Efficient Querying & Maintenance of Network Provenance at Internet-Scale

***Wenchao Zhou*[*], *Micah Sherr*[*], *Tao Tao*[*], *Xiaozhou Li*[*],**

***Boon Thau Loo*[*], *and Yun Mao*[+]**

[*]*University of Pennsylvania* [+]*AT&T Labs Research*

Introduction

- **Developing correct large-scale distributed systems is hard**
  - Design flaws and internal software bugs
  - Misconfiguration
  - Malicious attacks and intrusion
- **Goal: capability to discover the history of network state**
  - *Network diagnosis & forensics*: IP-Traceback [SDK+ SIGCOMM00], Pip [RKW+ NSDI06], X-Trace [FPK+ NSDI07]
  - *Network accountability*: AIP [ABF+ SIGCOMM08]
  - *Traffic classification & access control*: Pedigree [RMT+ SIGCOMM09(demo)]
  - **Problem**: existing solutions narrowly target specific challenges/applications

Challenges

- **A *generic* solution: *network provenance (or lineage)***
  - Explains the existence and derivation of any network state
  - Maps naturally into various applications
- **New challenges at Internet-scale**
  - Distributed computation across tens of thousands of nodes
  - Concurrent execution with existing network protocols
    - Minimal impact on the running protocols
    - New cost model based on bandwidth consumption and delay of convergence time

ExSPAN

- **EXtenSible Provenance-Aware Networks**
  - Data model: network provenance as distributed relational tables
  - Efficient provenance maintenance using *declarative networking*
  - Customizable provenance querying with optimizations to reduce bandwidth consumption
  - Prototype implementation and evaluation

Outline

- Introduction
- Taxonomy of Network Provenance
- Design of ExSPAN
- Implementation and Evaluation Results
- Conclusion and Future Work

Taxonomy of Network Provenance

- **Representation**
  - Graphs: dependency between tuples
  - Algebraic representation: e.g., sets, polynomials [GKT PODS07]
- **Distribution**
  - Centralized: provenance on a centralized server
  - Distributed value-based: entire provenance information with each tuple
  - Distributed ref-based: only pointers to direct contributing tuples

Systems	Representation	Distribution
IP-Traceback	Graphs	Dist. Ref-based
Pip	Graphs	Centralized
X-Trace	Graphs	Dist. Ref-based
AIP	Algebraic	Dist. Value-based
Pedigree	Algebraic	Dist. Value-based

Outline

- Introduction
- Taxonomy of Network Provenance
- **Design of ExSPAN**
  - Background: Declarative Networking
  - Data Model and Distributed Storage
  - Provenance Maintenance
  - Provenance Querying and Optimizations
- **Implementation and Evaluation Results**
- **Conclusion and Future Work**

Declarative Networking [LCG+ SIGMOD06]

- **Compact specification of network protocols**
- ***Network Datalog (NDlog)***
  - A distributed variant of *Datalog*
  - Continuous recursive queries over network state
- **Distributed Execution**
  - Compiled into distributed dataflows
  - Executed by a distributed query engine
  - Distribution: *location specifiers* – data placement of tuple

Example: Pairwise Minimal Cost

sp1: pathCost(@S,D,C) :- link(@S,D,C).

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

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

link(@Src,Dst,C) – "a direct link from node *Src* to *Dst* with cost *C*"

Example: Pairwise Minimal Cost

```
sp1: pathCost(@S,D,C) :- link(@S,D,C).

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

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

link(@Src,Dst,C) – "a direct link from node *Src* to *Dst* with cost *C*"

pathCost(@Src,Dst,C) – "a path from node *Src* to *Dst* with cost *C*"

Example: Pairwise Minimal Cost

sp1: pathCost(@S,D,C) :- link(@S,D,C). ← *One-hop paths*

sp2: pathCost(@Z,D,C1+C2) :- link(@S,Z,C1),
minCost(@S,D,C2). ← *Multi-hop paths*

sp3: minCost(@S,D,MIN<C>) :- pathCost(@S,D,C). ← *Aggregation for min cost*

link(@Src,Dst,C) – “a direct link from node *Src* to *Dst* with cost *C*”

pathCost(@Src,Dst,C) – “a path from node *Src* to *Dst* with cost *C*”

minCost(@Src,Dst,C) – “best path from node *Src* to *Dst* with minimal cost *C*”

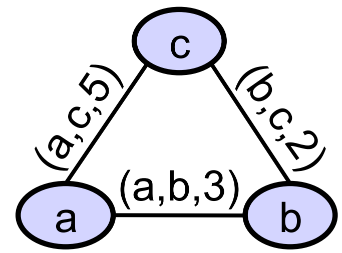
Data Model of Network Provenance

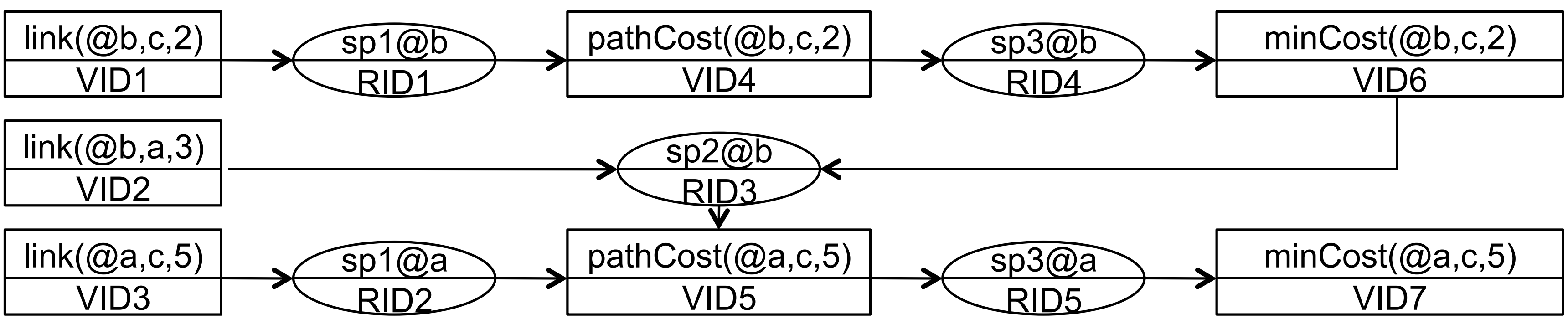
- **Data model – a directed graph**
  - *Tuple vertices* (rectangle) and *rule execution (oval) vertices*
  - Edges represent dataflows between tuple and rule execution vertices

sp1: pathCost(@S,D,C) :- link(@S,D,C).

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

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





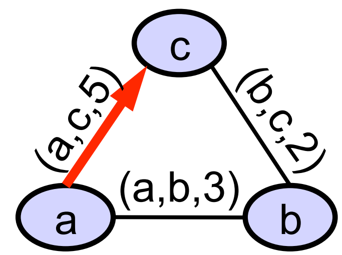
Data Model of Network Provenance

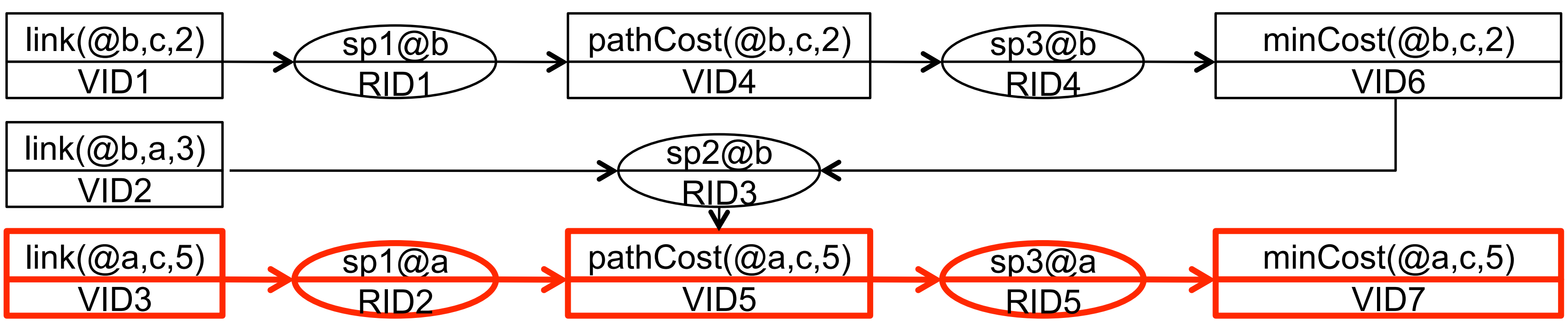
- **Data model – a directed graph**
  - *Tuple vertices* (rectangle) and *rule execution (oval) vertices*
  - Edges represent dataflows between tuple and rule execution vertices

sp1: pathCost(@S,D,C) :- link(@S,D,C).

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

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





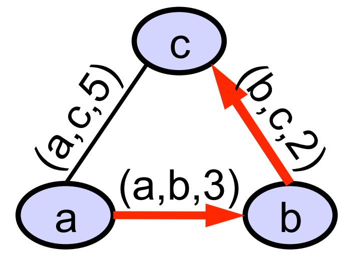
Data Model of Network Provenance

- **Data model – a directed graph**
  - *Tuple vertices* (rectangle) and *rule execution (oval) vertices*
  - Edges represent dataflows between tuple and rule execution vertices

sp1: pathCost(@S,D,C) :- link(@S,D,C).

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

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



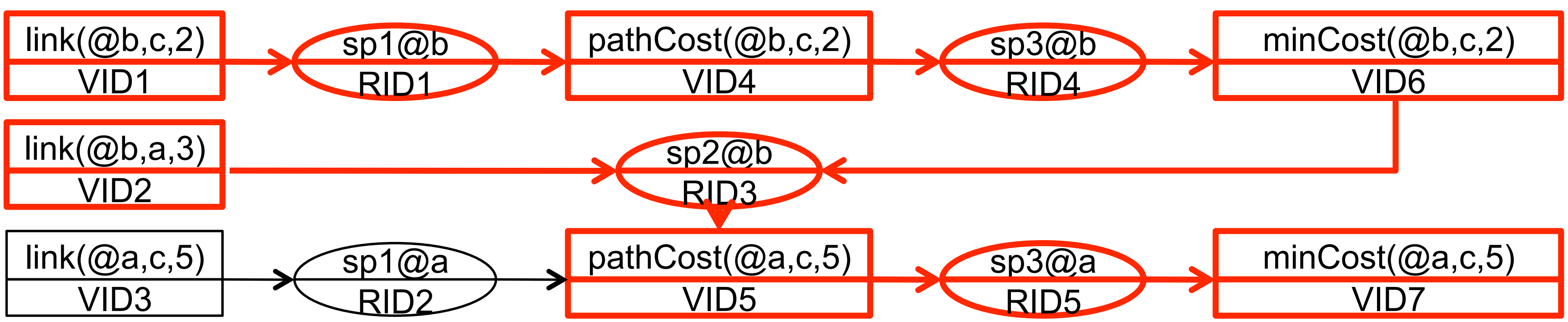
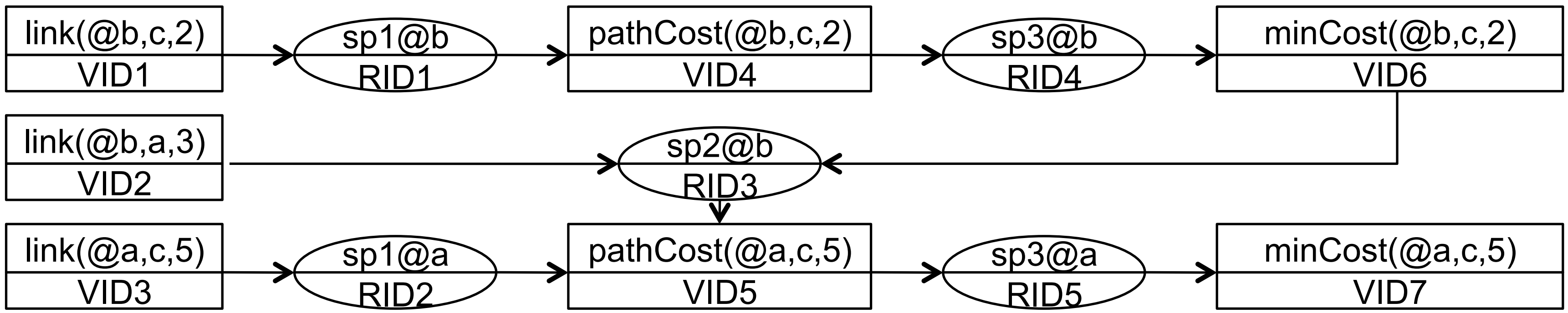


Table-based Distributed Storage



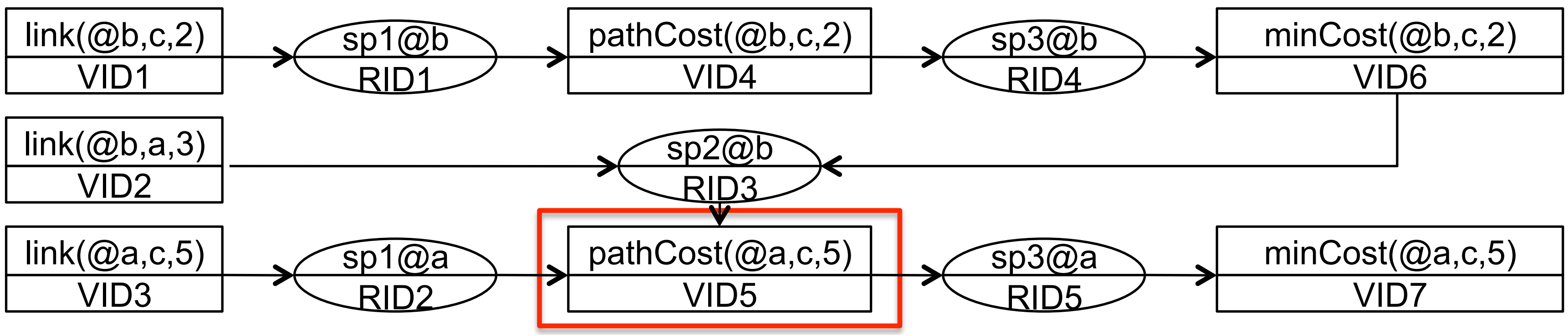
prov – direct derivation of a tuple

Loc	VID	RID	RLoc
a	VID5	RID2	a
a	VID5	RID3	b
a	VID7	RID5	a
b	VID4	RID1	b
b	VID6	RID4	b

ruleExec – metadata of a rule execution

RLoc	RID	R	VIDList
a	RID2	sp1	(VID3)
a	RID5	sp3	(VID5)
b	RID1	sp1	(VID1)
b	RID3	sp2	(VID2,VID7)
b	RID4	sp3	(VID4)

Table-based Distributed Storage



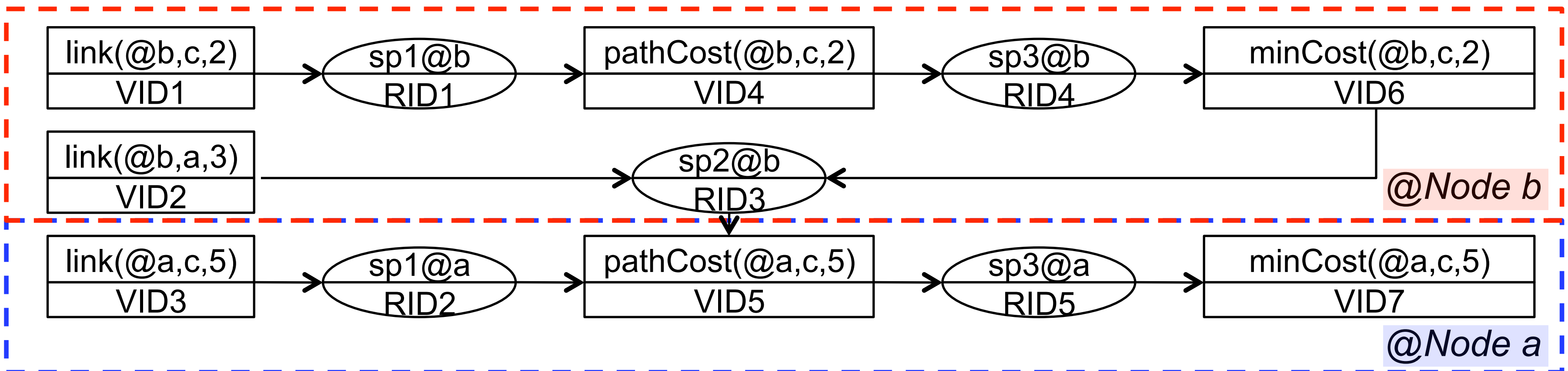
prov – direct derivation of a tuple

Loc	VID	RID	RLoc
a	**VID5**	**RID2**	**a**
a	**VID5**	**RID3**	**b**
a	VID7	RID5	a
b	VID4	RID1	b
b	VID6	RID4	b

ruleExec – metadata of a rule execution

RLoc	RID	R	VIDList
a	RID2	sp1	(VID3)
a	RID5	sp3	(VID5)
b	RID1	sp1	(VID1)
b	RID3	sp2	(VID2,VID7)
b	RID4	sp3	(VID4)

Table-based Distributed Storage



prov – direct derivation of a tuple

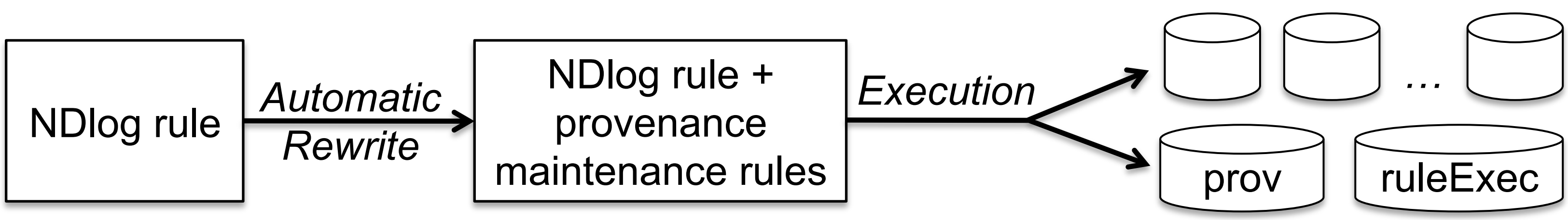
Loc	VID	RID	RLoc
a	VID5	RID2	a
a	VID5	RID3	b
a	VID7	RID5	a
b	VID4	RID1	b
b	VID6	RID4	b

ruleExec – metadata of a rule execution

RLoc	RID	R	VIDList
a	RID2	sp1	(VID3)
a	RID5	sp3	(VID5)
b	RID1	sp1	(VID1)
b	RID3	sp2	(VID2,VID7)
b	RID4	sp3	(VID4)

Provenance Maintenance

- **Provenance defined as views of network state**
  - Leverage the incremental view maintenance in declarative networking
  - Pipelined Semi-Naïve (PSN) [LCG+ SIGMOD06] evaluation
- **Automatic rule rewrite (more details in paper)**
  - Any NDlog rule is rewritten into an equivalent set of NDlog rules
  - Generate the same rule evaluation results
  - Additionally maintain information in prov and ruleExec tables



Provenance Querying

- **Generic framework for provenance querying**
  - Distributed queries on prov and ruleExec tables
  - Traversal of provenance graphs
    - Recursively query the tuples contributing to the current derivation
    - Buffer and combine sub-results
    - Return query results back along the reverse path

prov – direct derivation of a tuple

Loc	VID	RID	RLoc	Derivation
a	VID5	RID2	a	pathCost(@a,c,5)
a	VID5	RID3	b	pathCost(@a,c,5)
a	VID7	RID5	a	minCost(@a,c,5)
b	VID4	RID1	b	pathCost(@b,c,2)
b	VID6	RID4	b	minCost(@b,c,2)

ruleExec – metadata of a rule execution

RLoc	RID	R	VIDList	Derivation
a	RID2	sp1	(VID3)	pathCost(@a,c,5)
a	RID5	sp3	(VID5)	minCost(@a,c,5)
b	RID1	sp1	(VID1)	pathCost(@a,c,5)
b	RID3	sp2	(VID2,VID7)	pathCost(@b,c,2)
b	RID4	sp3	(VID4)	minCost(@b,c,2)

Provenance Querying

- **Generic framework for provenance querying**
  - Distributed queries on prov and ruleExec tables
  - Traversal of provenance graphs
    - Recursively query the tuples contributing to the current derivation
    - Buffer and combine sub-results
    - Return query results back along the reverse path

prov – direct derivation of a tuple

Loc	VID	RID	RLoc	Derivation
a	VID5	RID2	a	pathCost(@a,c,5)
a	VID5	RID3	b	pathCost(@a,c,5)
a	VID7	RID5	a	minCost(@a,c,5)
b	VID4	RID1	b	pathCost(@b,c,2)
b	VID6	RID4	b	minCost(@b,c,2)

ruleExec – metadata of a rule execution

RLoc	RID	R	VIDList	Derivation
a	RID2	sp1	(VID3)	pathCost(@a,c,5)
a	RID5	sp3	(VID5)	minCost(@a,c,5)
b	RID1	sp1	(VID1)	pathCost(@a,c,5)
b	RID3	sp2	(VID2,VID7)	pathCost(@b,c,2)
b	RID4	sp3	(VID4)	minCost(@b,c,2)

Provenance Querying

- **Generic framework for provenance querying**
  - Distributed queries on prov and ruleExec tables
  - Traversal of provenance graphs
    - Recursively query the tuples contributing to the current derivation
    - Buffer and combine sub-results
    - Return query results back along the reverse path

prov – direct derivation of a tuple

Loc	VID	RID	RLoc	Derivation
a	VID5	RID2	a	pathCost(@a,c,5)
a	VID5	RID3	b	pathCost(@a,c,5)
a	VID7	RID5	a	minCost(@a,c,5)
b	VID4	RID1	b	pathCost(@b,c,2)
b	VID6	RID4	b	minCost(@b,c,2)

ruleExec – metadata of a rule execution

RLoc	RID	R	VIDList	Derivation
a	RID2	sp1	(VID3)	pathCost(@a,c,5)
a	RID5	sp3	(VID5)	minCost(@a,c,5)
b	RID1	sp1	(VID1)	pathCost(@a,c,5)
b	RID3	sp2	(VID2,VID7)	pathCost(@b,c,2)
b	RID4	sp3	(VID4)	minCost(@b,c,2)

Provenance Querying

- **Generic framework for provenance querying**
  - Distributed queries on prov and ruleExec tables
  - Traversal of provenance graphs
    - Recursively query the tuples contributing to the current derivation
    - Buffer and combine sub-results
    - Return query results back along the reverse path

prov – direct derivation of a tuple

Loc	VID	RID	RLoc	Derivation
a	VID5	RID2	a	pathCost(@a,c,5)
a	VID5	RID3	b	pathCost(@a,c,5)
a	VID7	RID5	a	minCost(@a,c,5)
b	VID4	RID1	b	pathCost(@b,c,2)
b	VID6	RID4	b	minCost(@b,c,2)

ruleExec – metadata of a rule execution

RLoc	RID	R	VIDList	Derivation
a	RID2	sp1	(VID3)	pathCost(@a,c,5)
a	RID5	sp3	(VID5)	minCost(@a,c,5)
b	RID1	sp1	(VID1)	pathCost(@a,c,5)
b	RID3	sp2	(VID2,VID7)	pathCost(@b,c,2)
b	RID4	sp3	(VID4)	minCost(@b,c,2)

Provenance Querying

- **Generic framework for provenance querying**
  - Distributed queries on prov and ruleExec tables
  - Traversal of provenance graphs
    - Recursively query the tuples contributing to the current derivation
    - Buffer and combine sub-results
    - Return query results back along the reverse path

prov – direct derivation of a tuple

Loc	VID	RID	RLoc	Derivation
a	VID5	RID2	a	pathCost(@a,c,5)
a	VID5	RID3	b	pathCost(@a,c,5)
a	VID7	RID5	a	minCost(@a,c,5)
b	VID4	RID1	b	pathCost(@b,c,2)
b	VID6	RID4	b	minCost(@b,c,2)

ruleExec – metadata of a rule execution

RLoc	RID	R	VIDList	Derivation
a	RID2	sp1	(VID3)	pathCost(@a,c,5)
a	RID5	sp3	(VID5)	minCost(@a,c,5)
b	RID1	sp1	(VID1)	pathCost(@a,c,5)
b	RID3	sp2	(VID2,VID7)	pathCost(@b,c,2)
b	RID4	sp3	(VID4)	minCost(@b,c,2)

Provenance Querying

- **Generic framework for provenance querying**
  - Distributed queries on prov and ruleExec tables
  - Traversal of provenance graphs
    - Recursively query the tuples contributing to the current derivation
    - Buffer and combine sub-results
    - Return query results back along the reverse path
  - **Queries specified as compact NDlog rules**

prov – direct derivation of a tuple

Loc	VID	RID	RLoc	Derivation
a	VID5	RID2	a	pathCost(@a,c,5)
a	VID5	RID3	b	pathCost(@a,c,5)
a	VID7	RID5	a	minCost(@a,c,5)
b	VID4	RID1	b	pathCost(@b,c,2)
b	VID6	RID4	b	minCost(@b,c,2)

ruleExec – metadata of a rule execution

RLoc	RID	R	VIDList	Derivation
a	RID2	sp1	(VID3)	pathCost(@a,c,5)
a	RID5	sp3	(VID5)	minCost(@a,c,5)
b	RID1	sp1	(VID1)	pathCost(@a,c,5)
b	RID3	sp2	(VID2,VID7)	pathCost(@b,c,2)
b	RID4	sp3	(VID4)	minCost(@b,c,2)

Customizable Provenance Querying

- **Easy customization for various applications**
  - Annotate base tuples: f_pEDB()
  - Define how sub-results should be combined: f_pIDB(), f_pRule()
  - Define a *provenance semi-ring* [GKT+ PODS07]

Customizable Provenance Querying

- **Easy customization for various applications**
  - Annotate base tuples: f_pEDB()
  - Define how sub-results should be combined: f_pIDB(), f_pRule()
  - Define a *provenance semi-ring* [GKT+ PODS07]

	f_pEDB()	f_pIDB()	f_pRule()	Applications
Node Set	{NodeID}	$\bigcup_{i=1}^{n} D_i$	$\bigcup_{i=1}^{n} P_i$	Traffic Classification / Root Cause Analysis
# of Derivations	1	$\sum_{i=1}^{n} D_i$	$\prod_{i=1}^{n} P_i$	Trust Management
Derivability Test	True	$\vee_{i=1}^{n} D_i$	$\wedge_{i=1}^{n} P_i$	Incremental View Maintenance
Prov. Polynomial	VID	$(D_1 \cdots D_n)@Loc$	$\langle R@RLoc \rangle (P_1 \cdots P_n)$	Graph -> Algebraic Representation

Query Optimizations

- **Query results caching**
  - Low bandwidth consumption in maintenance
  - Queries are frequent: subsequent queries benefit from caches
- **Alternative query traversal orders**
  - Breadth First Search (BFS) vs Depth First Search (DFS)
  - Tradeoff between query latency and bandwidth consumption
- **Compression with Binary Decision Diagram (BDD)**
  - Retain sufficient information for specific applications [LTZ+ ICDE09]

Outline

- Introduction
- Taxonomy of Network Provenance
- Design of ExSPAN
- **Implementation and Evaluation Results**
- **Conclusion and Future Work**

Implementation and Evaluation Results

- **Based on the RapidNet declarative networking engine** *http://netdb.cis.upenn.edu/rapidnet*
  - Simulation based on the ns-3 network simulator
  - Actual implementation
- **Performance of provenance maintenance**
  - Simulated network up to 500 nodes generated by GT-ITM
  - Protocols: MinCost / Path-Vector / Packet-Forwarding
  - Workload: fixpoint computation from scratch / incremental maintenance
  - **Results: negligible overhead for provenance maintenance**
    - Less than 6% increase in communication overhead
    - Does not affect the scalability of the network protocols

Implementation and Evaluation Results

- **Based on the RapidNet declarative networking engine** *http://netdb.cis.upenn.edu/rapidnet*
  - Simulation based on the ns-3 network simulator
  - Actual implementation
- **Performance of provenance maintenance**
- **Performance of provenance querying and optimizations**
  - Workload: queries for complete provenance / # of alternative derivations
  - **Results: reasonable querying overhead**
    - Complete provenance: <10KB aggregate communication per query
    - Optimizations effectively reduce the communication overhead

Implementation and Evaluation Results

- **Based on the RapidNet declarative networking engine** *http://netdb.cis.upenn.edu/rapidnet*
  - Simulation based on the ns-3 network simulator
  - Actual implementation
- **Performance of provenance maintenance**
- **Performance of provenance querying and optimizations**
- **Evaluation of actual implementation**
  - On a local cluster with 48 running instances
  - Similar observation and conclusion as simulation

Outline

- Introduction
- Taxonomy of Network Provenance
- Design of ExSPAN
- Implementation and Evaluation Results
- **Conclusion and Future Work**

Conclusion and Future Work

- **A generic and extensible framework for network provenance**
  - Provenance modeled as distributed relational tables
  - Efficient and customizable provenance maintenance and querying
  - Prototype implementation based on the RapidNet engine
- **Future work**
  - Network provenance with security guarantees
  - Integration with legacy applications

Thank You …

Project website: *http://netdb.cis.upenn.edu/ds2.*

Efficient querying and maintenance of network provenance at Internet-scale,

Wenchao Zhou, Micah Sherr, Tao Tao, Xiaozhou Li, Boon Thau Loo, and Yun Mao