid,x) :- sensor(x,posx), isTriggered(x), mainSensorInRegion(rid,x).
(r2) activeRegion(rid,y) :- sensor(x,posx), sensor(y,posy), isTriggered(x), activeRegion(rid,x), distance(posx,posy) < k.

It finds contiguous (within k meters) triggered nodes and adds them to regions.

Existing Solutions

• DReD [Gupta’93]: recursive view maintenance using delta rules
  - Insertions: incremental maintenance
  - Deletions: over-delete and re-derive
• Drawbacks:
  - centralised mechanism, not scalable to hundreds of nodes
  - bad performance, especially when deletions are frequent
  - worse than naive method in some cases
• What we want:
  - efficiency
  - scalability
  - low communication overhead
  - generality

Our Approach: Absorption Provenance

We annotate tuple with Boolean expressions such that the tuple is in the view iff the Boolean expression evaluates to true.

Insertions: record all the derivations of each tuple in an absorbed Boolean expression.

Deletions: zero-out the provenance tokens in each Boolean expression.

• Rules:
  \( \sigma[\text{link}] (\text{tuple}) \) : For each output tuple 
  \( \text{link}(t, \text{tuple}) \) : For each input tuple
  \( \text{link}(\text{tuple}, \text{sensor}) \) : For each sensor
  \( \text{sensor}(x) \) : For each sensor location
  \( \text{isTriggered}(x) \) : For each triggered sensor
  \( \text{mainSensorInRegion}(\text{rid}, x) \) : For each main sensor

Distributed Optimization Techniques

• Fixpoint
  We reach a fixpoint when we can no longer derive any new results that affect the Absorption Provenance of any tuple!

• MinShip
  We only propagate the first derivation, and buffer the subsequent derivation. If the buffer size is large enough, this is called Lazy Propagation.

• Aggregate Selection
  We always push down selections before aggregations, and design a provenance-aware module to handle insertions as well as deletions.

Performance Evaluation

• Incremental Maintenance

• Scaling data sets

• Aggregate Selection

Publications