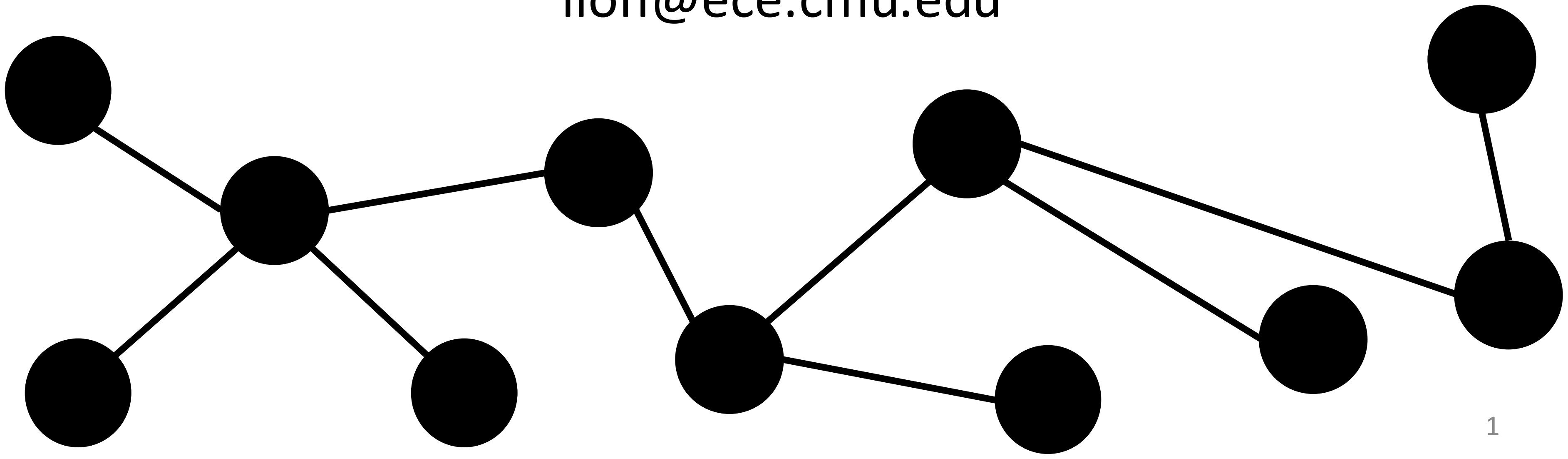
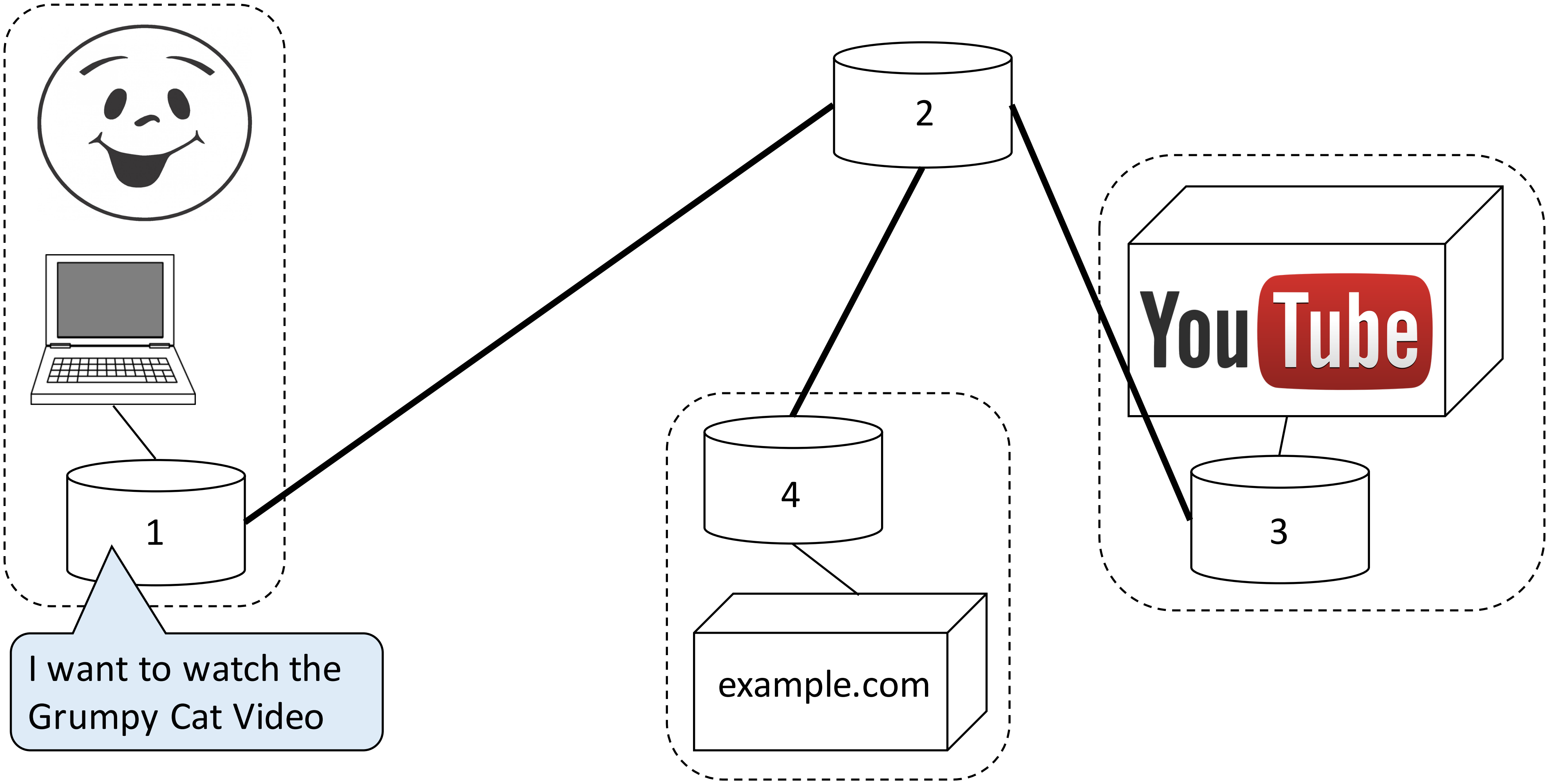
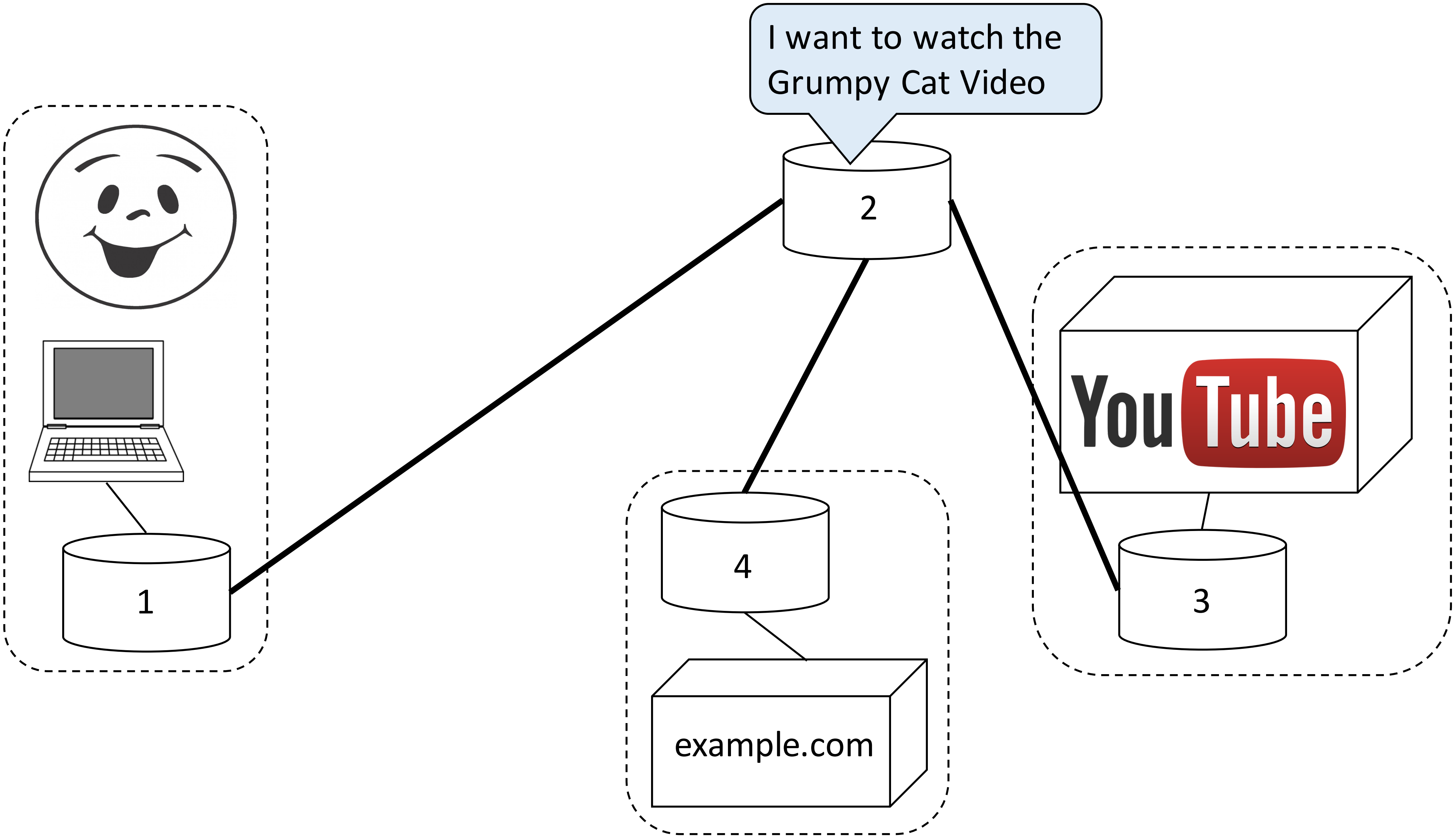
Distributed Provenance Compression

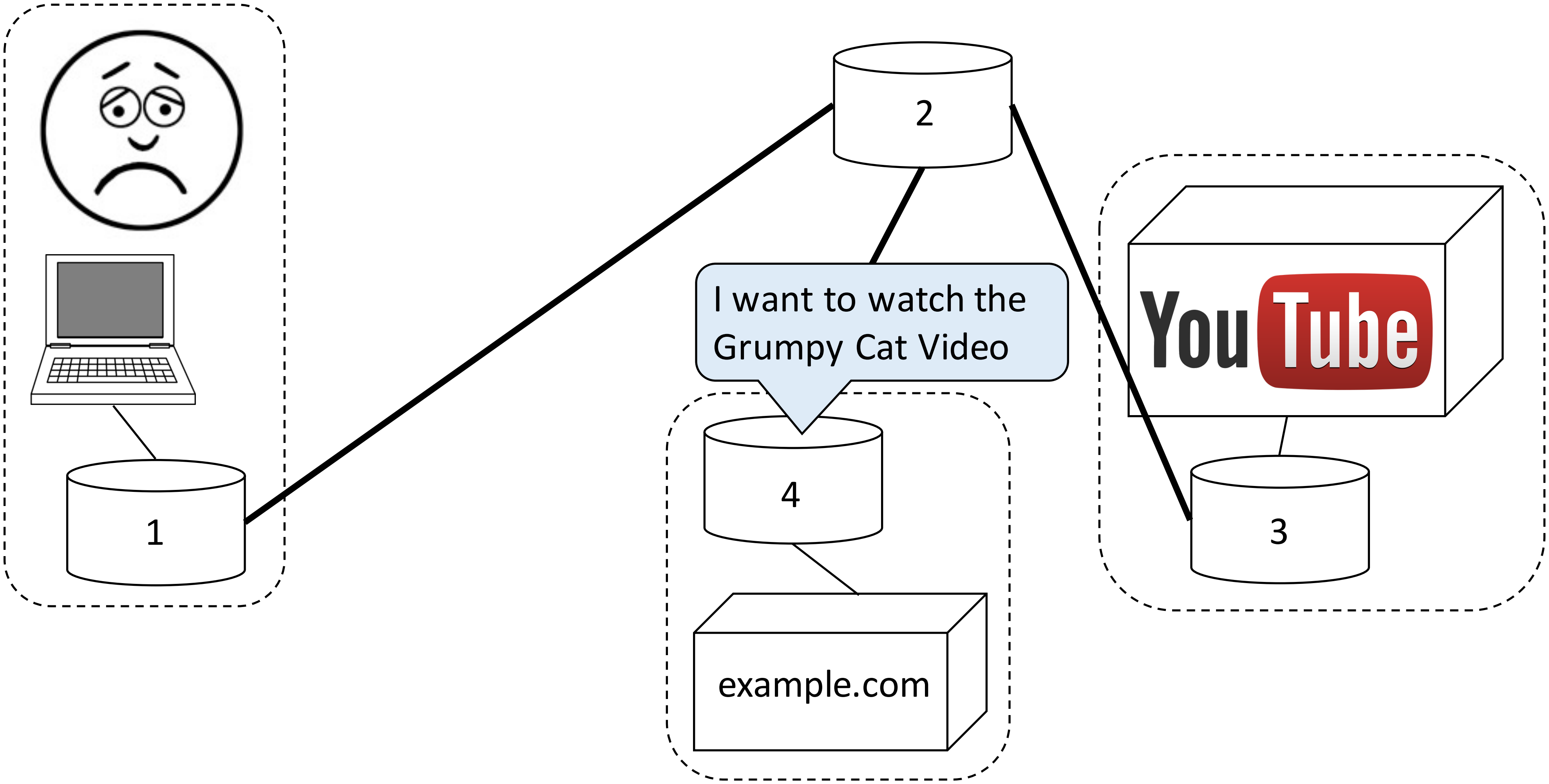
Lay Kuan Loh

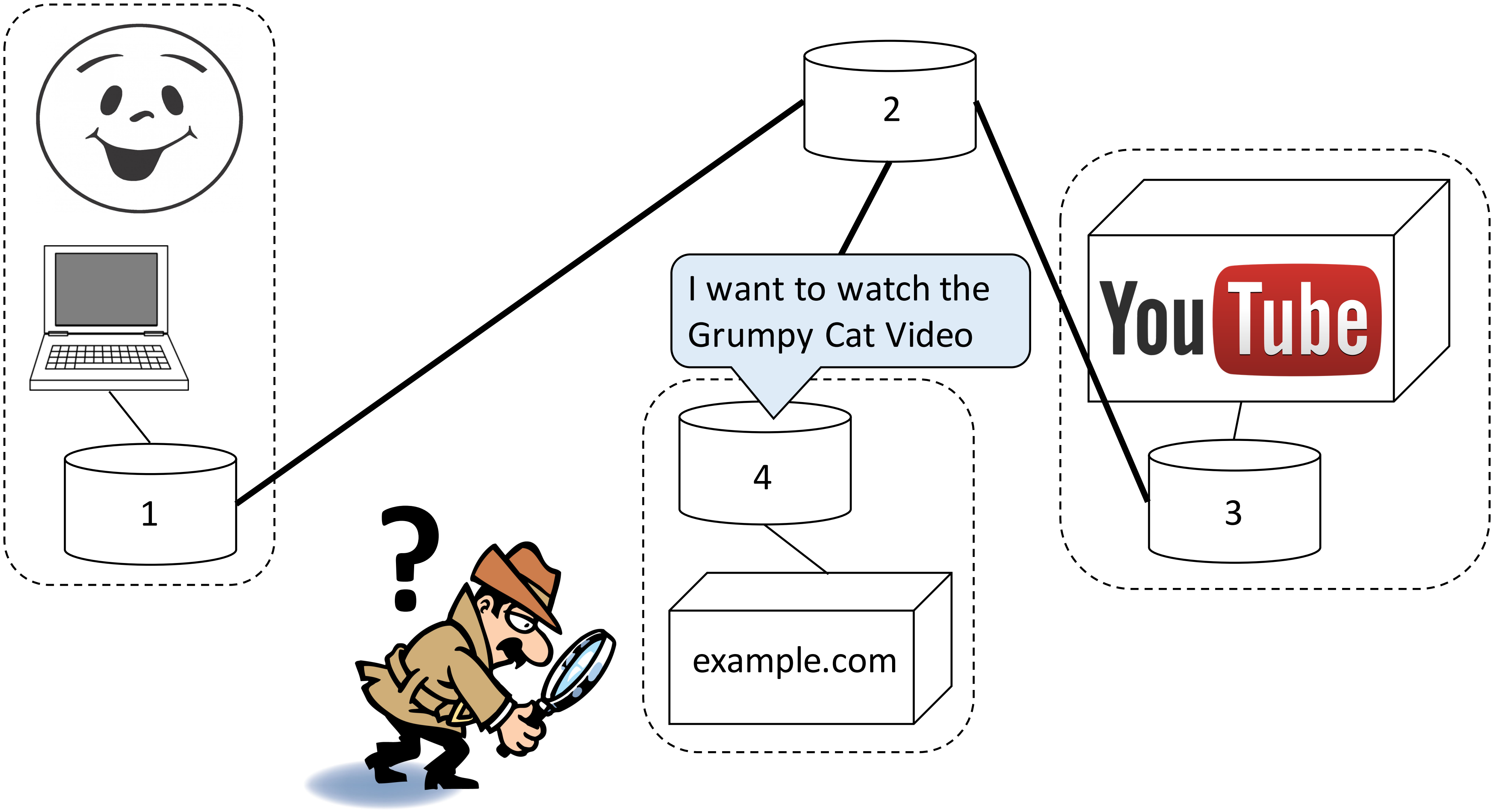
lloh@ece.cmu.edu




2
1
I want to watch the Grumpy Cat Video
4
example.com
YouTube
3

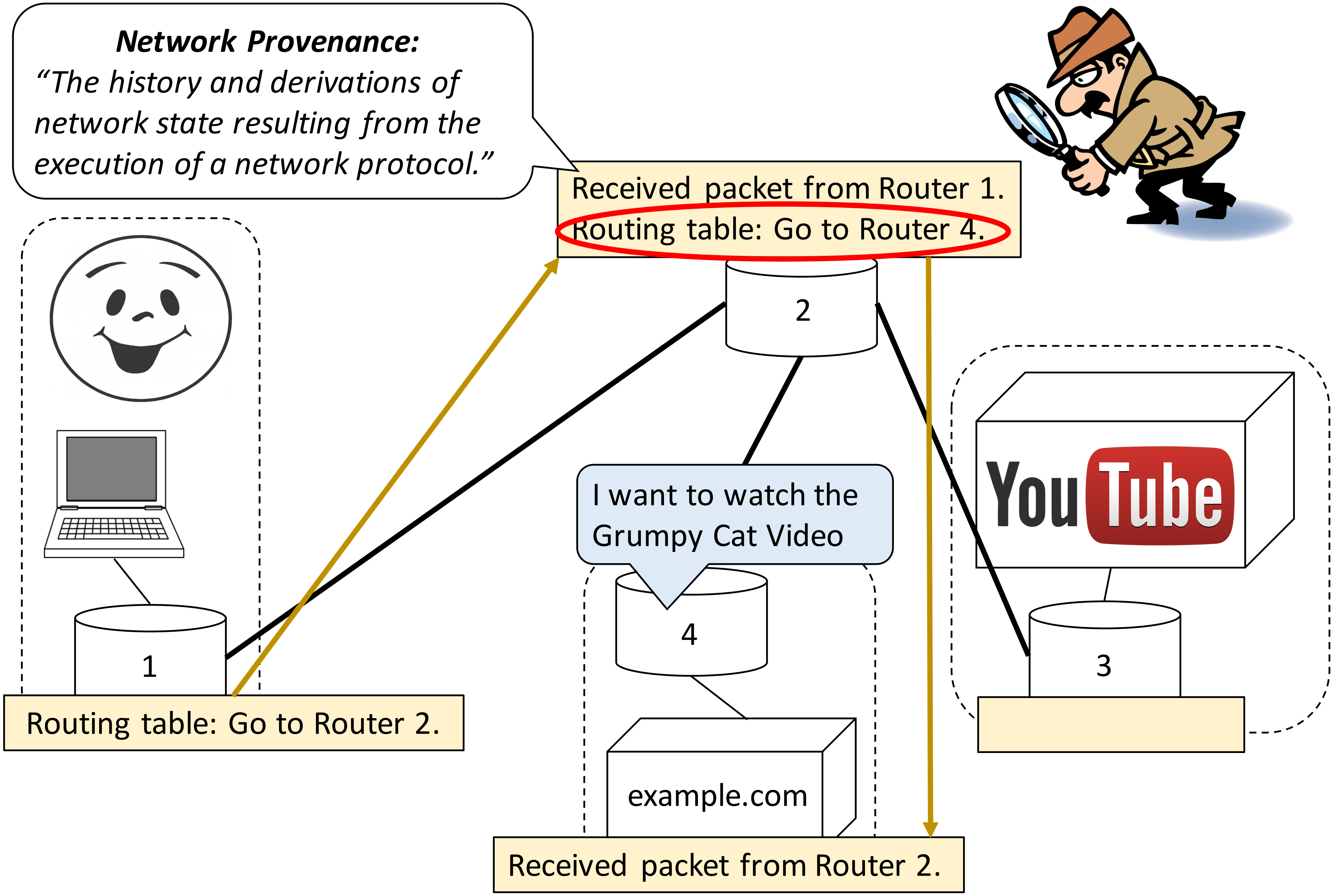

I want to watch the Grumpy Cat Video
1
2
3
4
YouTube
example.com


2
I want to watch the Grumpy Cat Video
YouTube
1
4
3
example.com



1
2
3
4
I want to watch the Grumpy Cat Video
example.com
YouTube
?

Network Provenance:
"The history and derivations of network state resulting from the execution of a network protocol."


Received packet from Router 1.
Routing table: Go to Router 4.
2
1
Routing table: Go to Router 2.
I want to watch the Grumpy Cat Video
4
example.com
Received packet from Router 2.
YouTube
3

Challenge

Large amount of storage needed to maintain network provenance at Internet-scale.

Our Solution

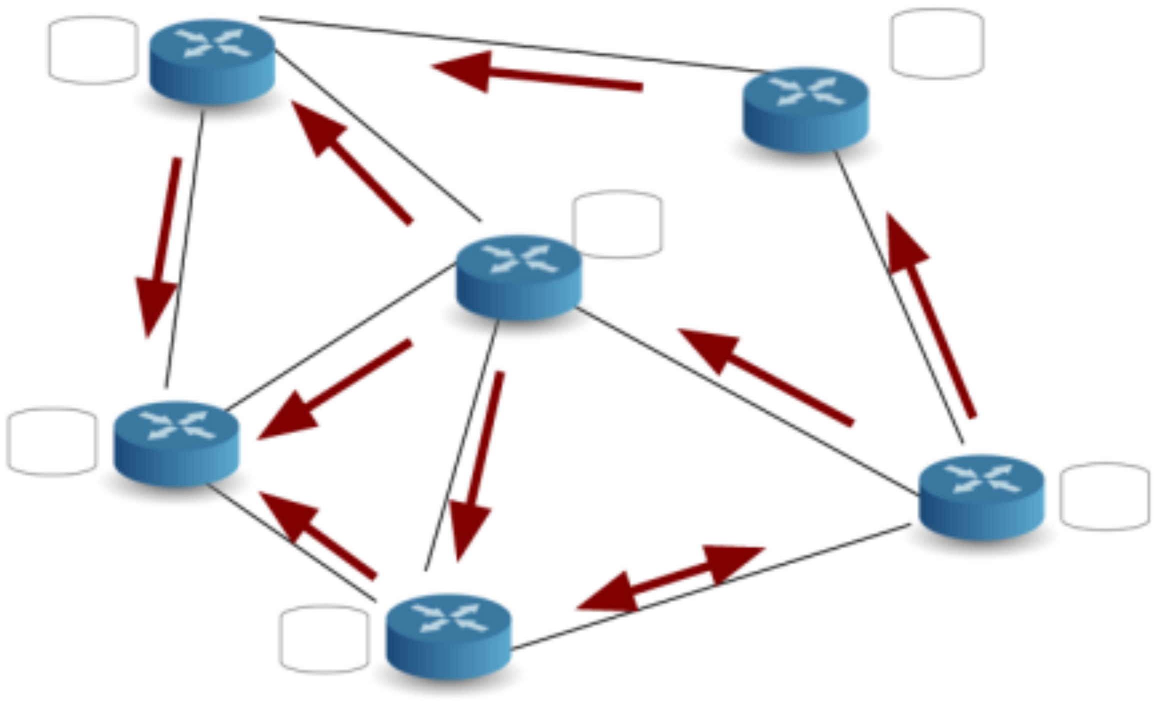
Identify and remove redundancy in network provenance before storing at runtime.

Roadmap

- ***Background***
- Key insights
- Our compression scheme
- Conclusion

Loo et al., *Declarative Networking.* SIGMOD'06.

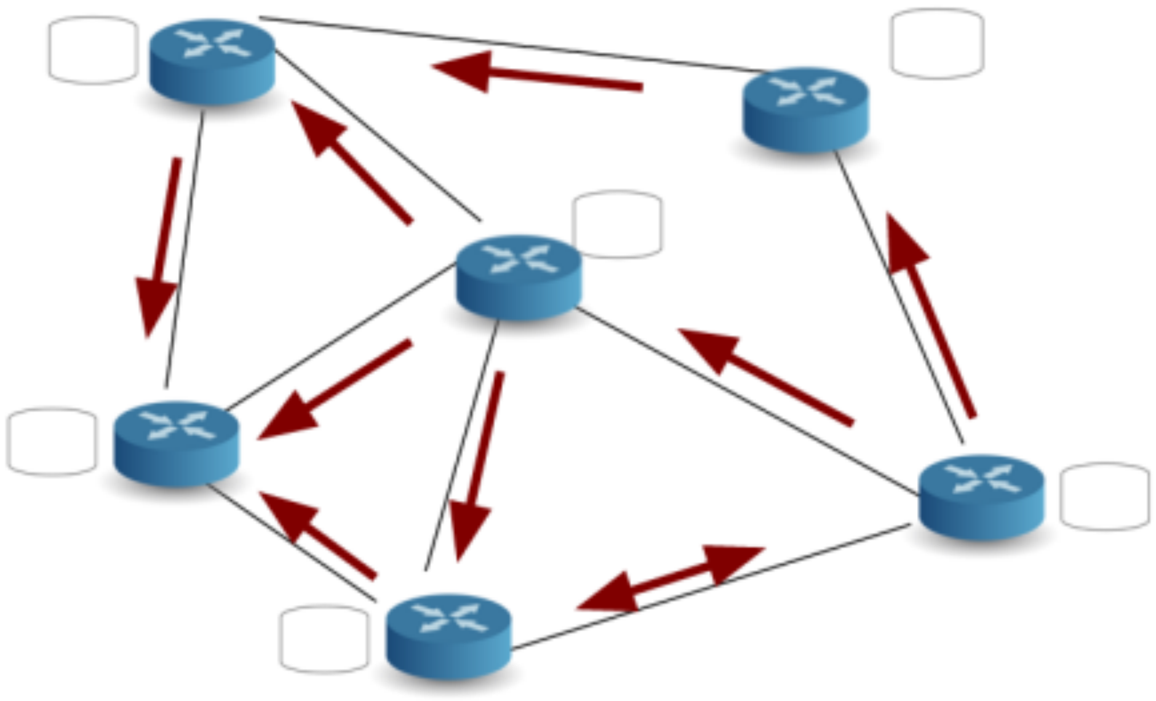
Declarative Networks



Traditional Network	Declarative Network
Network State • e.g. routing table	Distributed database • Info in routing table stored as tuples

Loo et al., *Declarative Networking.* SIGMOD'06.

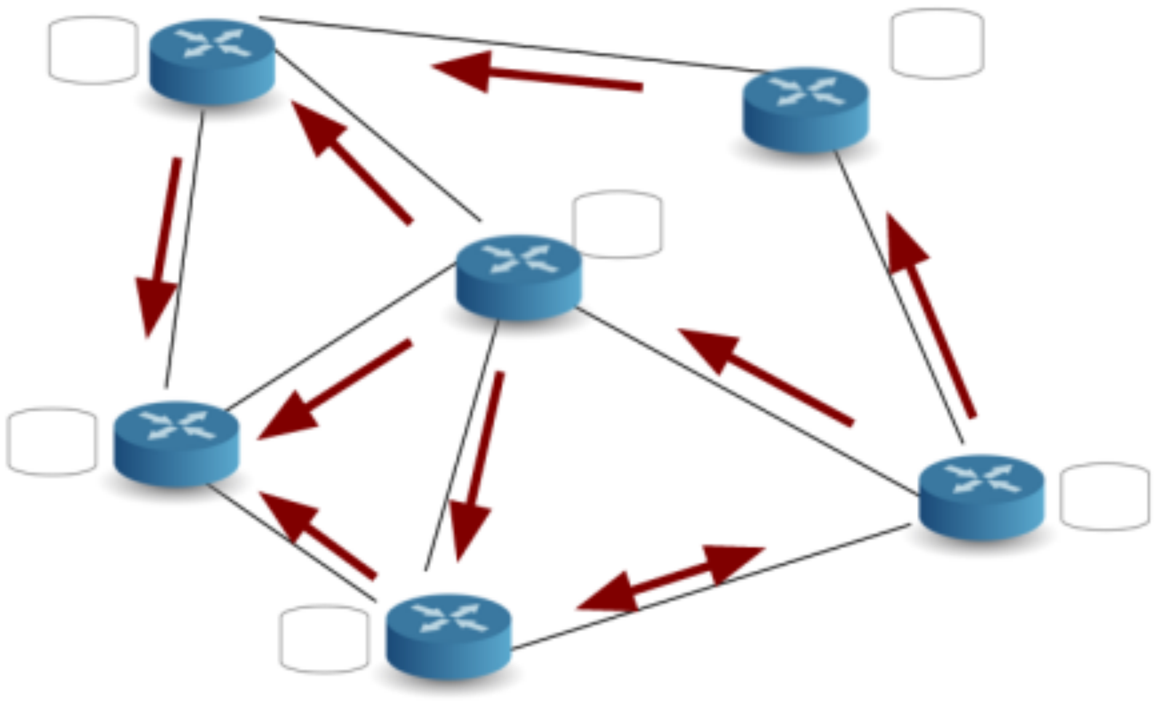
Declarative Networks



Traditional Network	**Declarative Network**
Network State • e.g. routing table	Distributed database • Info in routing table stored as ***tuples***
Network Protocol	***Network Datalog (NDLog)*** program

Loo et al., *Declarative Networking.* SIGMOD'06.

Declarative Networks



Q: Why ***NDLog***?
A: Concise

Traditional Network	**Declarative Network**
Network State • e.g. routing table	Distributed database • Info in routing table stored as ***tuples***
Network Protocol	***Network Datalog (NDLog)*** program

Loo et al., *Declarative Networking.* SIGMOD'06.

Review of Network Datalog (NDLog)

$$\boxed{\langle\text{result}\rangle \text{ :- } \underbrace{\langle\text{condition}_1\rangle, \ldots, \langle\text{condition}_N\rangle}_{\textit{Rule Body}}}$$

Rule Head

Packet Forwarding

```
r1 packet(@Neigh,Src,Dst,Payload)
    :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).

r2 recv(@Loc,Src,Dst,Payload)
    :- packet(@Loc,Src,Dst,Payload),  Dst==Loc.
```

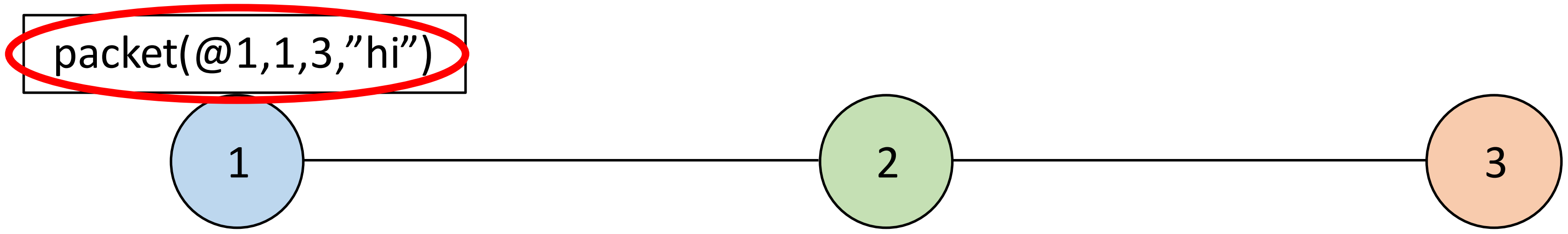
Loo et al., *Declarative Networking.* SIGMOD'06.

An example NDLog Program

```
Packet Forwarding
r1 packet(@Neigh,Src,Dst,Payload)
     :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).
r2 recv(@Loc,Src,Dst,Payload)
     :- packet(@Loc,Src,Dst,Payload),  Dst==Loc.
```

Fast-changing

Updated automatically by program execution



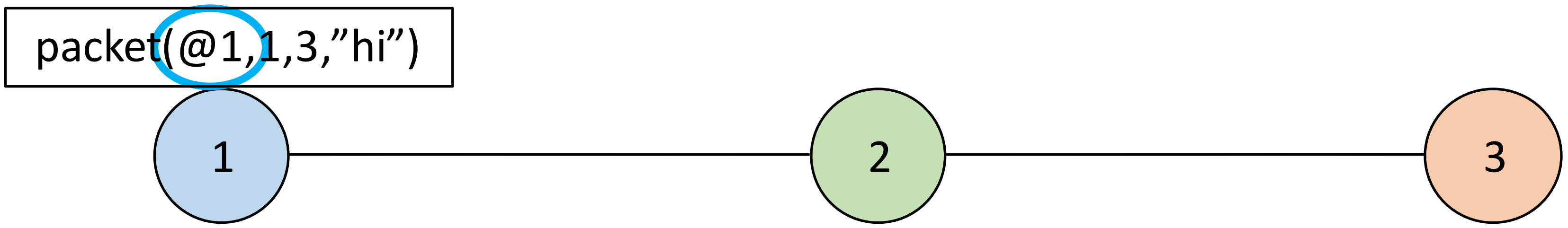
Loo et al., *Declarative Networking.* SIGMOD'06.

An example NDLog Program

Packet Forwarding

```
r1 packet(@Neigh,Src,Dst,Payload)
    :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).
r2 recv(@Loc,Src,Dst,Payload)
    :- packet(@Loc,Src,Dst,Payload),  Dst==Loc.
```

Location Specifier



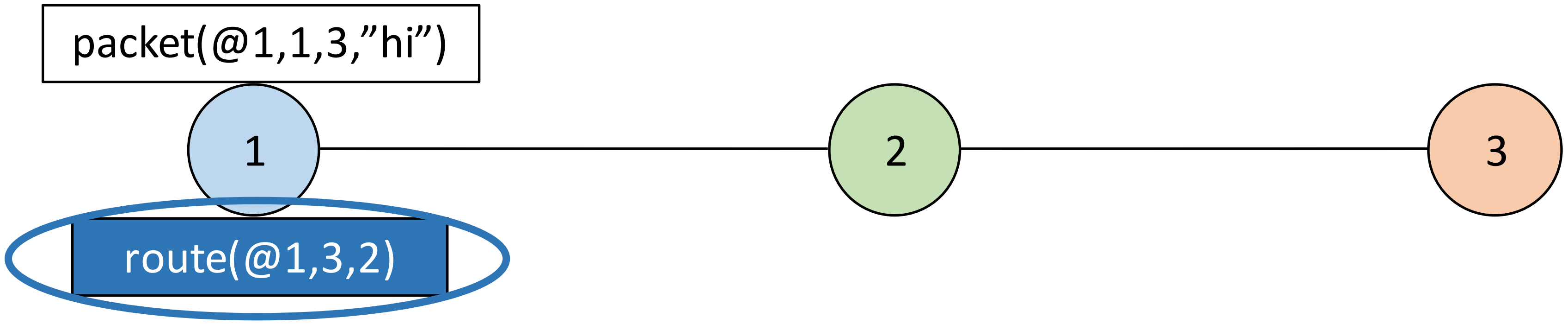
Loo et al., *Declarative Networking.* SIGMOD'06.

An example NDLog Program

<u>Packet Forwarding</u>

```
r1 packet(@Neigh,Src,Dst,Payload)
    :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).
r2 recv(@Loc,Src,Dst,Payload)
    :- packet(@Loc,Src,Dst,Payload), Dst==Loc.
```

Slow-changing
Interval between updates is usually longer than the lifespan of program execution



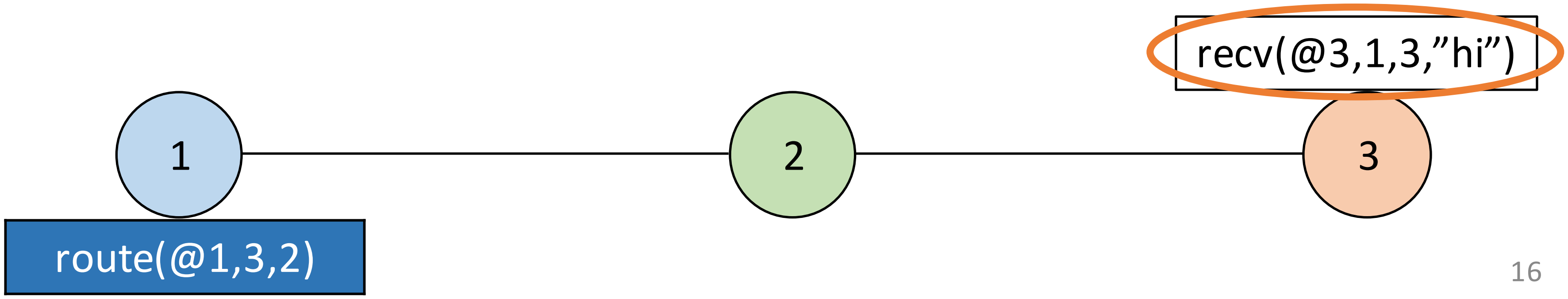
Loo et al., *Declarative Networking.* SIGMOD’06.

An example NDLog Program

```
Packet Forwarding
r1 packet(@Neigh,Src,Dst,Payload)
    :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).
r2 recv(@Loc,Src,Dst,Payload)
    :- packet(@Loc,Src,Dst,Payload),  Dst==Loc.
```

Fast-changing

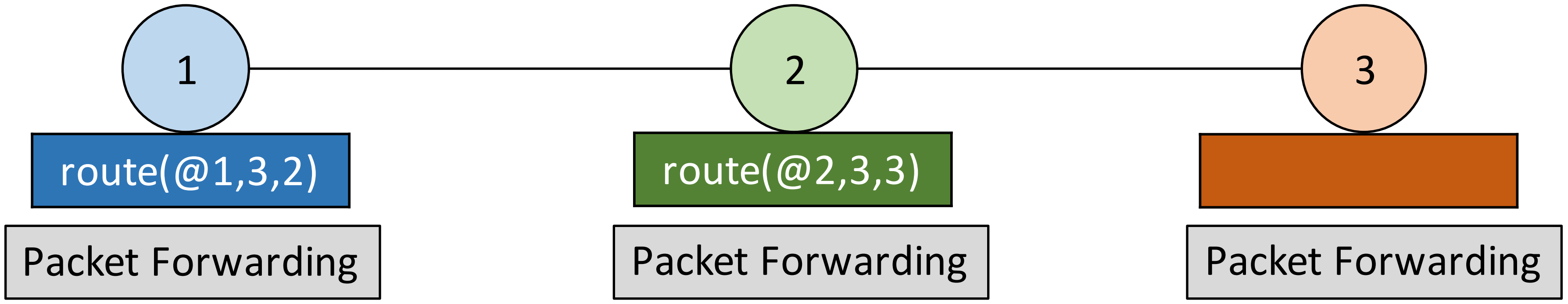
Updated automatically by program execution



Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

Packet Forwarding

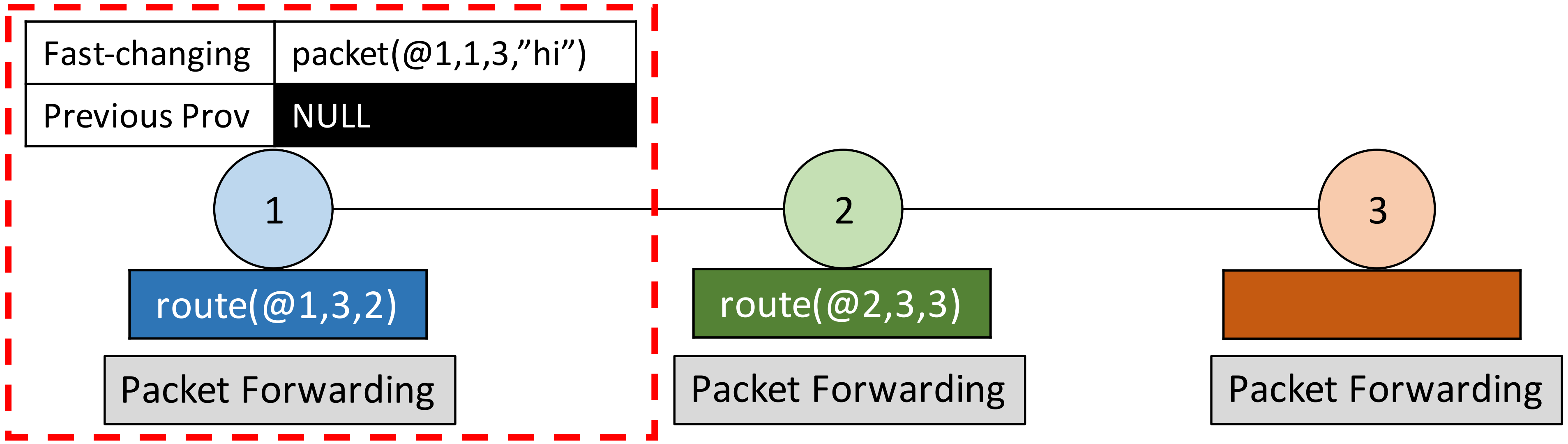
```
r1 packet(@Neigh,Src,Dst,Payload)  :- packet(@Loc,Src,Dst,Payload),  route(@Loc,Dst,Neigh).
r2 recv(@Loc,Src,Dst,Payload)       :- packet(@Loc,Src,Dst,Payload),  Dst==Loc.
```



Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

Packet Forwarding

```
r1 packet(@Neigh,Src,Dst,Payload)  :- packet(@Loc,Src,Dst,Payload),  route(@Loc,Dst,Neigh).
r2 recv(@Loc,Src,Dst,Payload)       :- packet(@Loc,Src,Dst,Payload),   Dst==Loc.
```

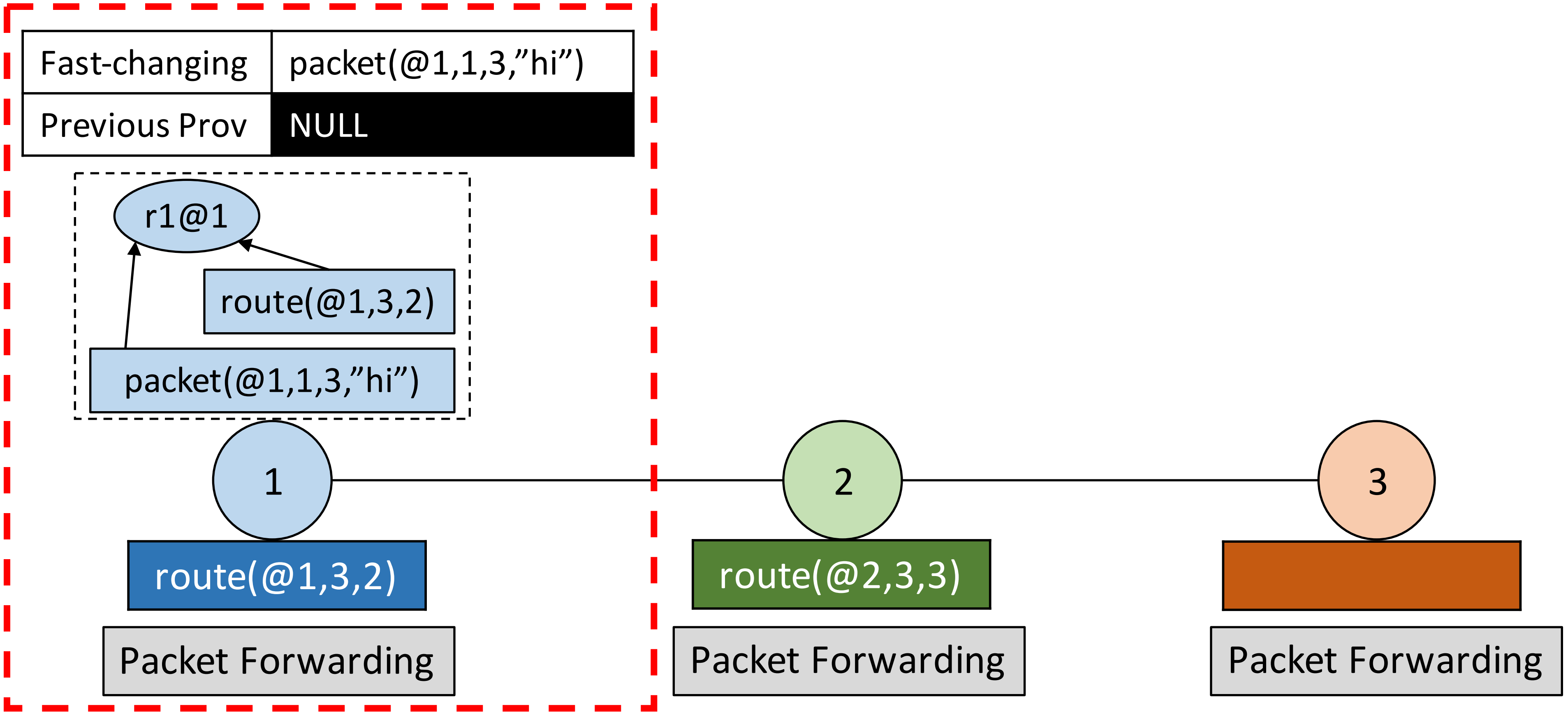


Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

Packet Forwarding

r1 packet(@Neigh,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).

r2 recv(@Loc,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), Dst==Loc.

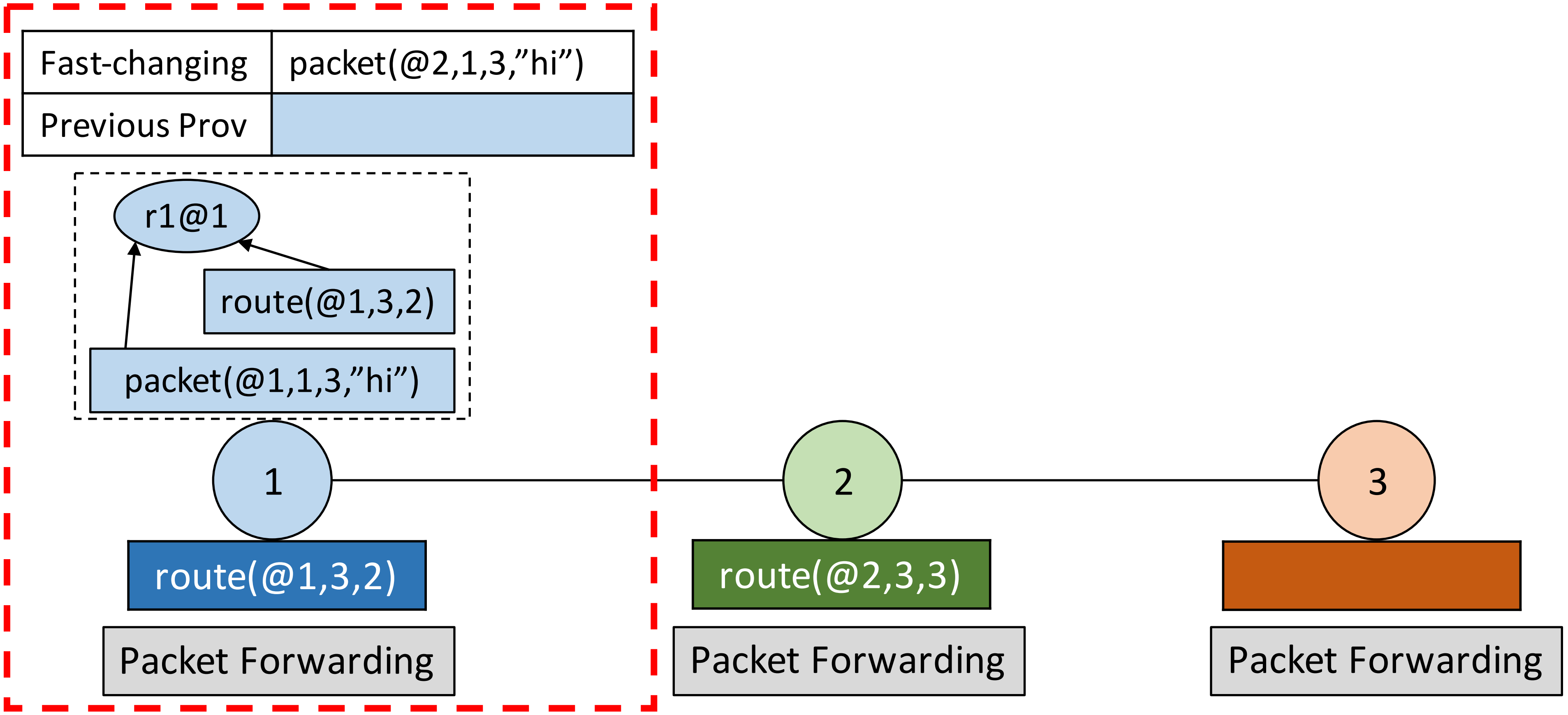


Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

Packet Forwarding

r1 packet(@Neigh,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).

r2 recv(@Loc,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), Dst==Loc.

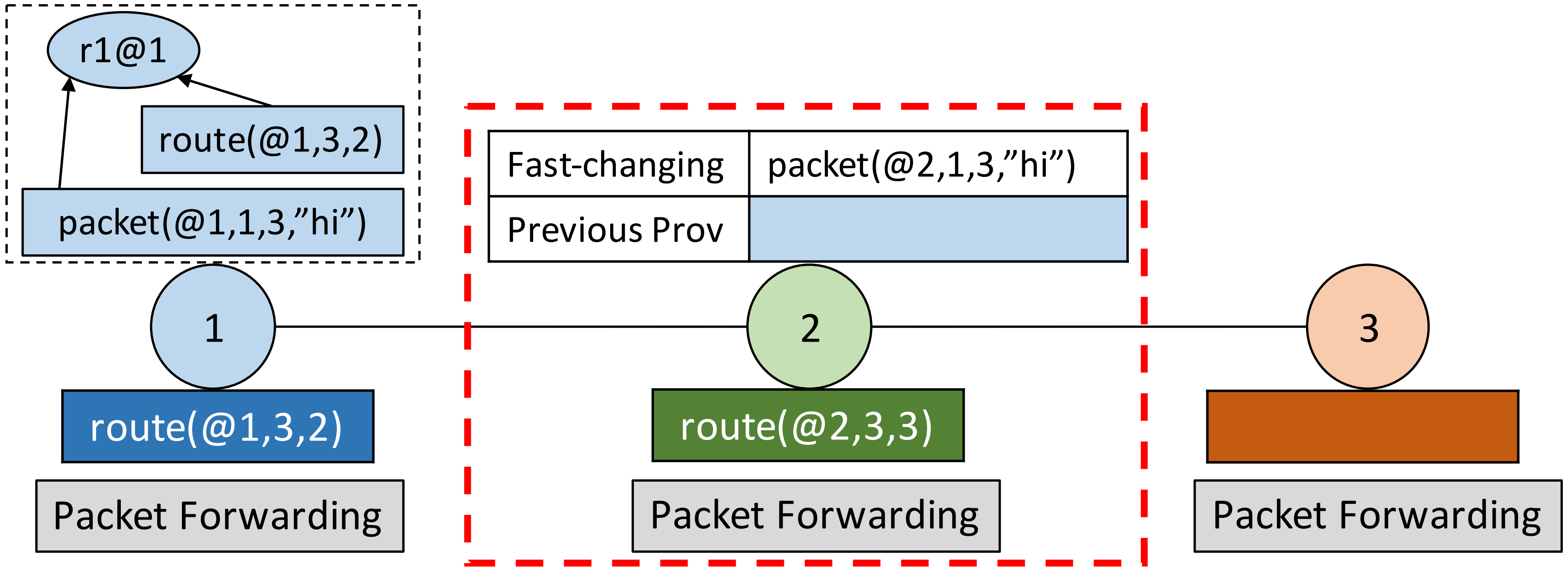


Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

Packet Forwarding

r1 packet(@Neigh,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).

r2 recv(@Loc,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), Dst==Loc.

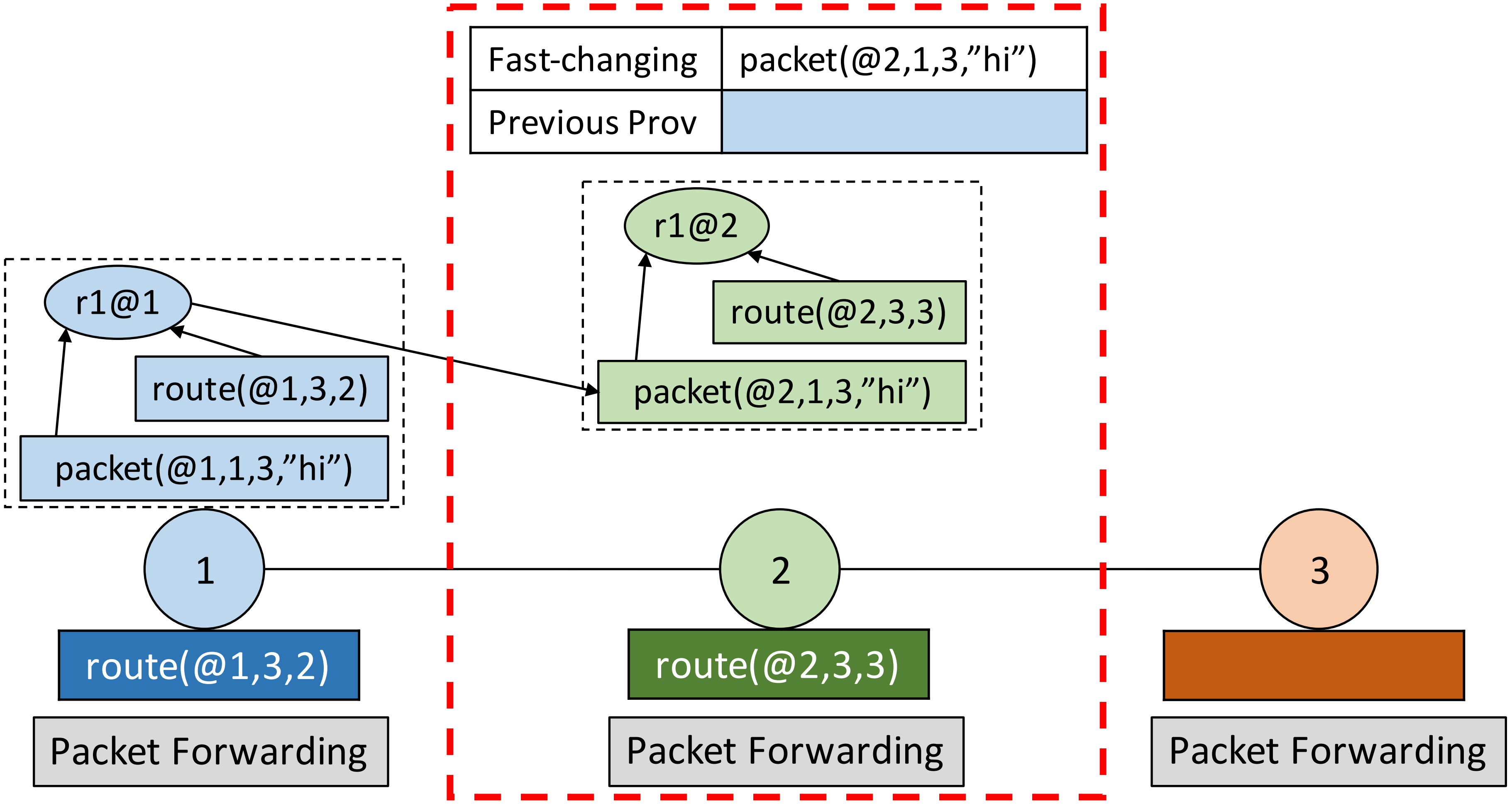


Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

<u>Packet Forwarding</u>

r1 packet(@Neigh,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).

r2 recv(@Loc,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), Dst==Loc.

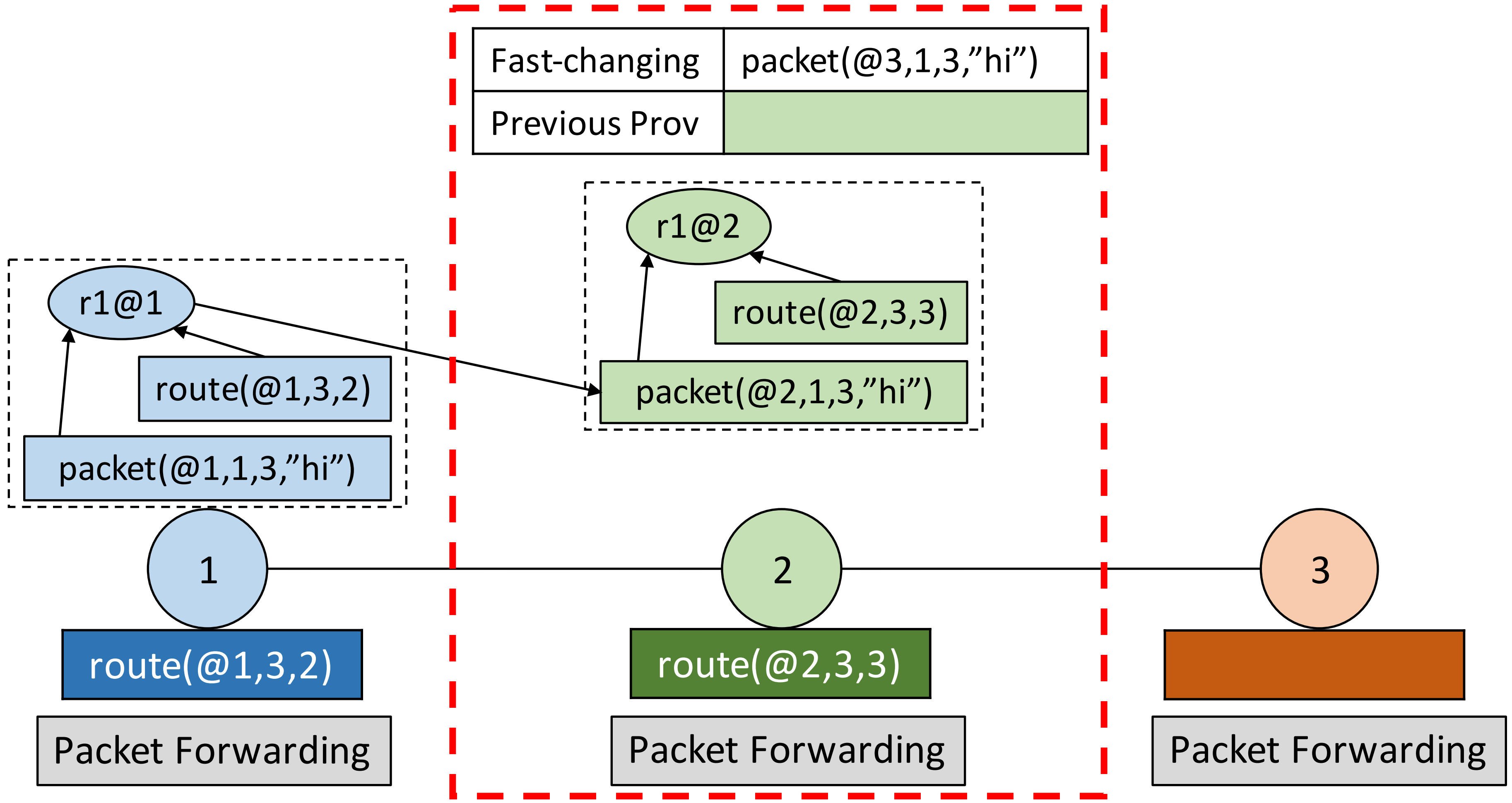


Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

Packet Forwarding

r1 packet(@Neigh,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).

r2 recv(@Loc,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), Dst==Loc.

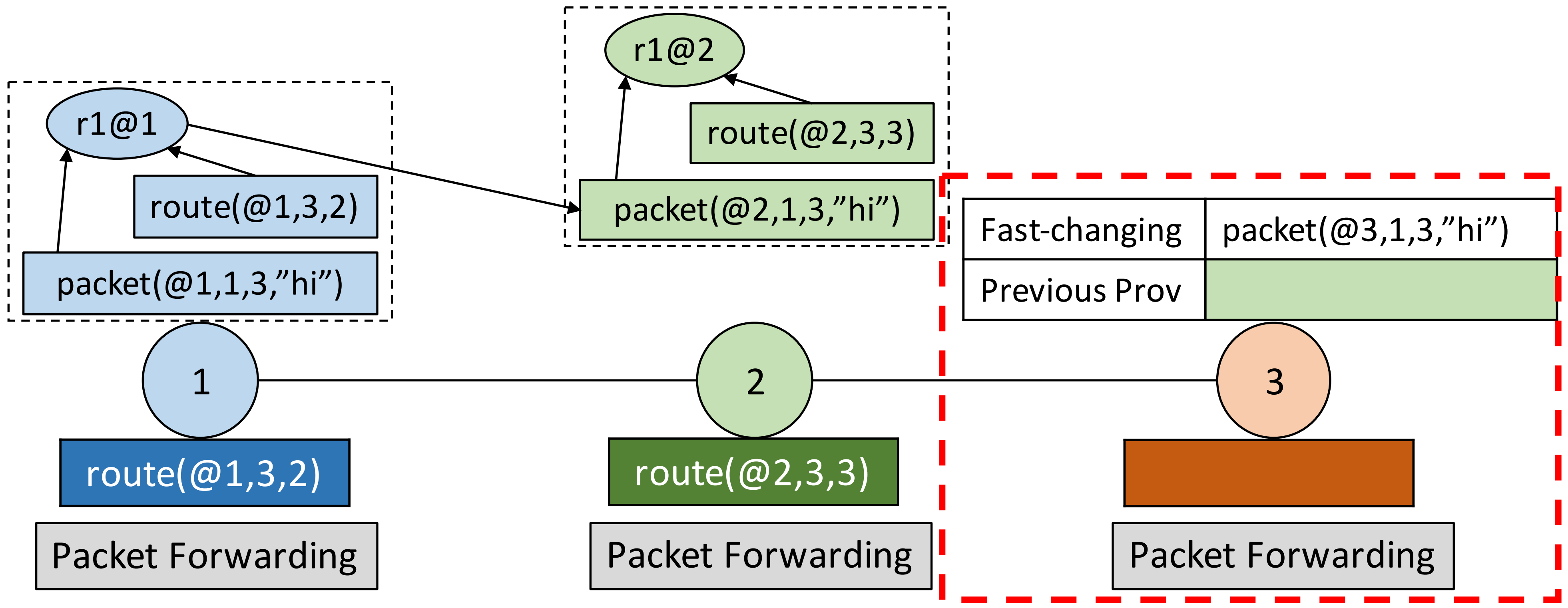


Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

Packet Forwarding

r1 packet(@Neigh,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).

r2 recv(@Loc,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), Dst==Loc.

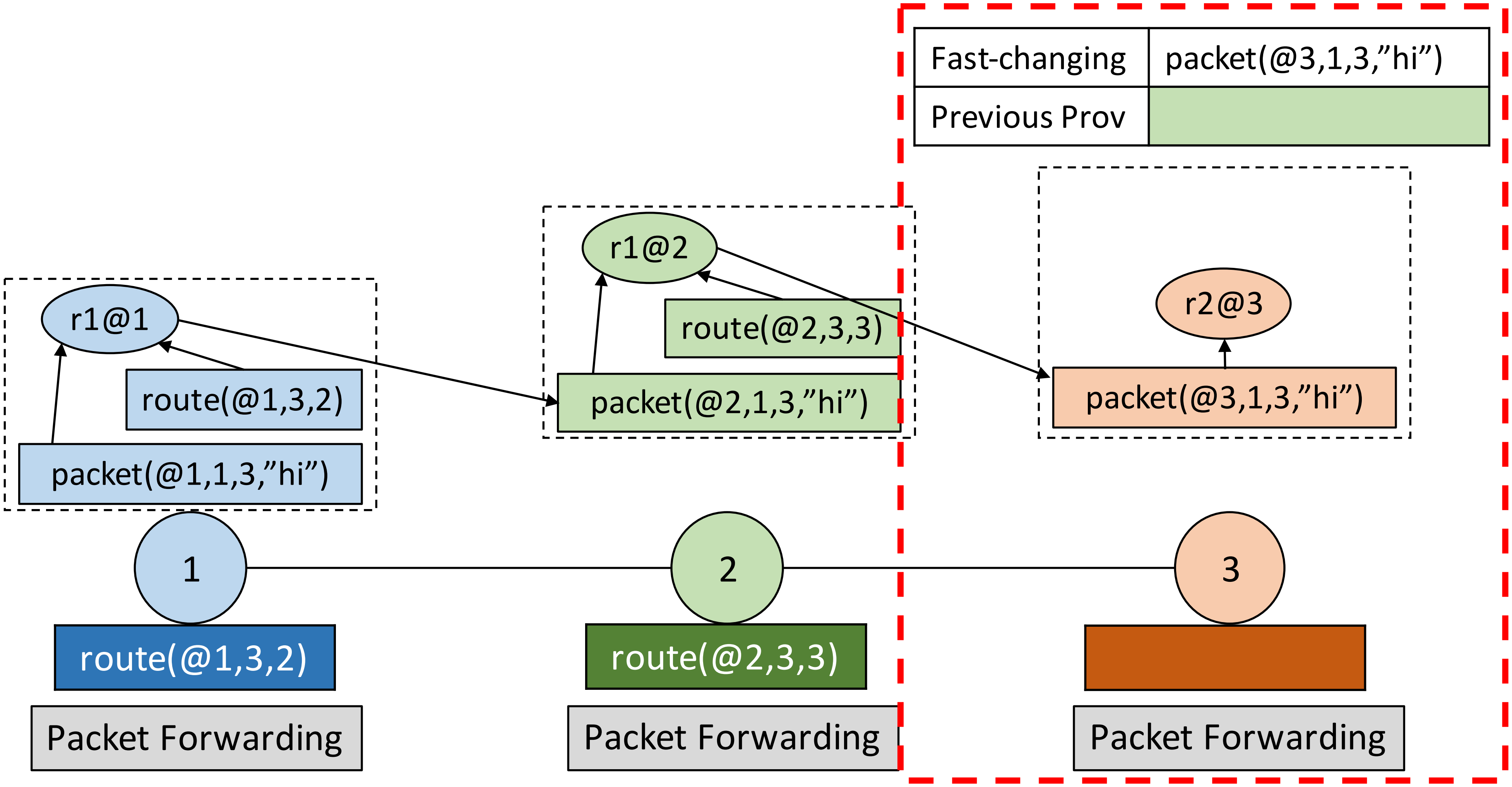


Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

Packet Forwarding

r1 packet(@Neigh,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).

r2 recv(@Loc,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), Dst==Loc.



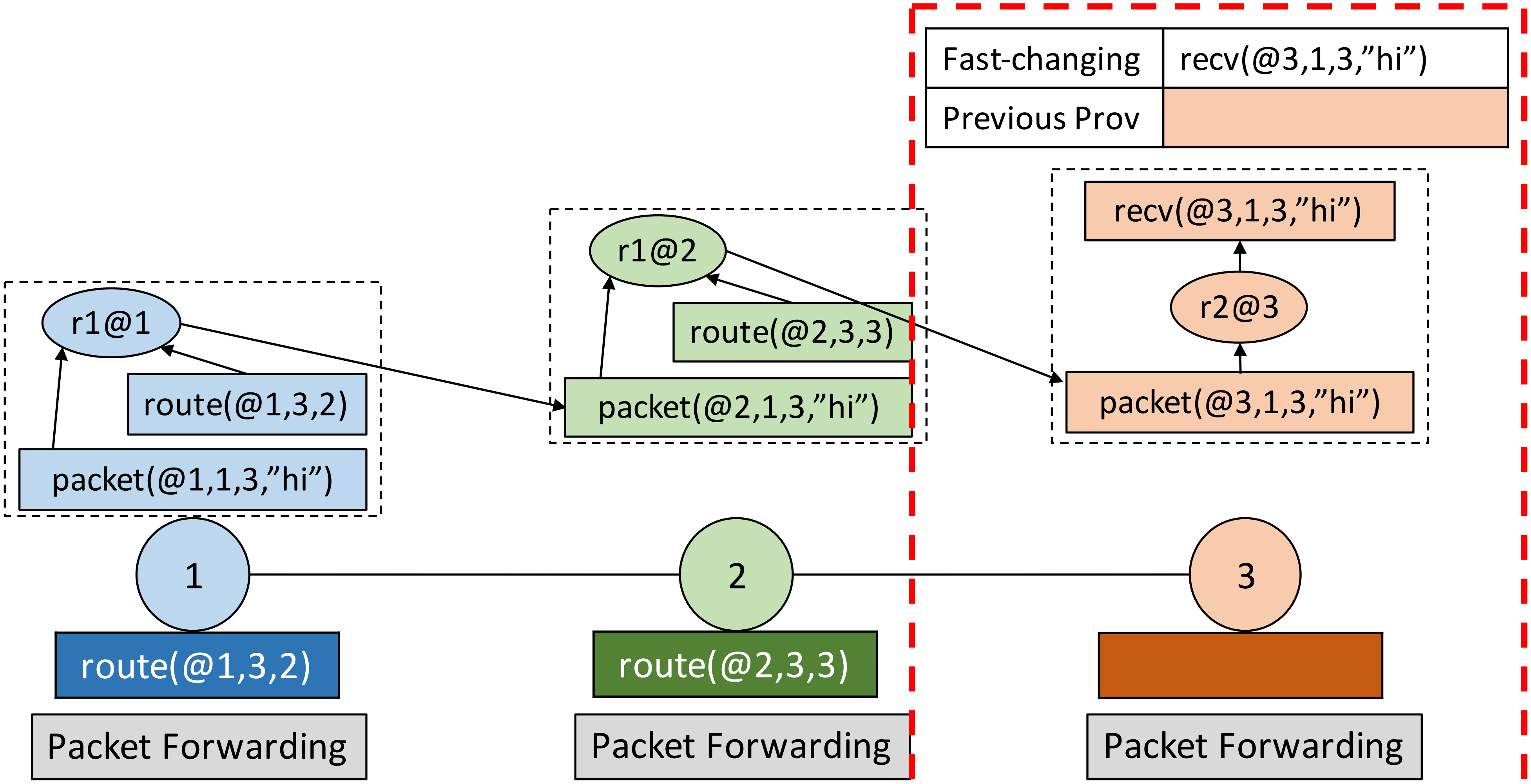
Fast-changing	packet(@3,1,3,"hi")
Previous Prov	🟩

Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

<u>Packet Forwarding</u>

r1 packet(@Neigh,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).

r2 recv(@Loc,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), Dst==Loc.



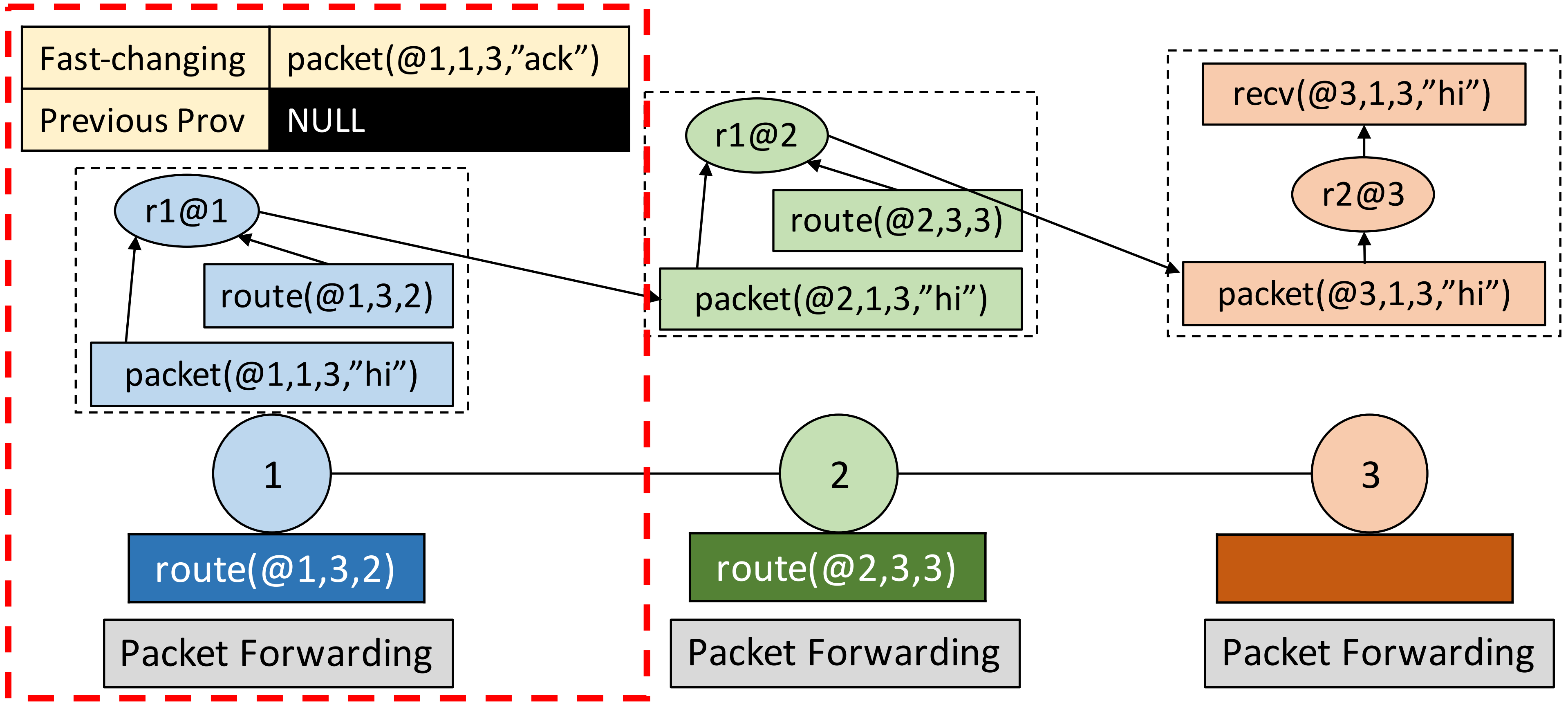
Fast-changing	recv(@3,1,3,"hi")
Previous Prov	

Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

Packet Forwarding

r1 packet(@Neigh,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).

r2 recv(@Loc,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), Dst==Loc.

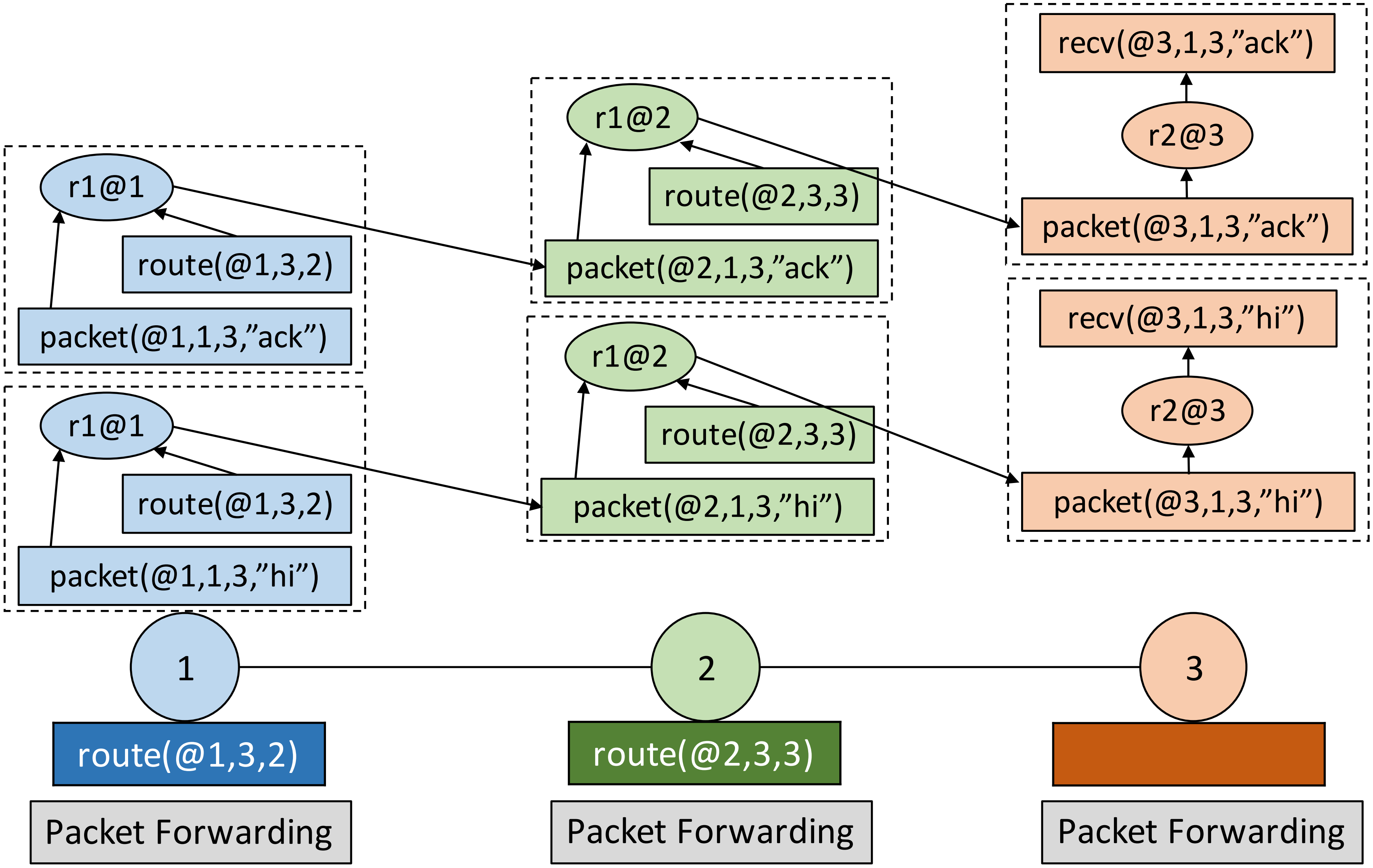


Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

Packet Forwarding

r1 packet(@Neigh,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).

r2 recv(@Loc,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), Dst==Loc.

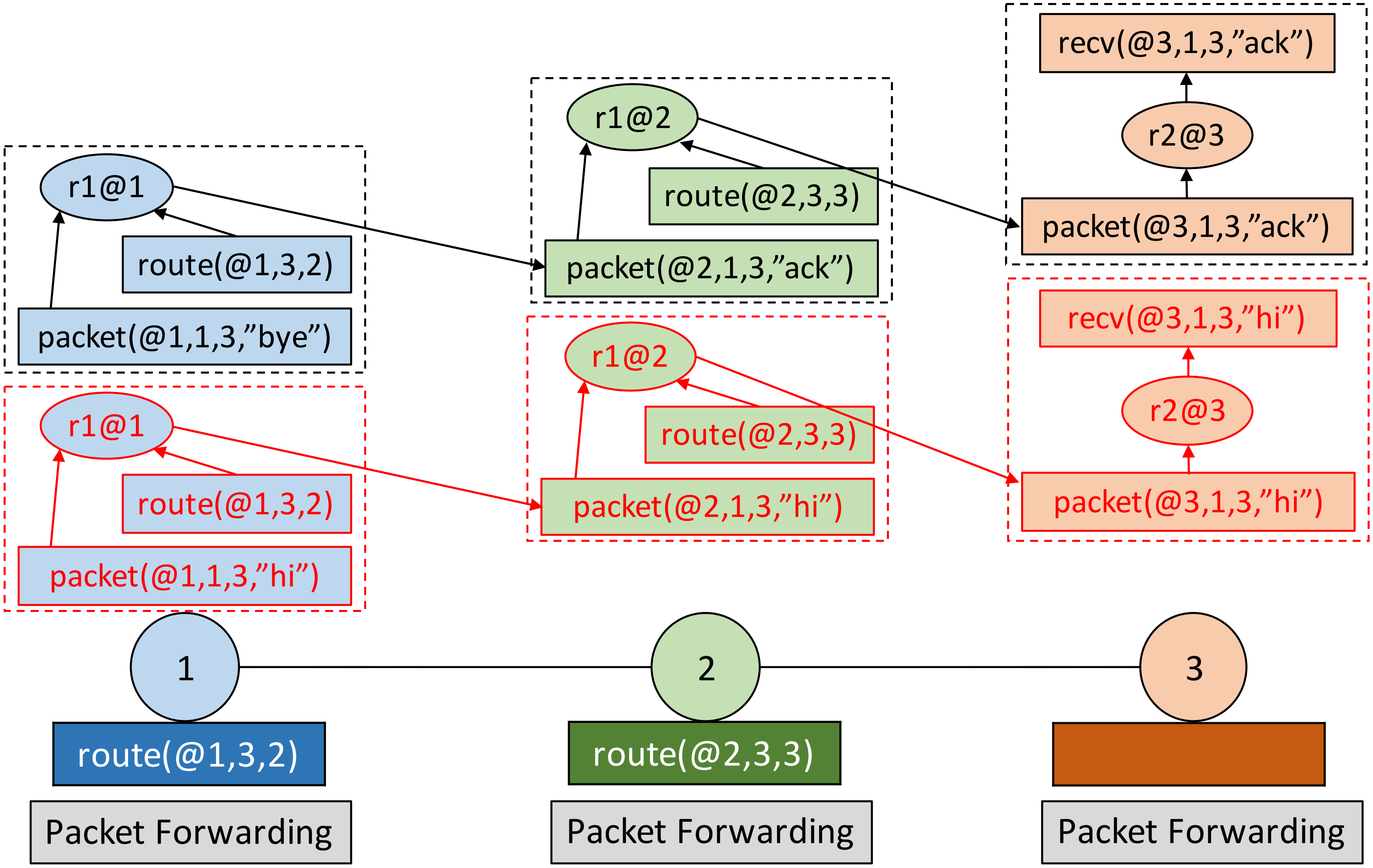


Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

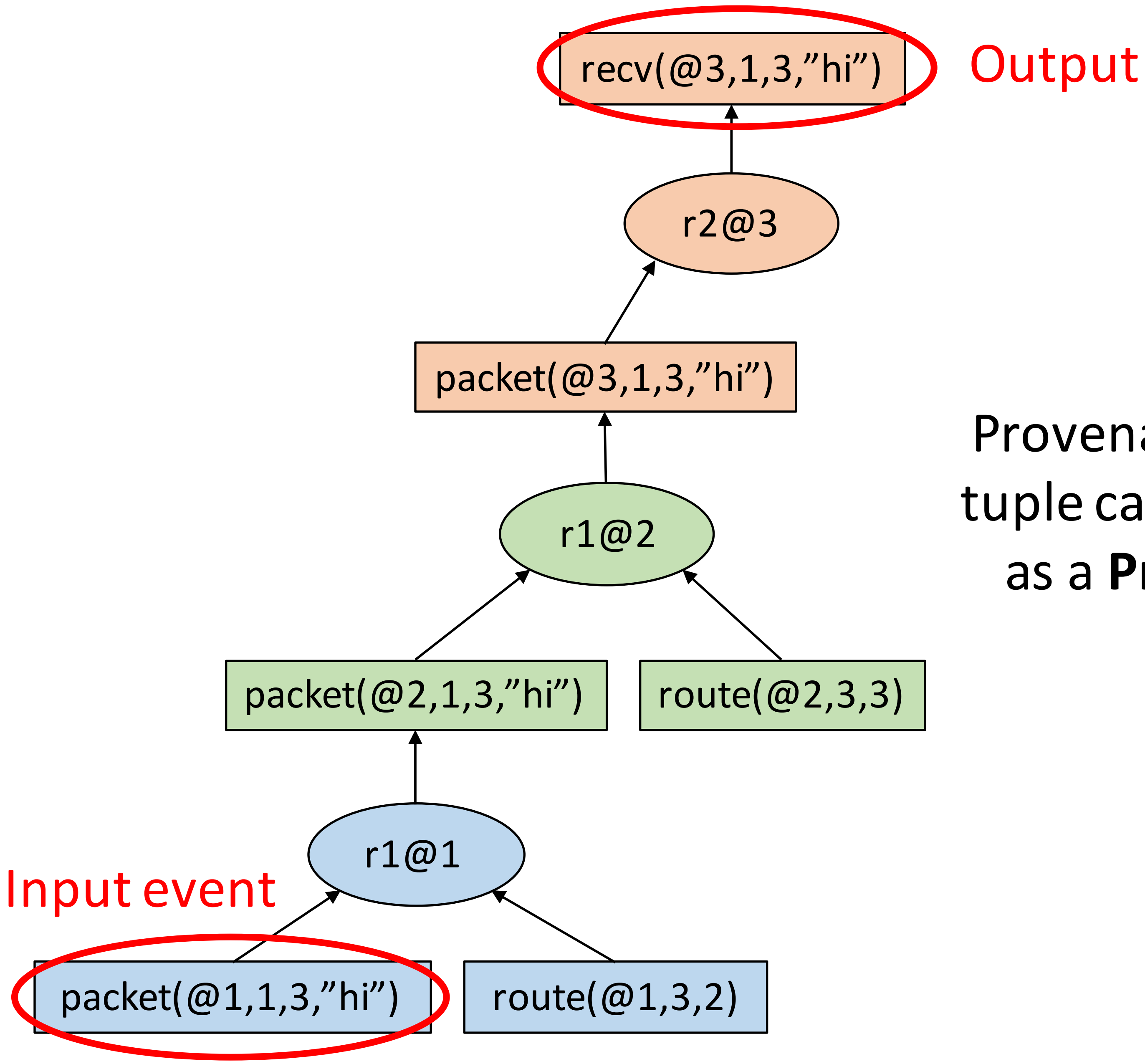
Packet Forwarding

r1 packet(@Neigh,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), route(@Loc,Dst,Neigh).

r2 recv(@Loc,Src,Dst,Payload) :- packet(@Loc,Src,Dst,Payload), Dst==Loc.



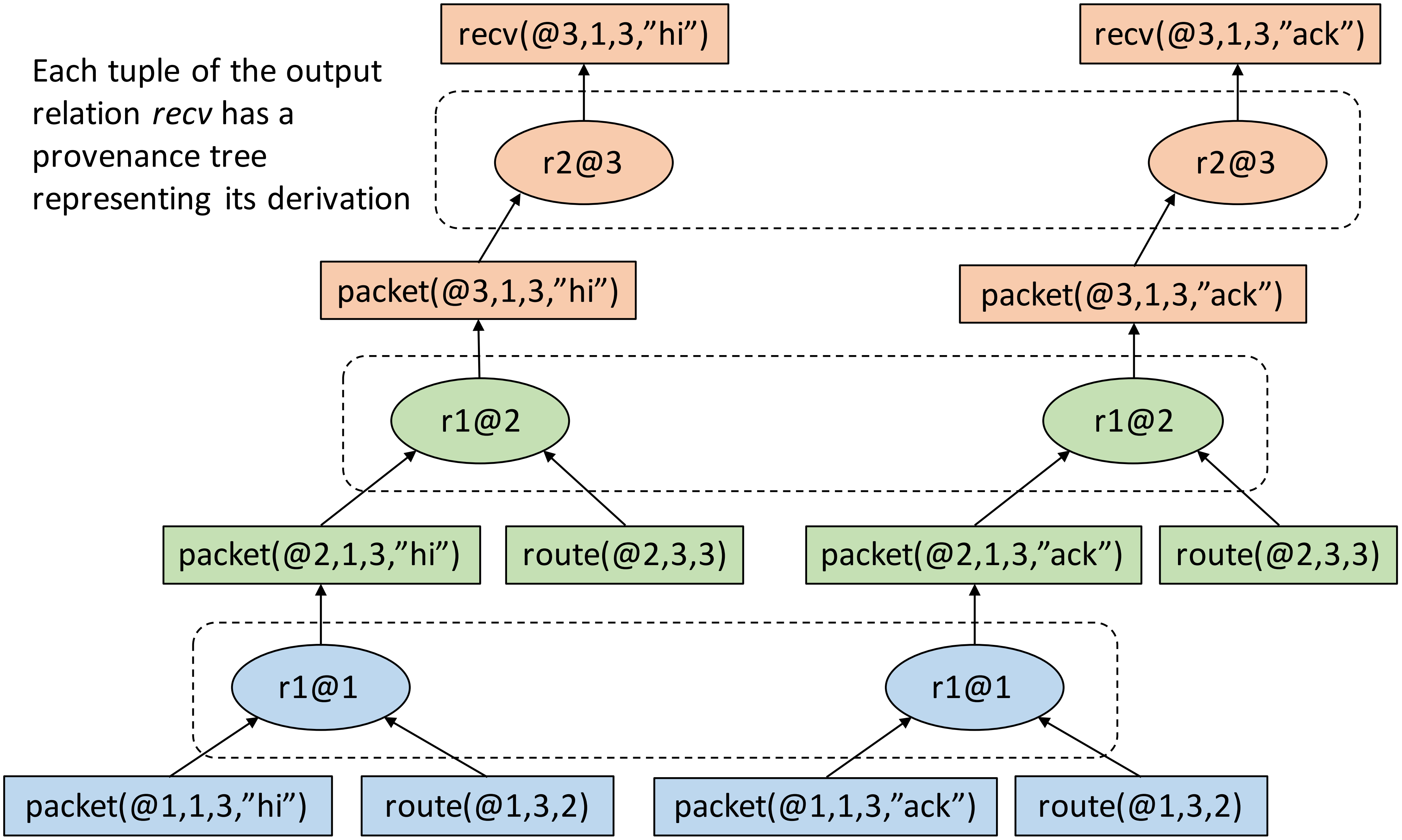
Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10



Provenance of a derived tuple can be represented as a **Provenance Tree**

Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale.* SIGMOD'10

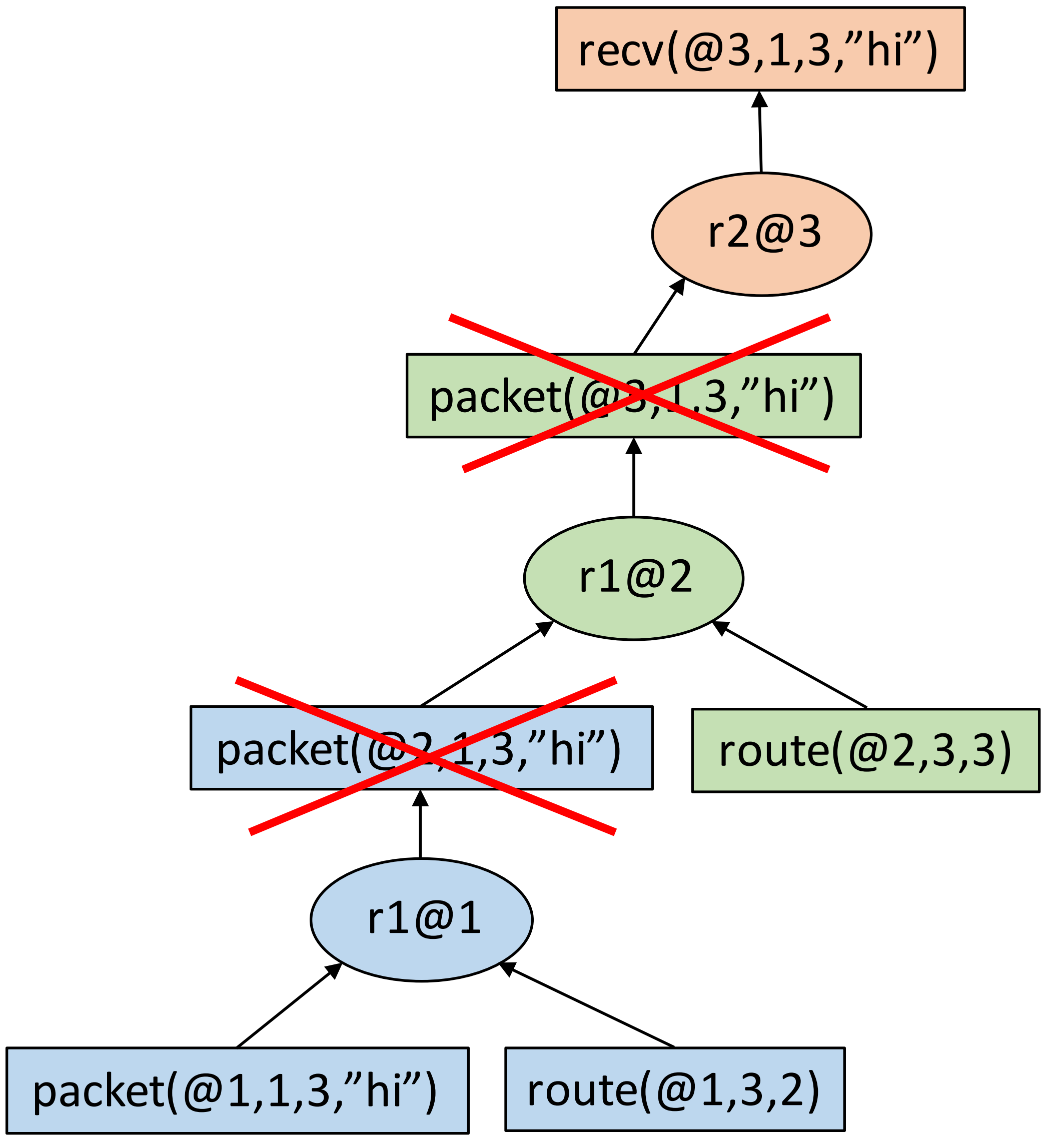
Each tuple of the output relation *recv* has a provenance tree representing its derivation



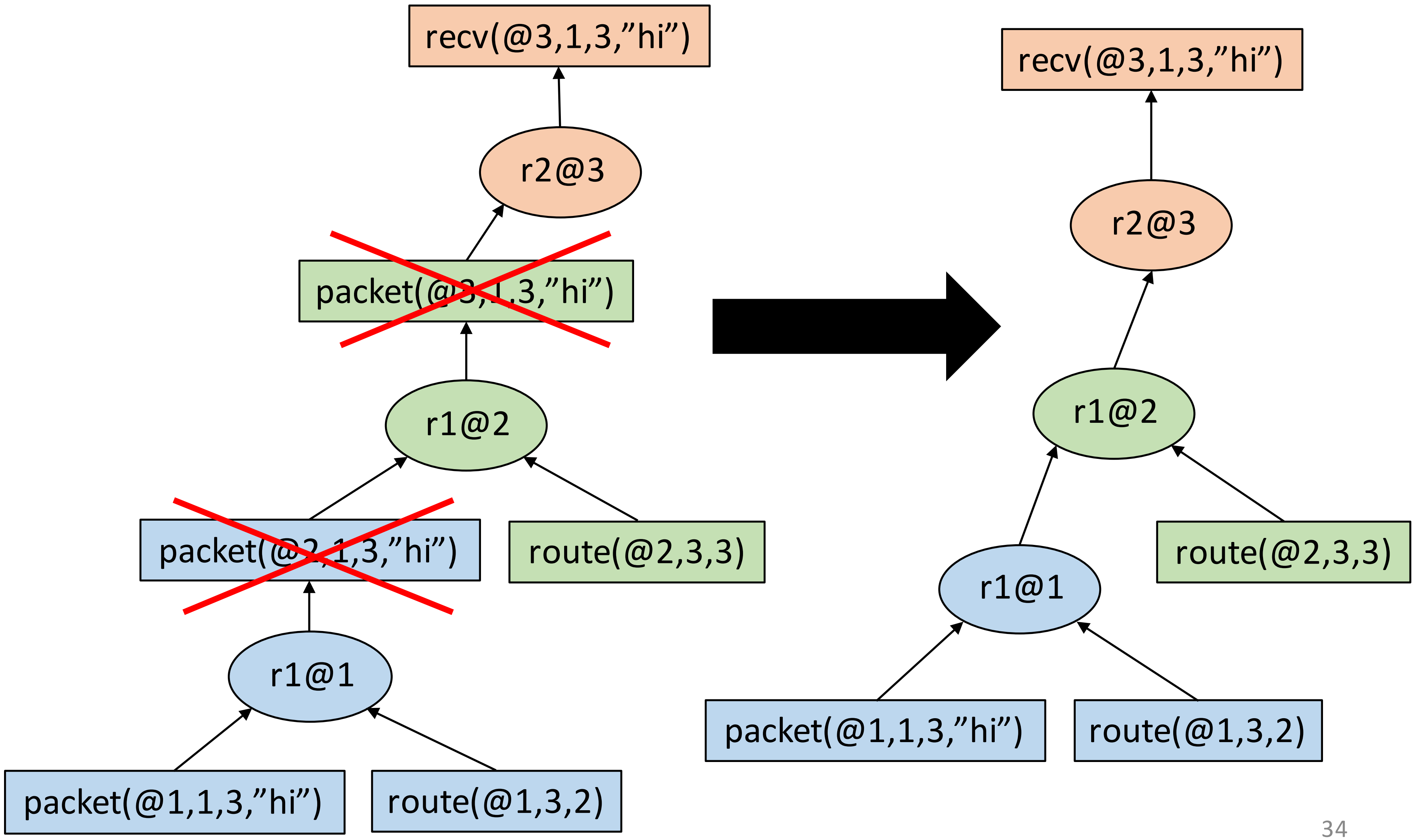
Roadmap

- Background
- ***Key insights***
- Our compression scheme
- Conclusion

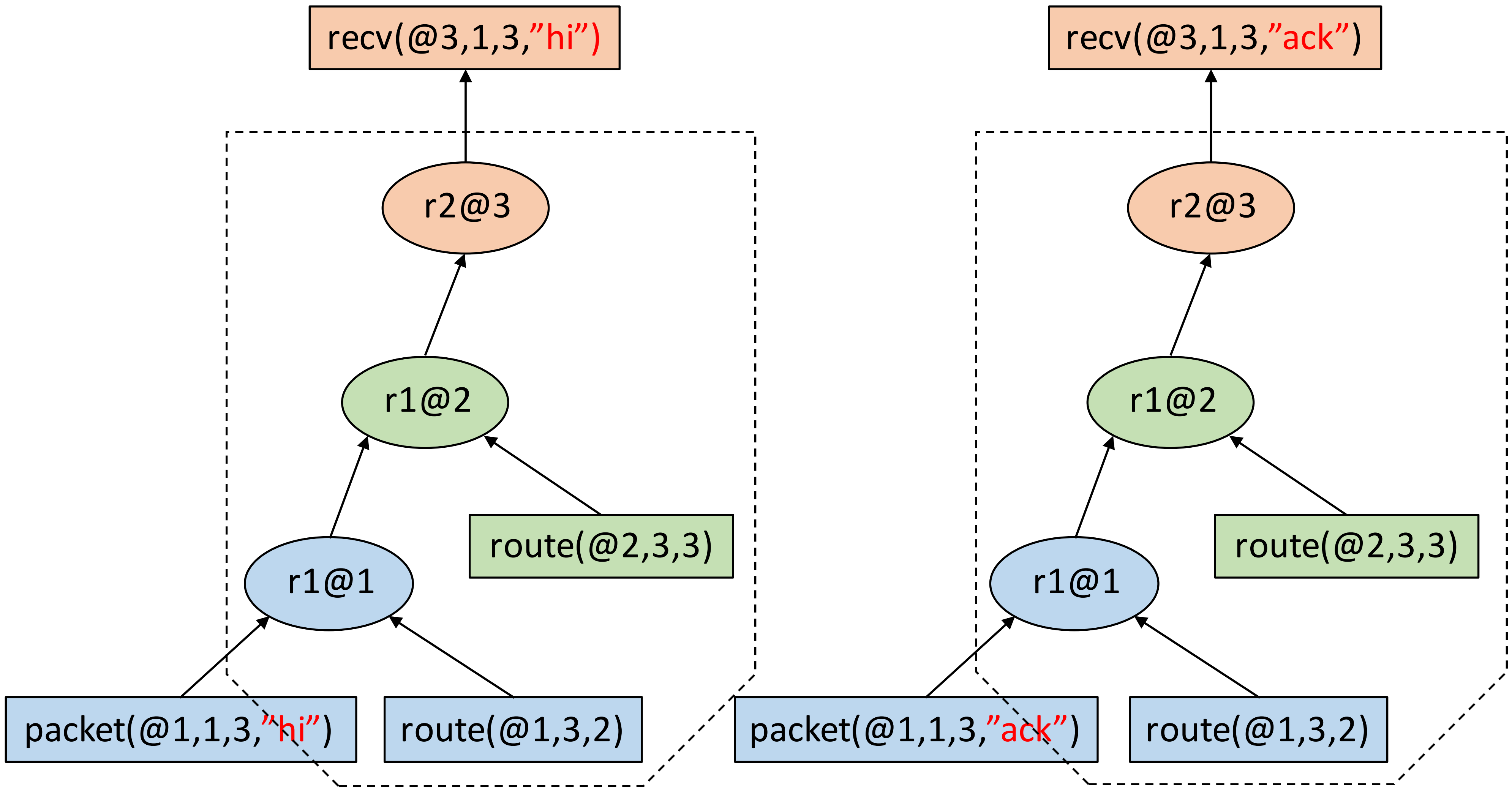
Insight #1: Remove Redundancy



Insight #1: Remove Redundancy



Insight #2:
Different packets may follow the same path



Simplifying Assumption #1

One fast-changing relation per NDLog rule

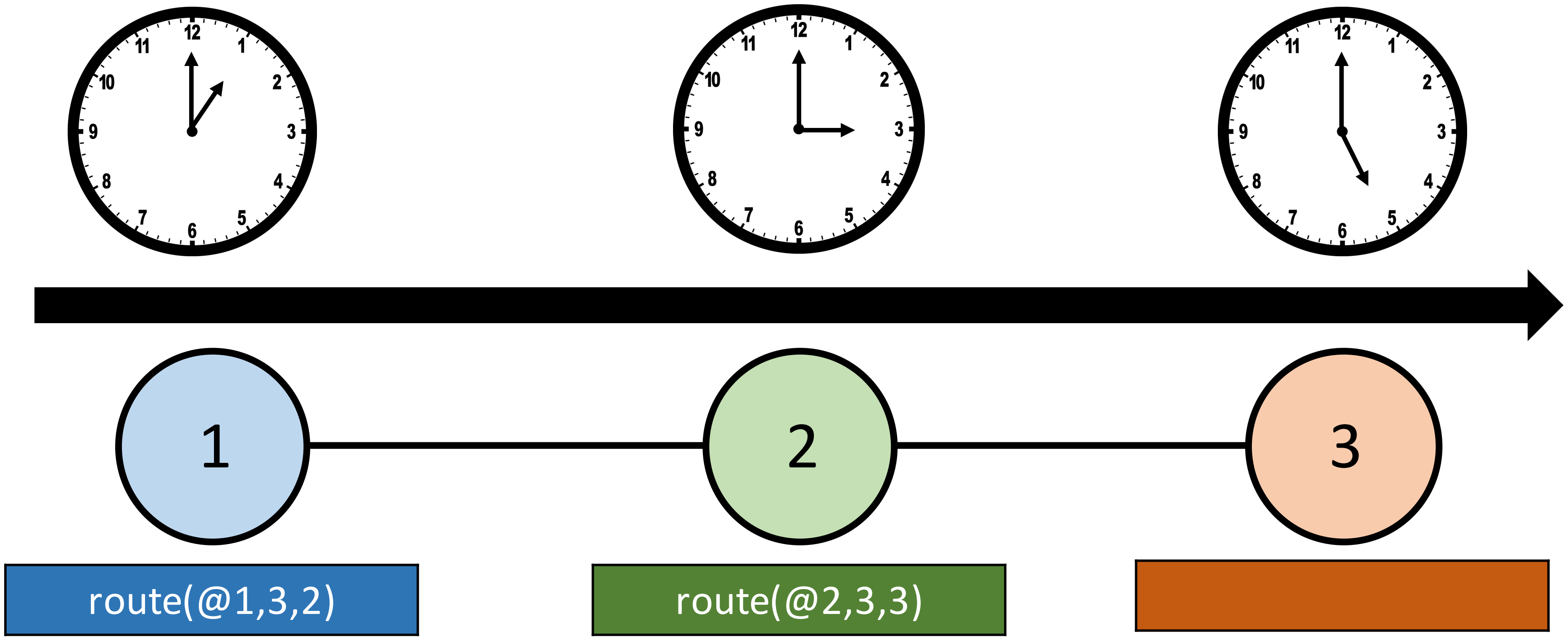
<result> :- <condition$_1$>, $\underbrace{\text{<condition}_2\text{>, … , <condition}_N\text{>}}_{\text{Slow-changing}}$

Fast-changing

```
Packet Forwarding
r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).
r2 recv(@L,S,D,T)     :- packet(@L,S,D,T),  D==L.
```

Simplifying Assumption #2

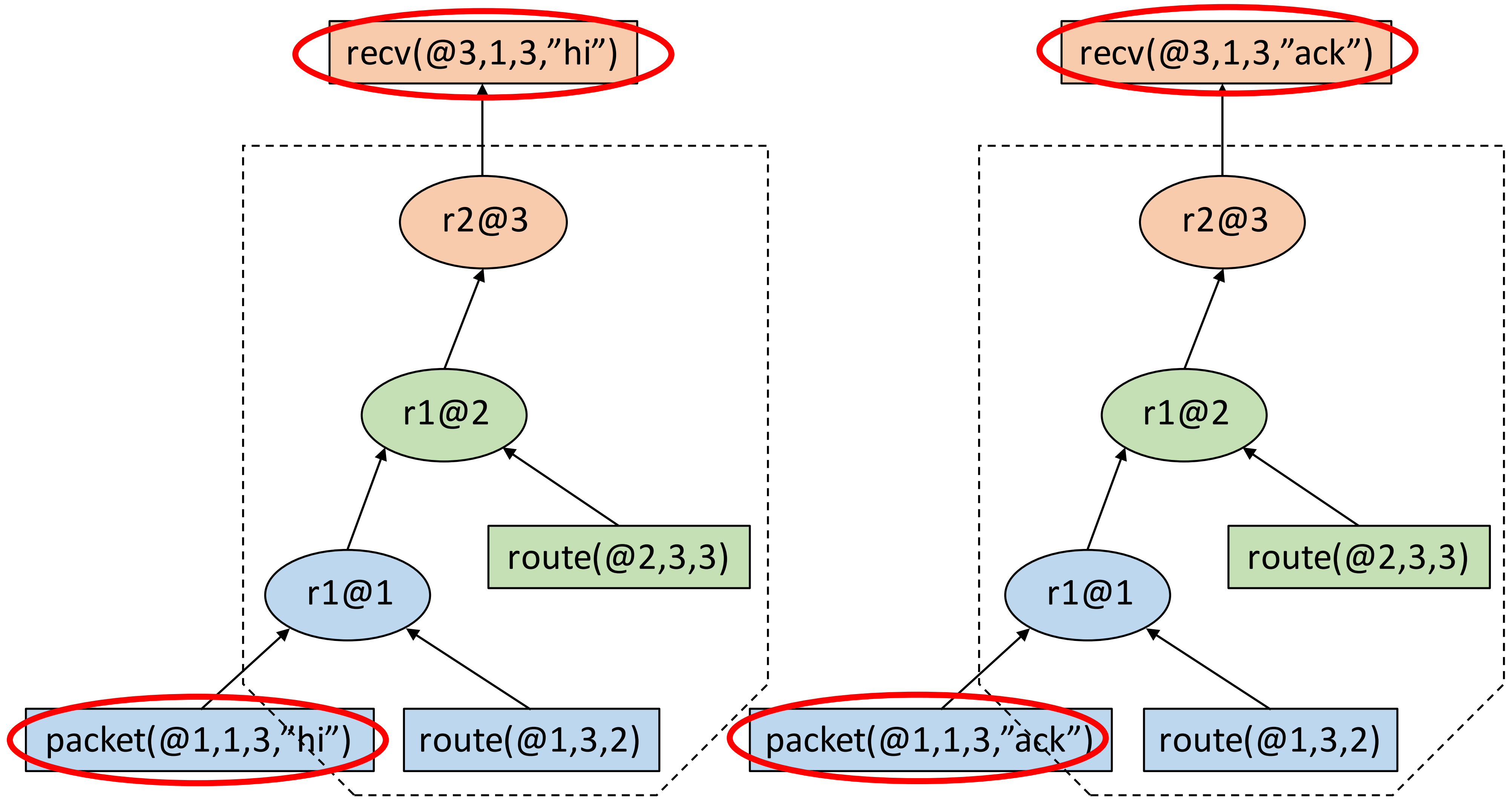
The set of slow changing tuples is ***constant***



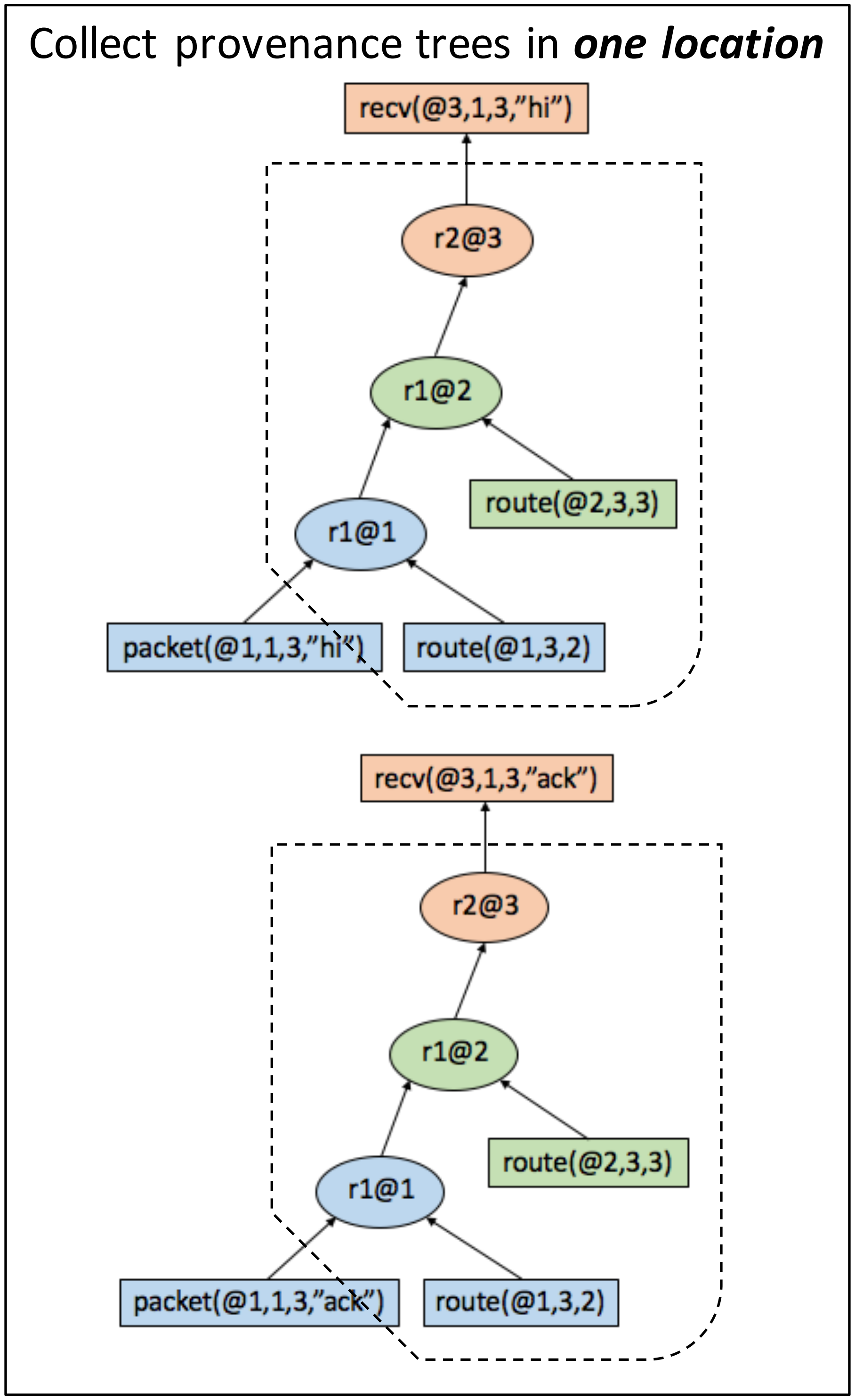
Roadmap

- Background
- Key insights
- ***Our compression scheme***
- Conclusion

Key Idea: Group provenance trees into *Equivalence Classes*



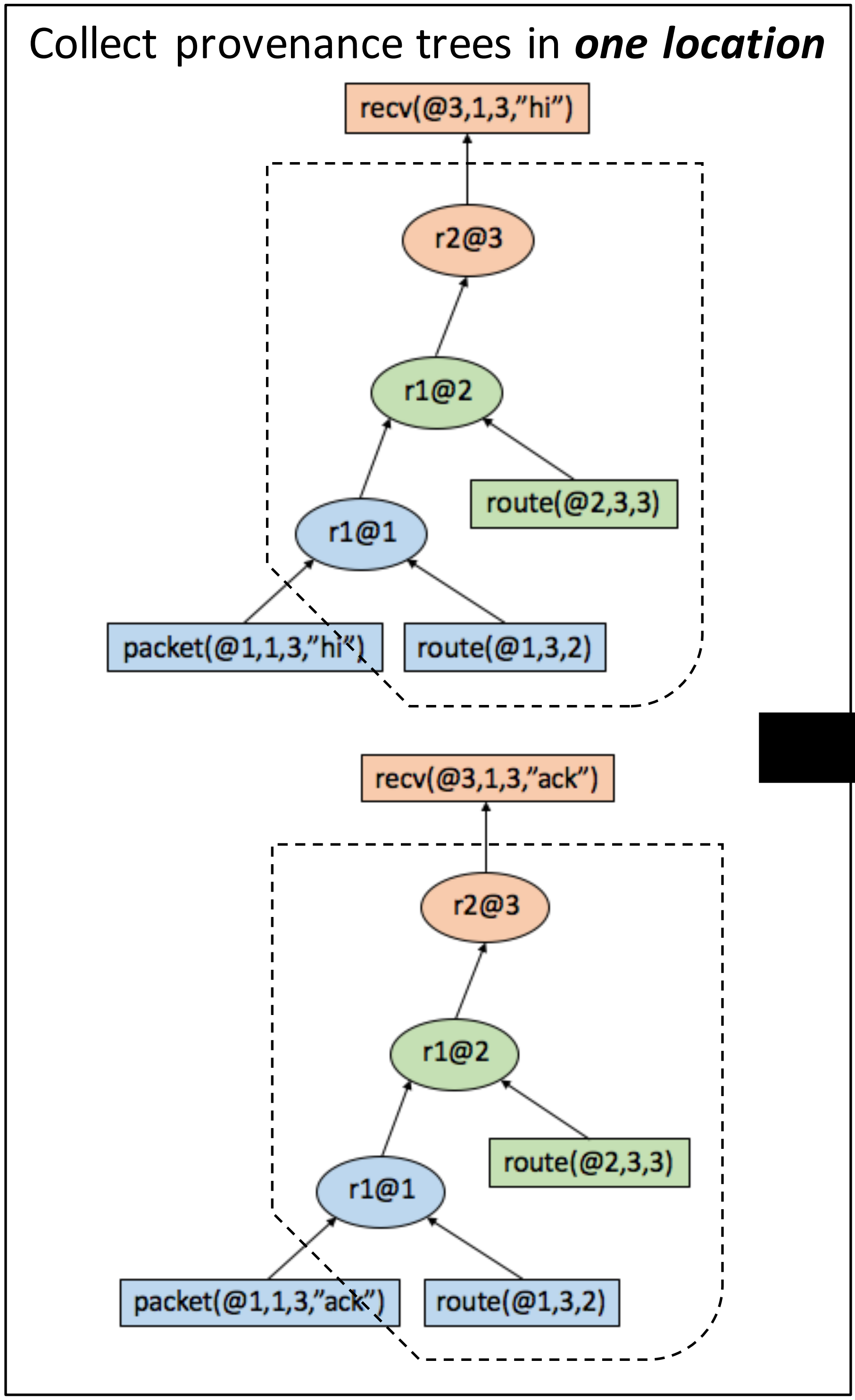
First attempt at compression

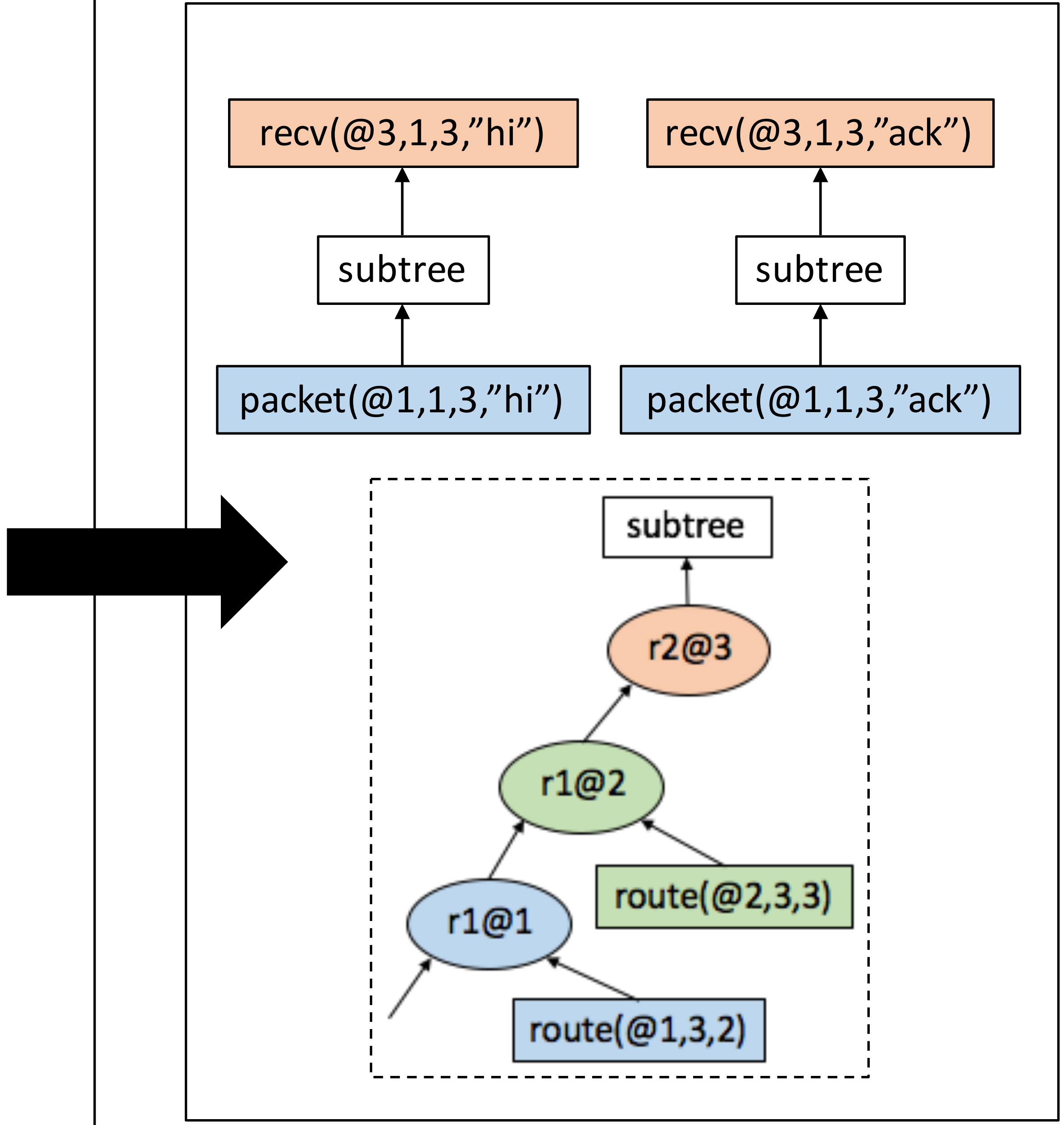


First attempt at compression

Chapman et al., *Efficient Provenance Storage*. SIGMOD'08

Collect provenance trees in ***one location***

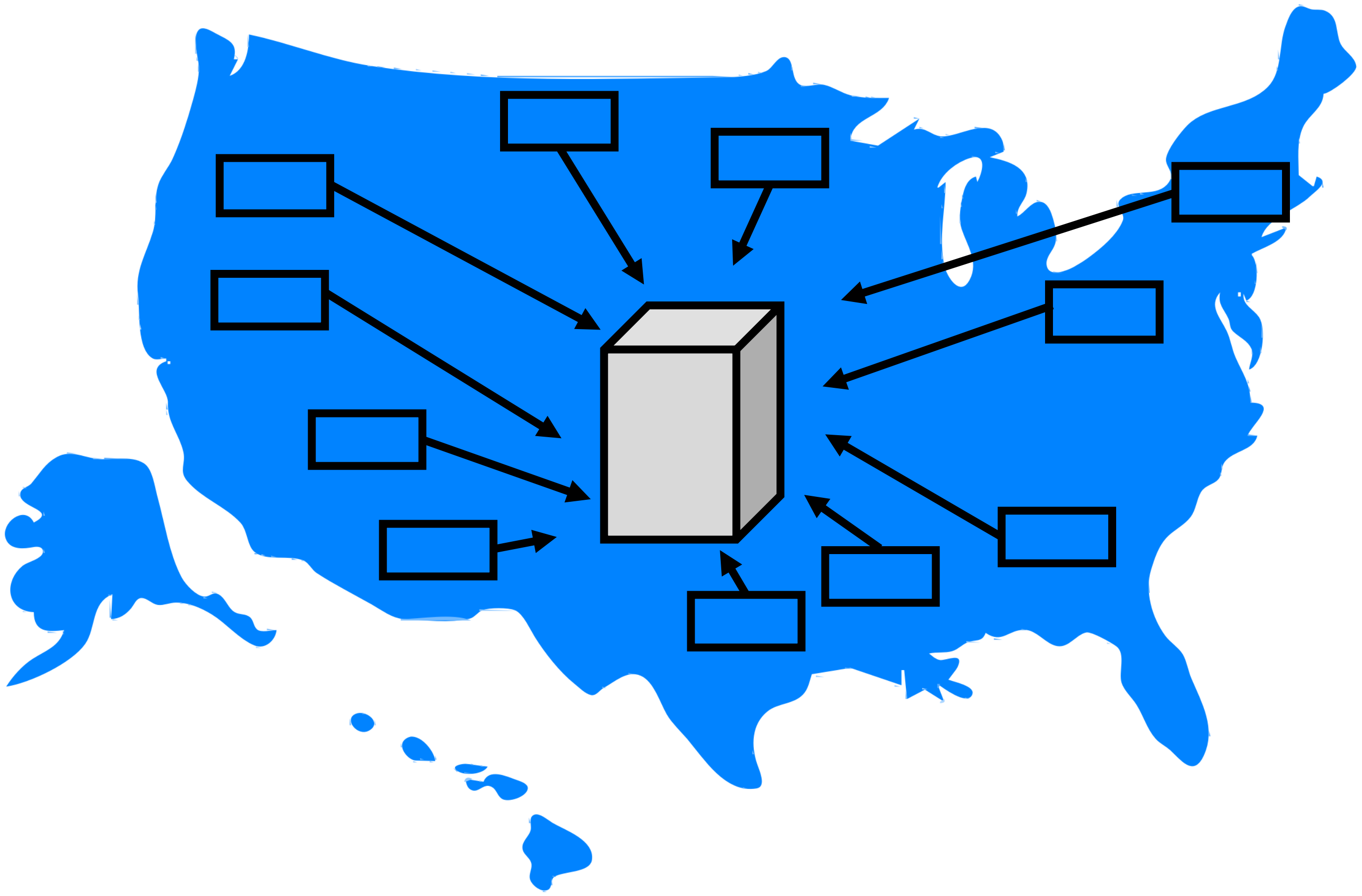




Zhou et al., *Efficient Querying & Maintenance of Network Provenance at Internet-Scale*. SIGMOD'10

Chapman et al., *Efficient Provenance Storage*. SIGMOD'08

Why not collect provenance in a centralized server?

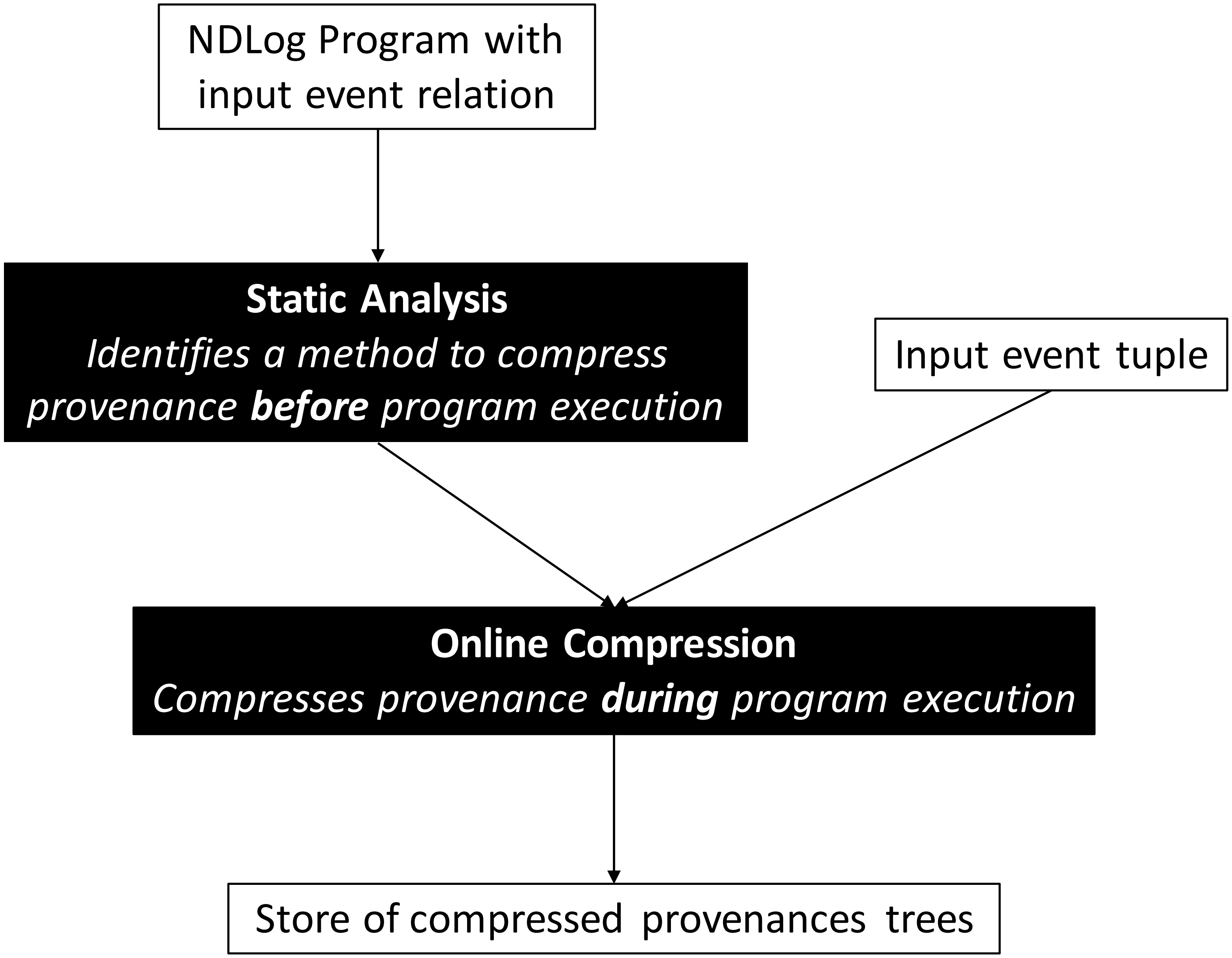


1. Bottleneck at the centralized server
2. High bandwidth utilization
3. Feasibility – entities in the network may span large geographic locations

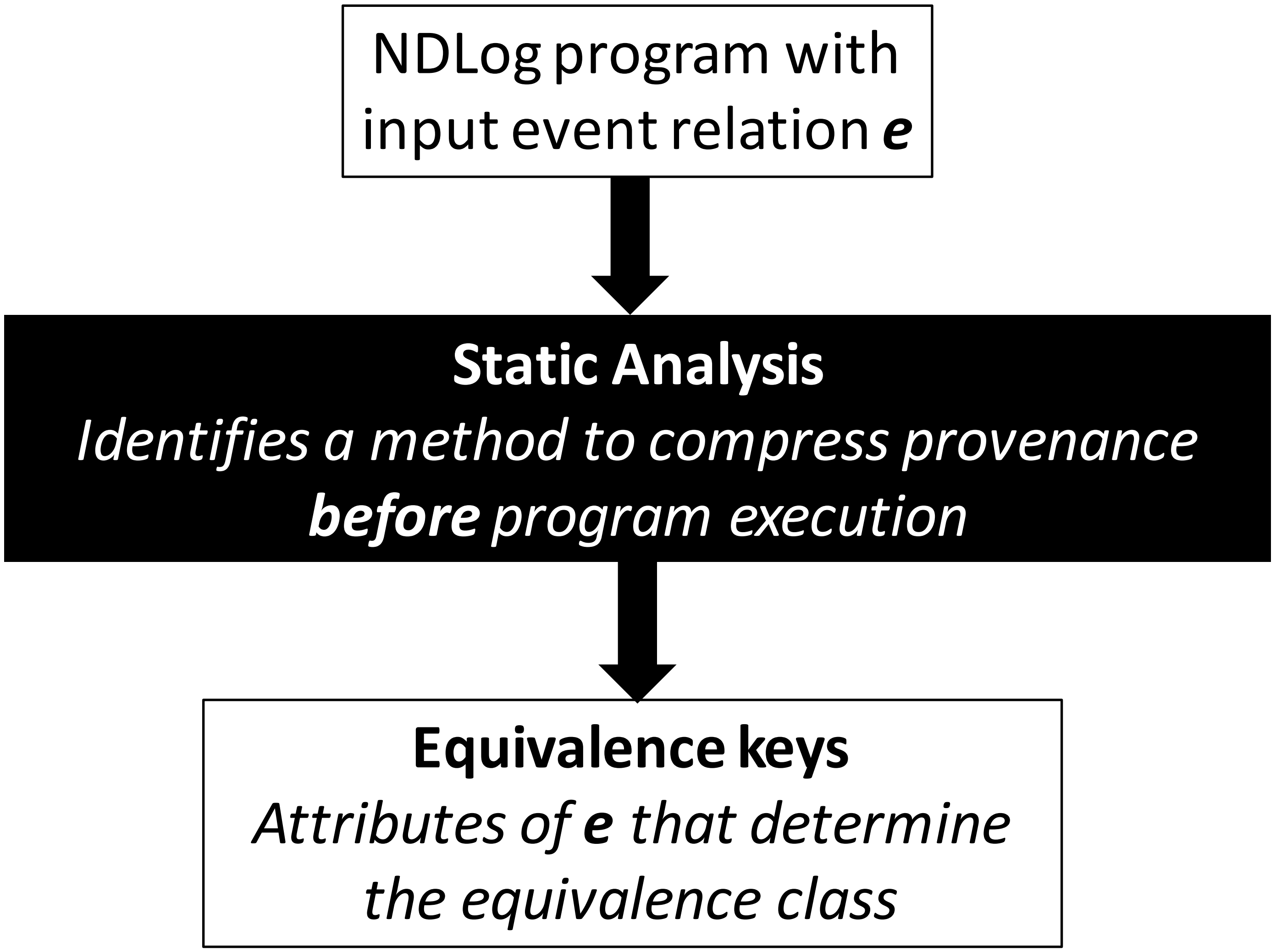
Our Compression Scheme

- Identifies how to share storage ***before*** program execution.
- Stores compressed provenances in a ***distributed*** setting ***at runtime***.

Workflow for our Compression Scheme


NDLog Program with input event relation
Static Analysis
*Identifies a method to compress provenance **before** program execution*
Input event tuple
Online Compression
*Compresses provenance **during** program execution*
Store of compressed provenances trees

Workflow of Static Analysis



NDLog program with input event relation ***e***

Static Analysis

Identifies a method to compress provenance ***before*** *program execution*

Equivalence keys

Attributes of **e** *that determine the equivalence class*

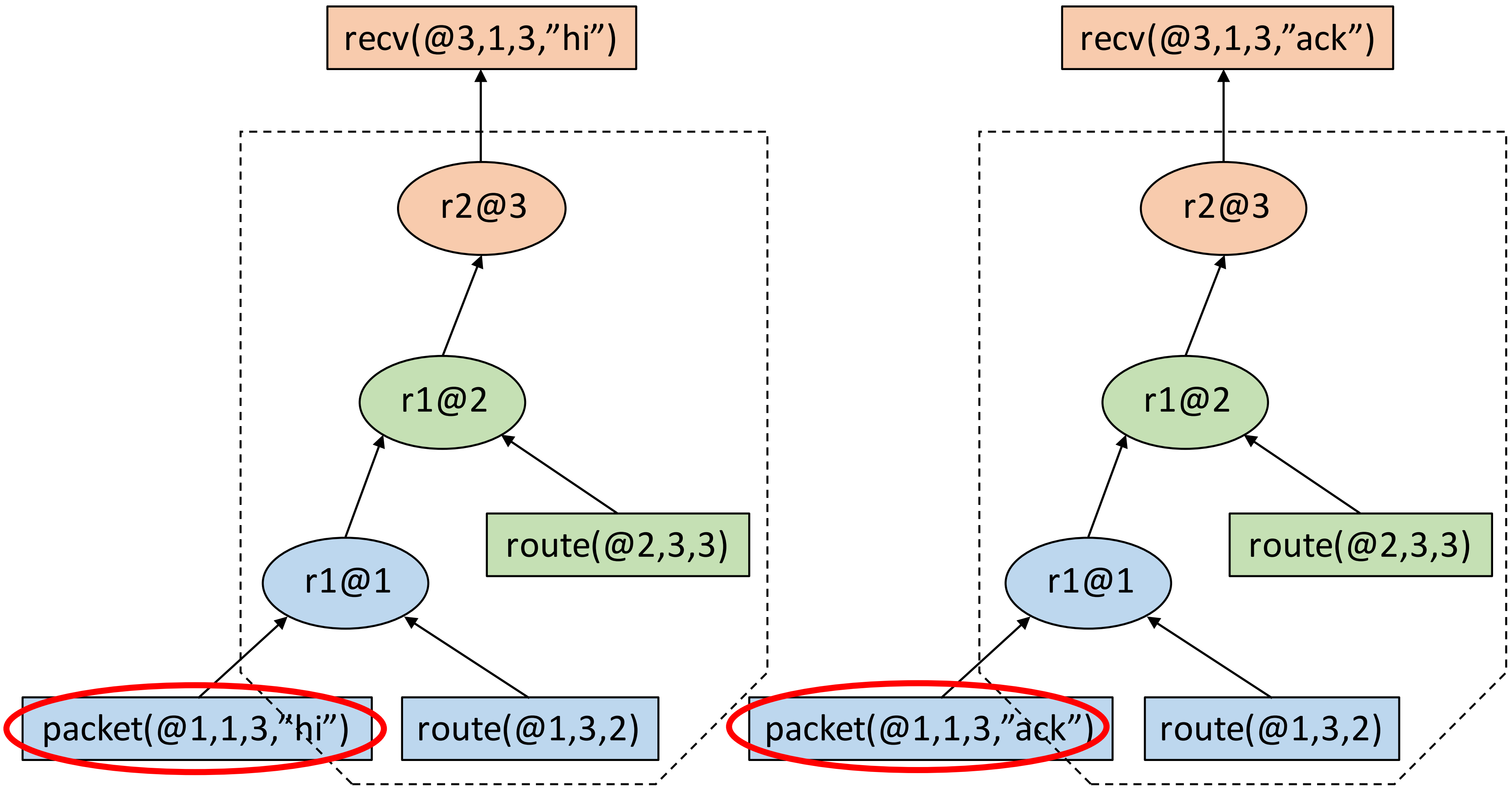
Which attributes of ***packet*** affect the provenance tree generated?

Packet Forwarding

```
r1 packet(@N,S,D,T)  :- packet(@L,S,D,T), route(@L,D,N).
r2 recv(@L,S,D,T)       :- packet(@L,S,D,T),   D==L.
```

packet(@1,1,3,"hi")

Provenance trees in the same equivalence class

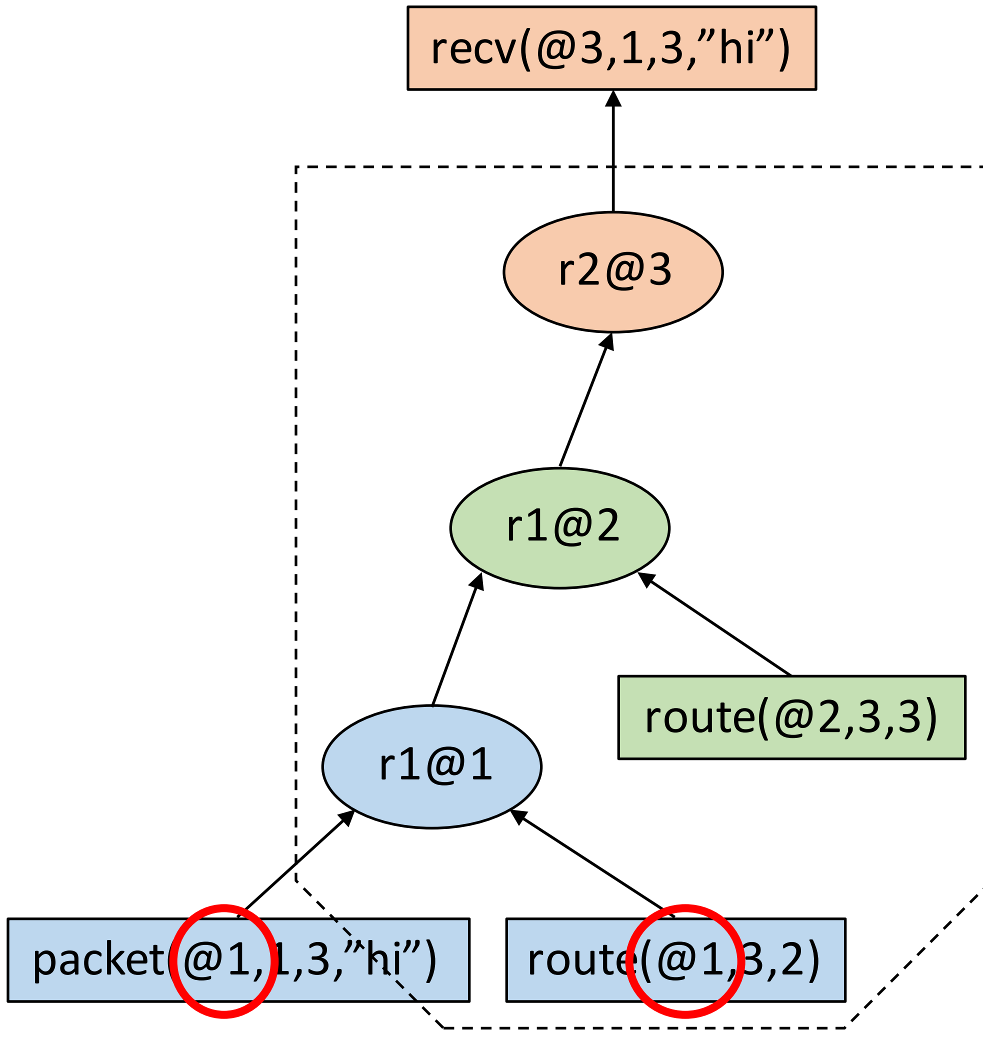


Identifying an equivalence key

Packet Forwarding

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.

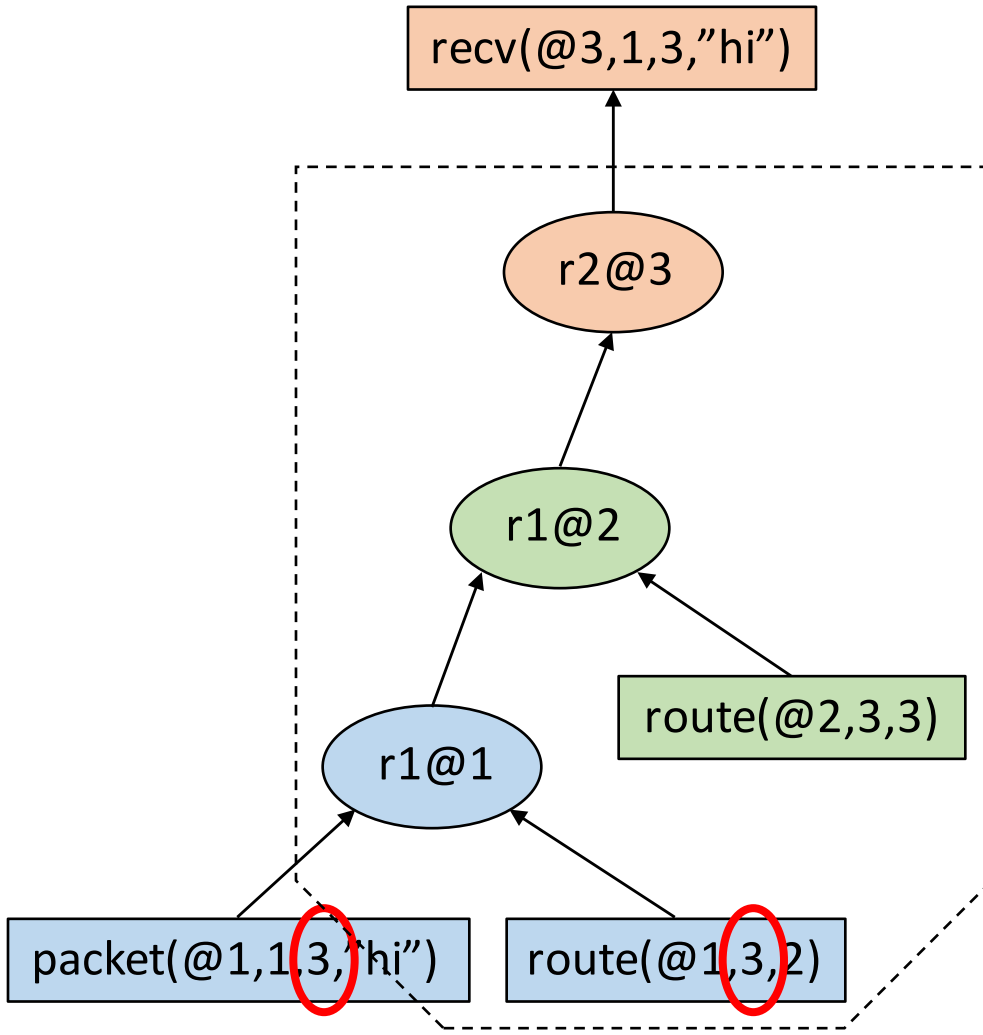


Identifying an equivalence key

Packet Forwarding

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.

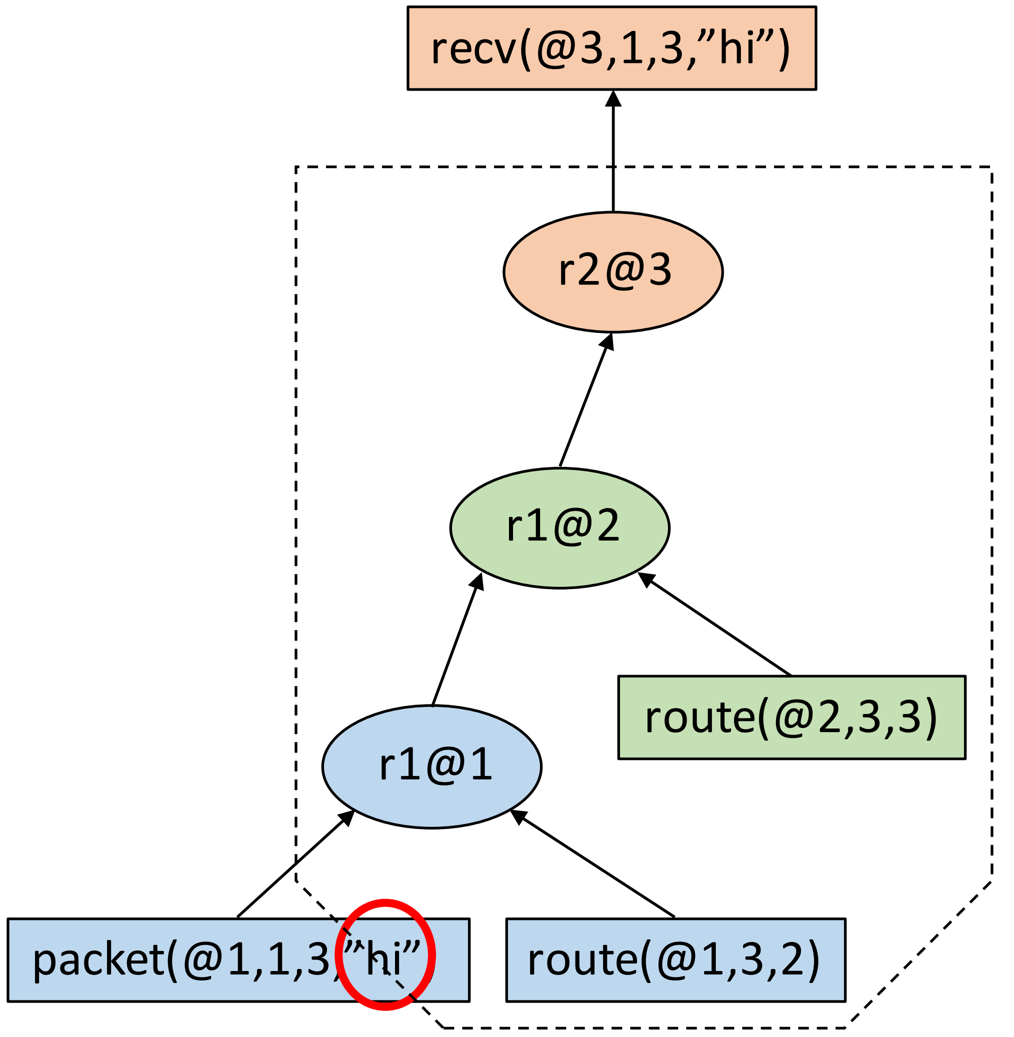


Not an equivalence key

Packet Forwarding

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.



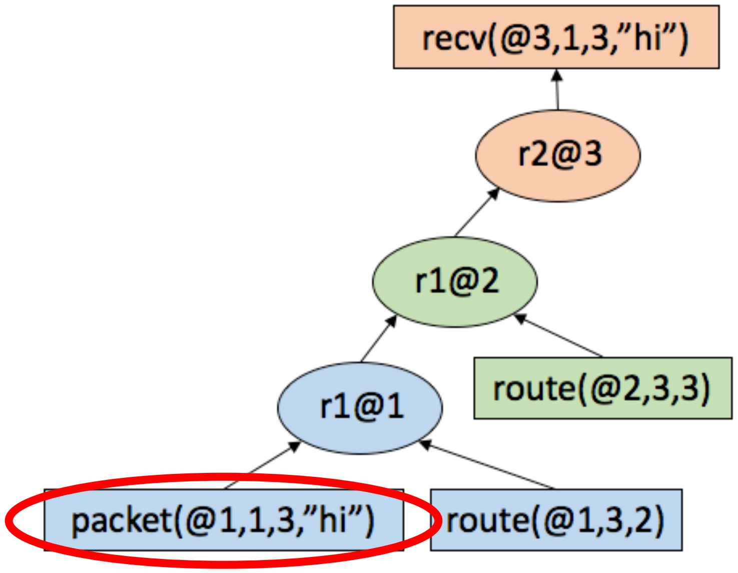
Correctness of Static Analysis

NDLog program with input event relation
Packet Forwarding

Equivalence keys:
packet(@L,*,S,*)

Instances of the input event
packet(@1,1,3,"hi") ~
packet(@1,1,3,"ack")

Slow-changing tuples
route(@2,3,3)
route(@1,3,2)



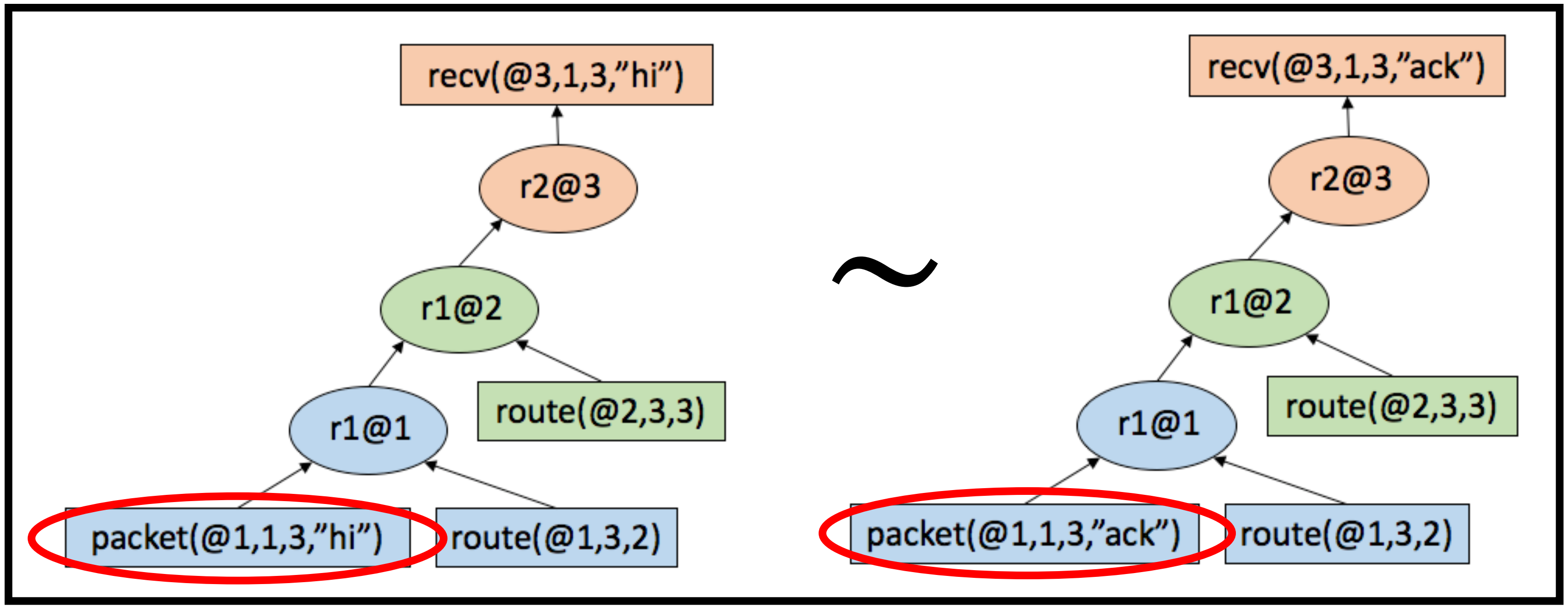
Correctness of Static Analysis

NDLog program with input event relation
Packet Forwarding

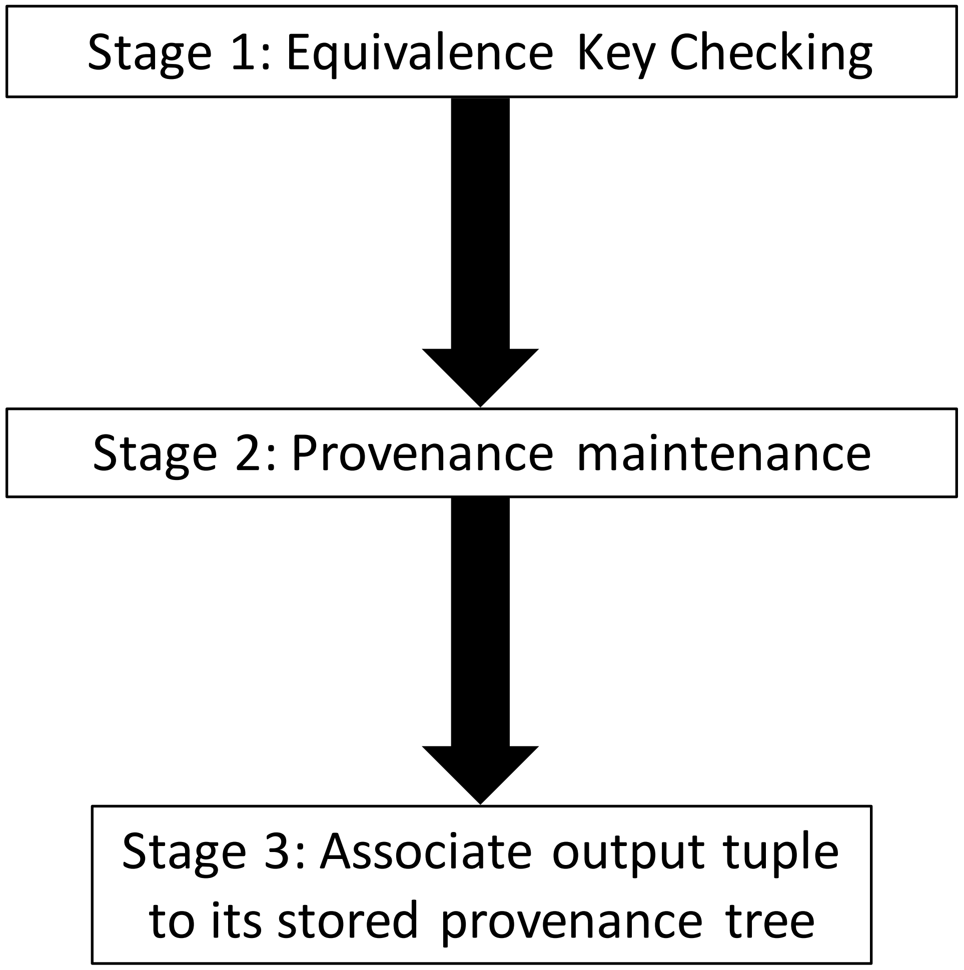
Equivalence keys:
packet(@L,*,S,*)

Instances of the input event
packet(@1,1,3,"hi") ~
packet(@1,1,3,"ack")

Slow-changing tuples
route(@2,3,3)
route(@1,3,2)



Workflow of Online Compression


Stage 1: Equivalence Key Checking
Stage 2: Provenance maintenance
Stage 3: Associate output tuple to its stored provenance tree

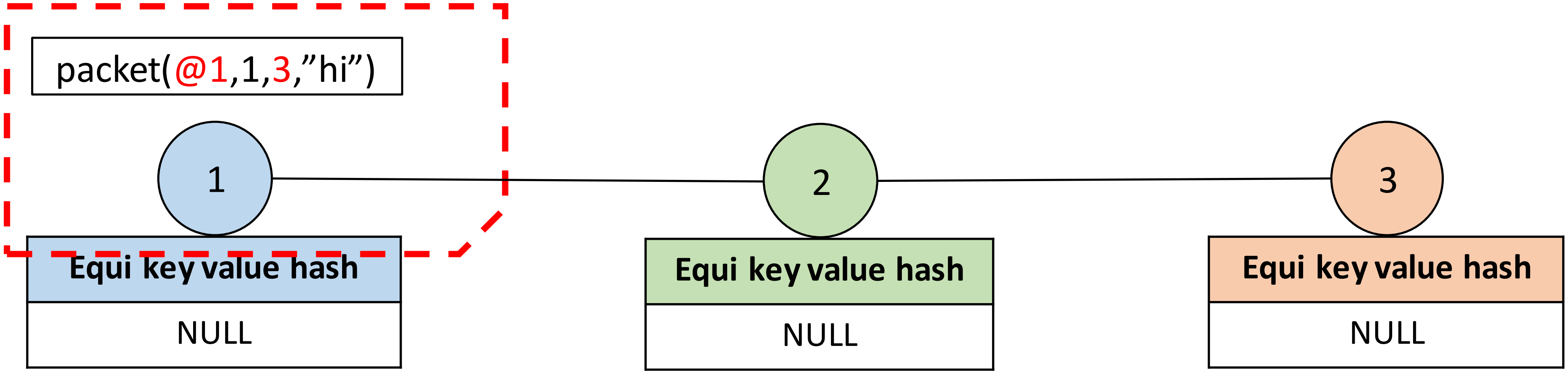
Creating a new provenance tree

Stage 1: Equivalence Key Checking

Packet Forwarding

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.

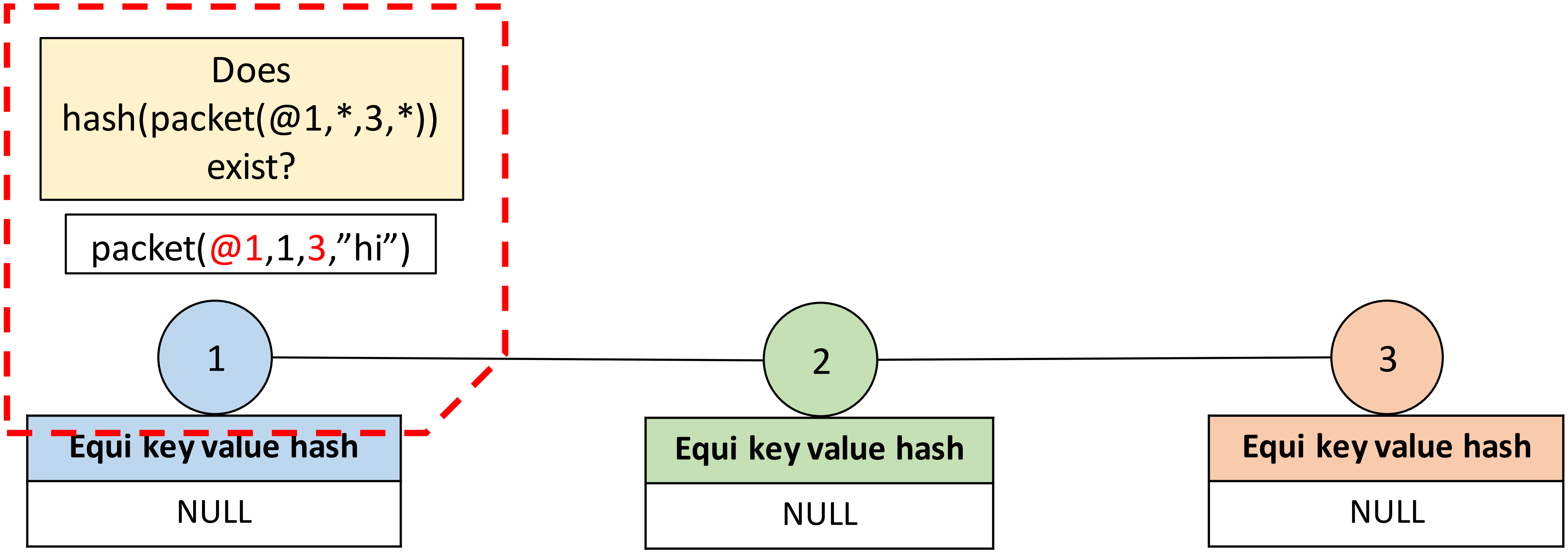


Creating a new provenance tree

Stage 1: Equivalence Key Checking

<u>Packet Forwarding</u>

```
r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).
r2 recv(@L,S,D,T)   :- packet(@L,S,D,T),  D==L.
```

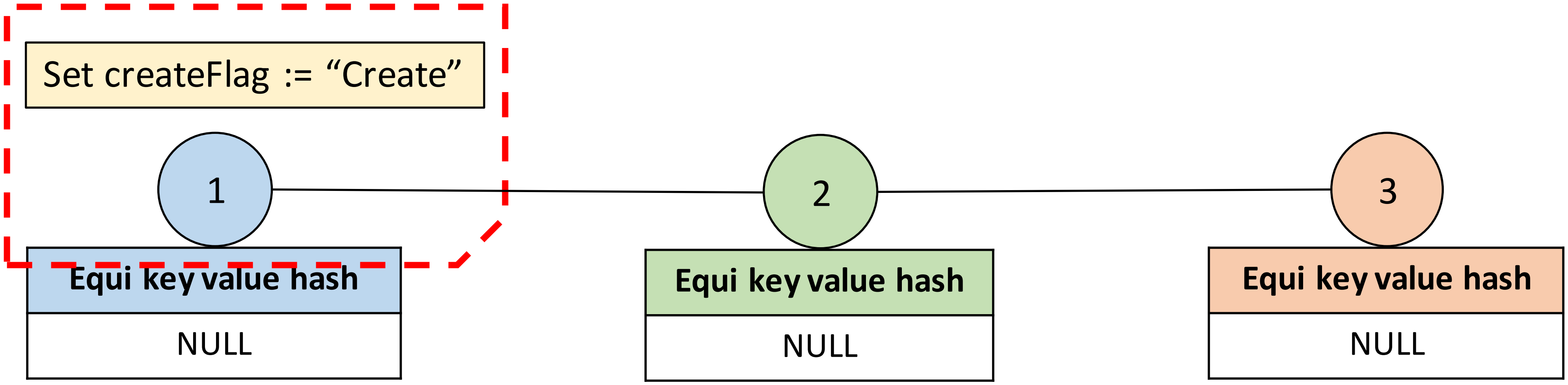


Creating a new provenance tree

Stage 1: Equivalence Key Checking

<u>Packet Forwarding</u>

```
r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).
r2 recv(@L,S,D,T)   :- packet(@L,S,D,T),  D==L.
```

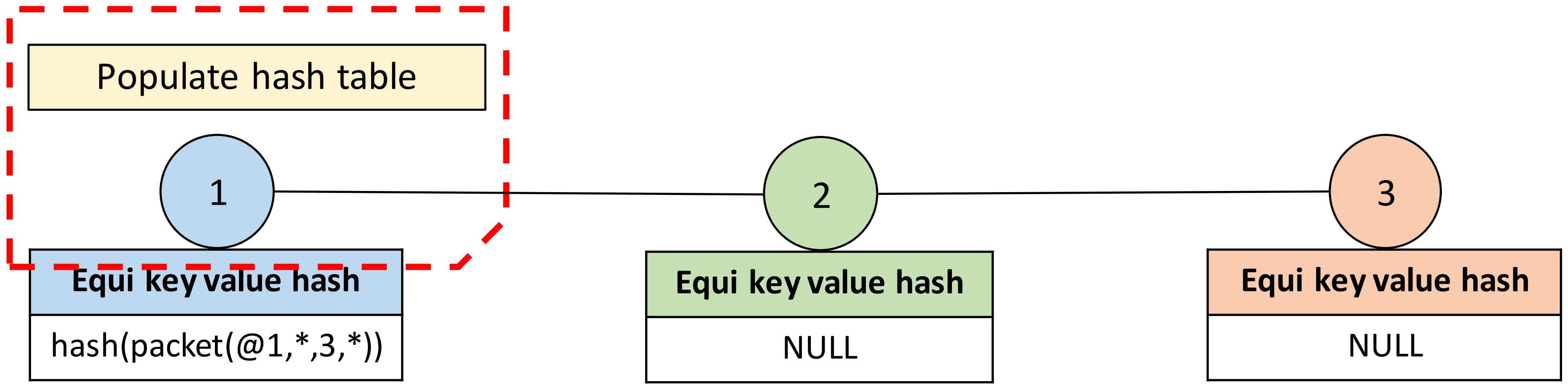


Creating a new provenance tree

Stage 1: Equivalence Key Checking

<u>Packet Forwarding</u>

```
r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).
r2 recv(@L,S,D,T)   :- packet(@L,S,D,T),  D==L.
```



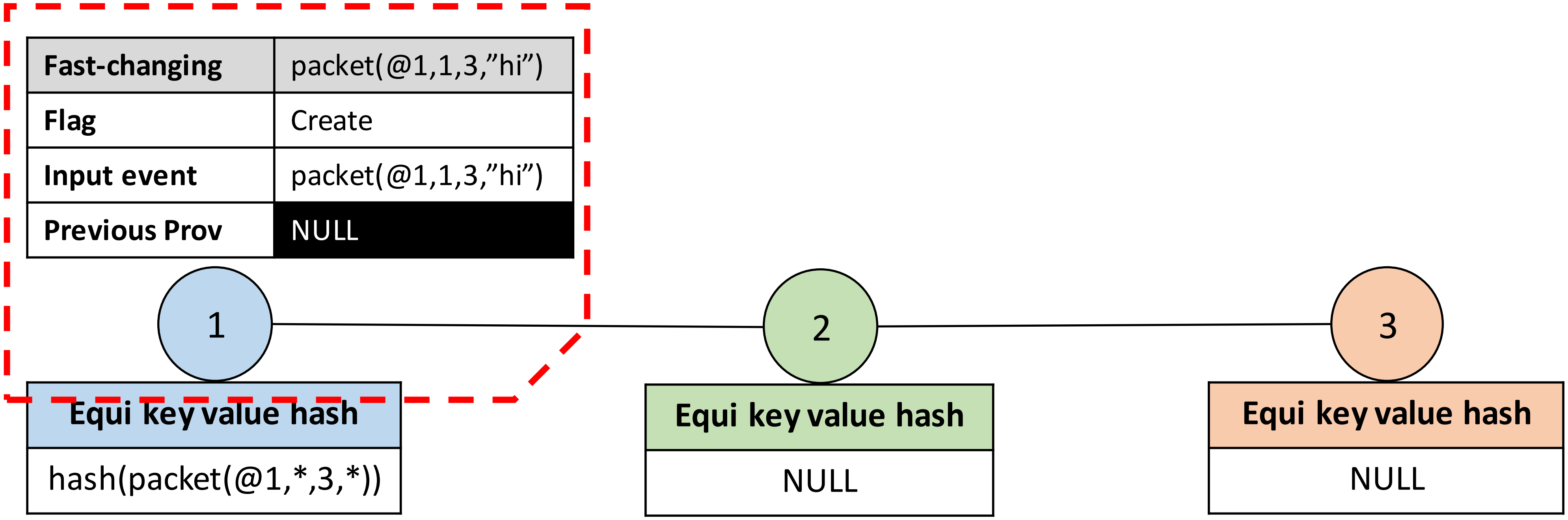
Creating a new provenance tree

Stage 1: Equivalence Key Checking

Packet Forwarding

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.



Fast-changing	packet(@1,1,3,"hi")
Flag	Create
Input event	packet(@1,1,3,"hi")
Previous Prov	NULL

Equi key value hash
hash(packet(@1,*,3,*))

Equi key value hash
NULL

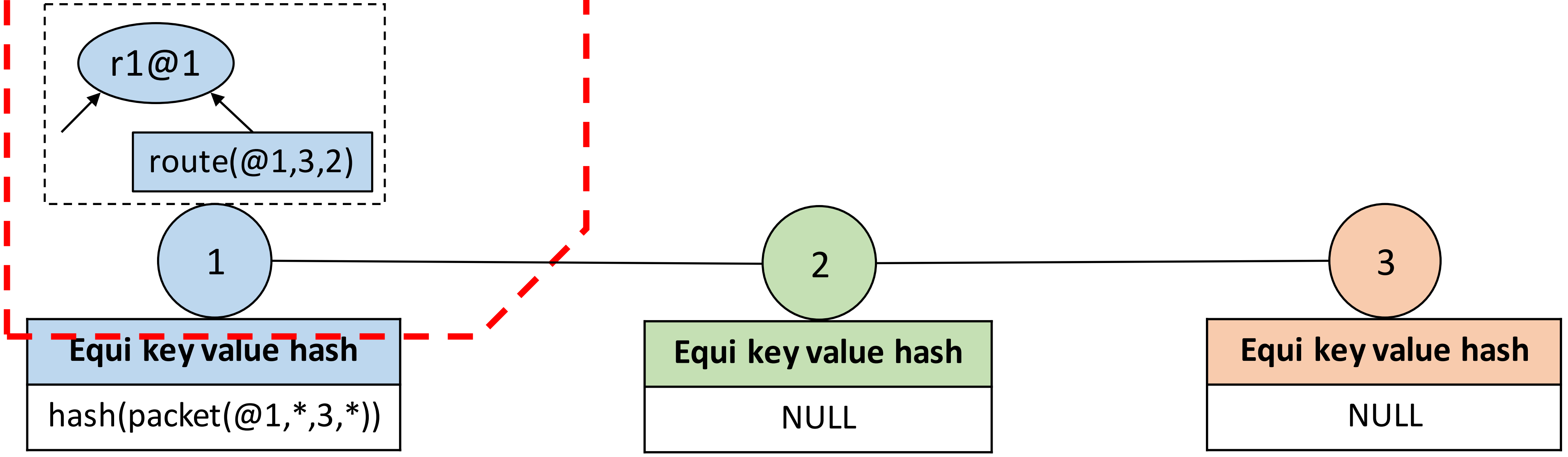
Equi key value hash
NULL

Creating a new provenance tree

Stage 2: Provenance maintenance

<u>Packet Forwarding</u>
r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).
r2 recv(@L,S,D,T)   :- packet(@L,S,D,T), D==L.

Fast-changing	packet(@1,1,3,"hi")
Flag	Create
Input event	packet(@1,1,3,"hi")
Previous Prov	NULL



Creating a new provenance tree

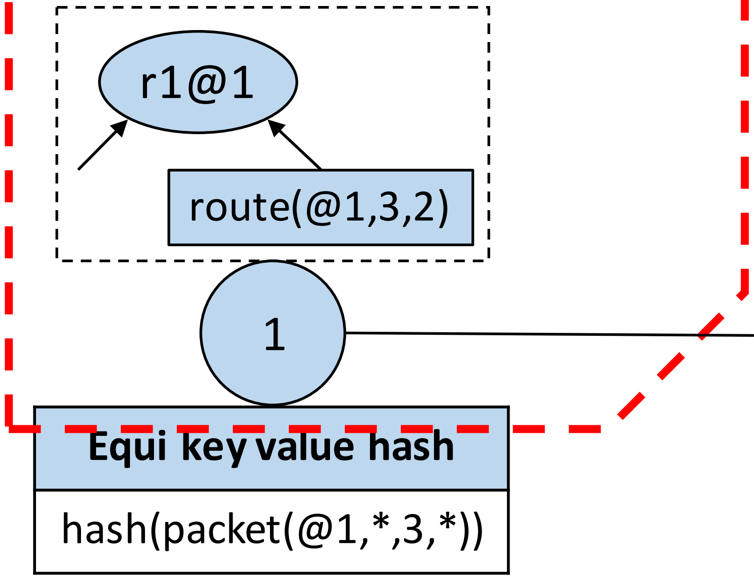
Stage 2: Provenance maintenance

Packet Forwarding

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.

Fast-changing	packet(@2,1,3,"hi")
Flag	Create
Input event	packet(@1,1,3,"hi")
Previous Prov	

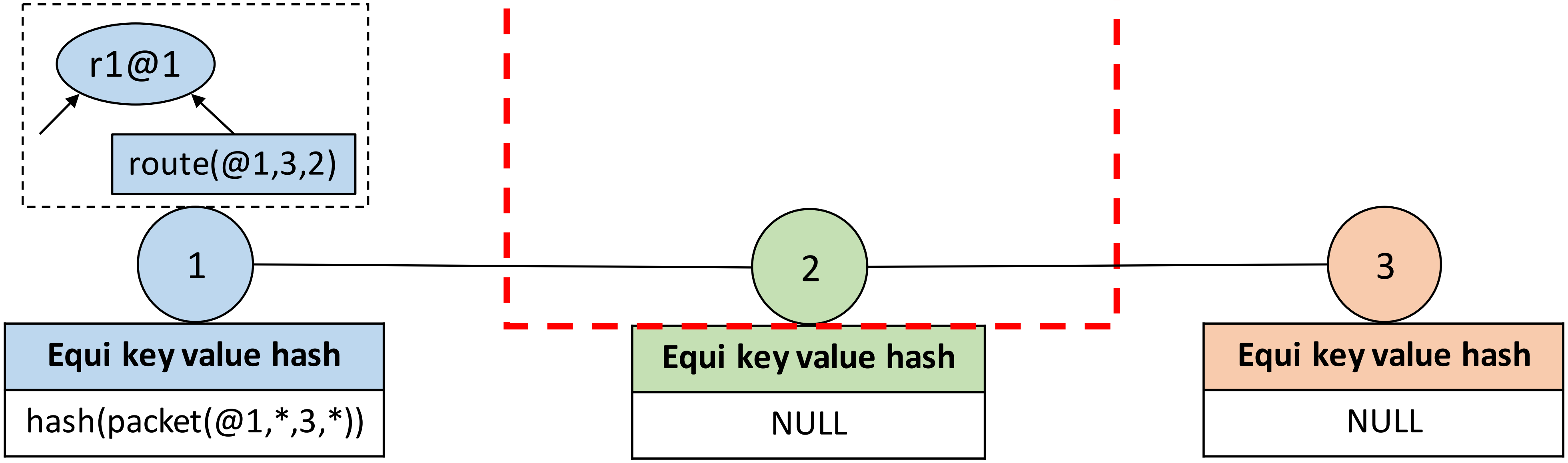


Creating a new provenance tree

Stage 2: Provenance maintenance

Packet Forwarding
r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).
r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.

Fast-changing	packet(@2,1,3,"hi")
Flag	Create
Input event	packet(@1,1,3,"hi")
Previous Prov	



Creating a new provenance tree

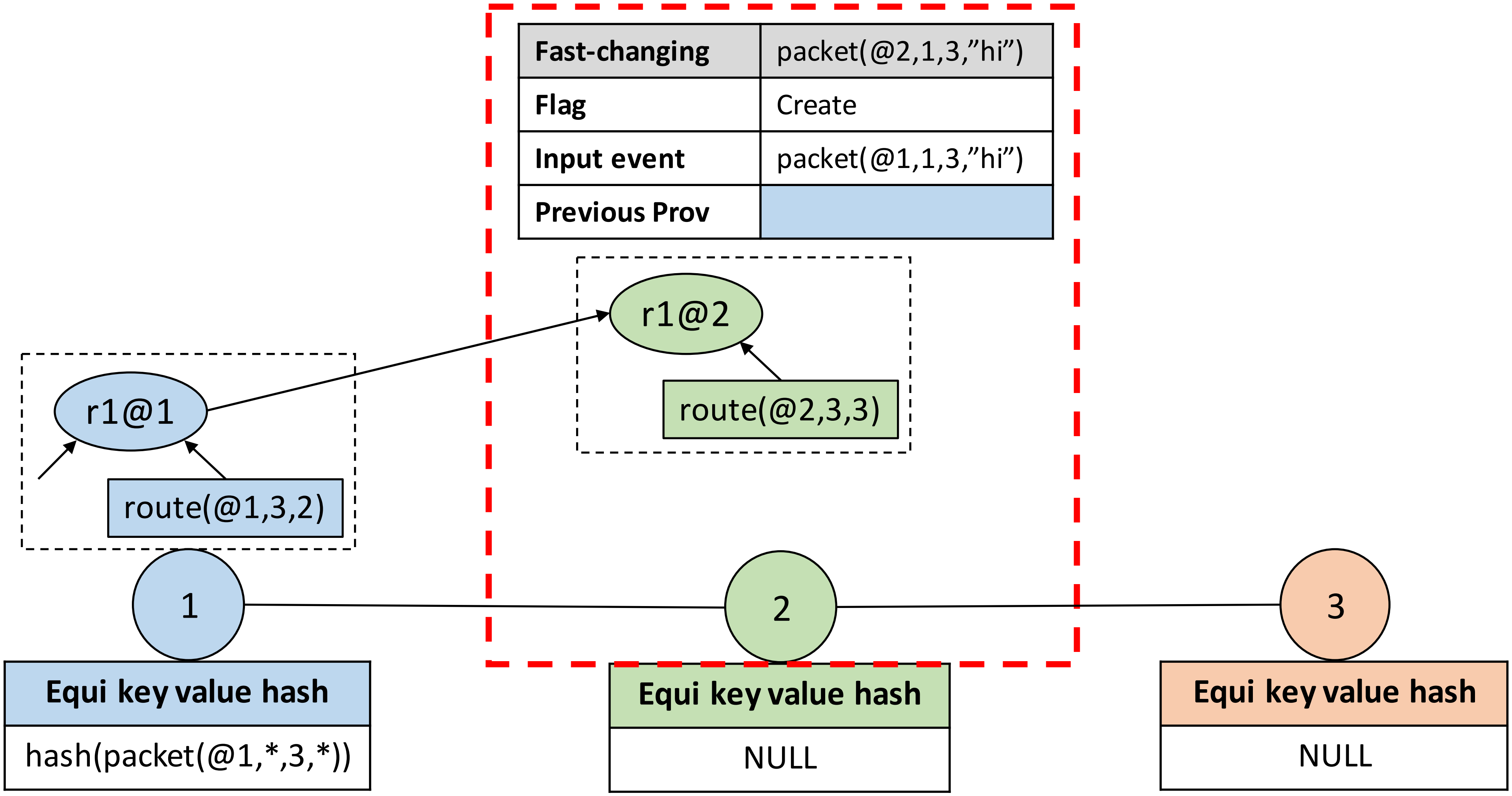
Stage 2: Provenance maintenance

Packet Forwarding

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.

Fast-changing	packet(@2,1,3,"hi")
Flag	Create
Input event	packet(@1,1,3,"hi")
Previous Prov	



Creating a new provenance tree

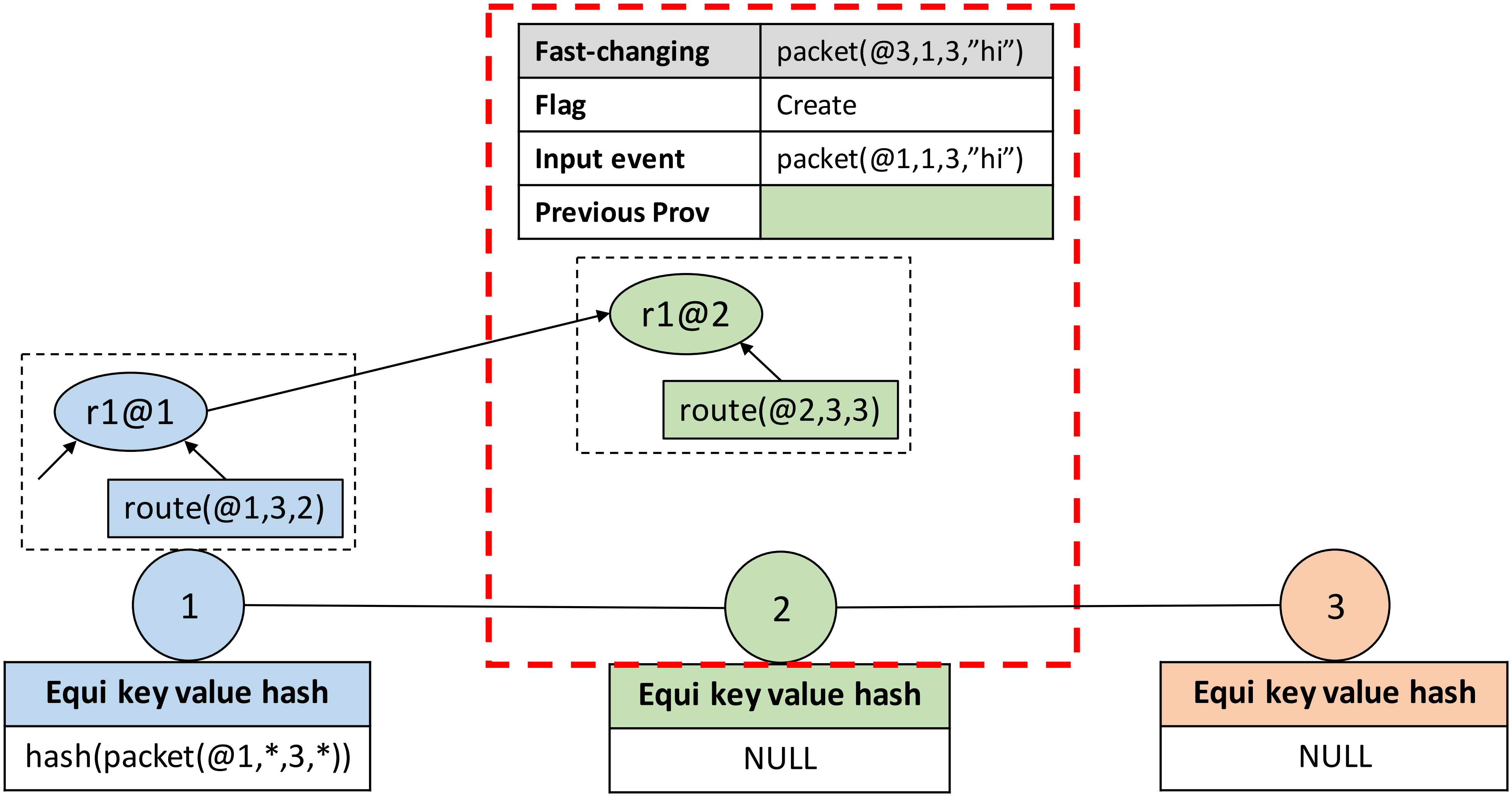
Stage 2: Provenance maintenance

Packet Forwarding

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.

Fast-changing	packet(@3,1,3,"hi")
Flag	Create
Input event	packet(@1,1,3,"hi")
Previous Prov	🟩

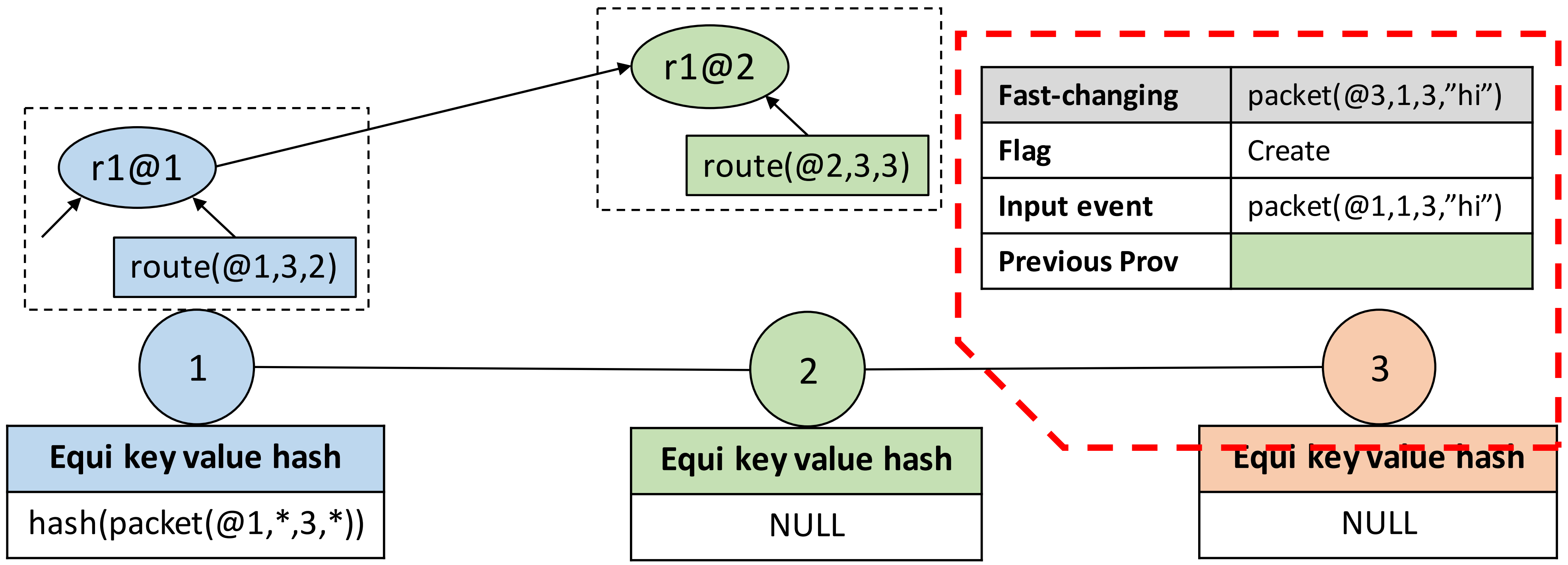


Creating a new provenance tree

Stage 2: Provenance maintenance

Packet Forwarding

```
r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).
r2 recv(@L,S,D,T)   :- packet(@L,S,D,T),  D==L.
```



Fast-changing	packet(@3,1,3,"hi")
Flag	Create
Input event	packet(@1,1,3,"hi")
Previous Prov	🟩

Equi key value hash
hash(packet(@1,*,3,*))

Equi key value hash
NULL

Equi key value hash
NULL

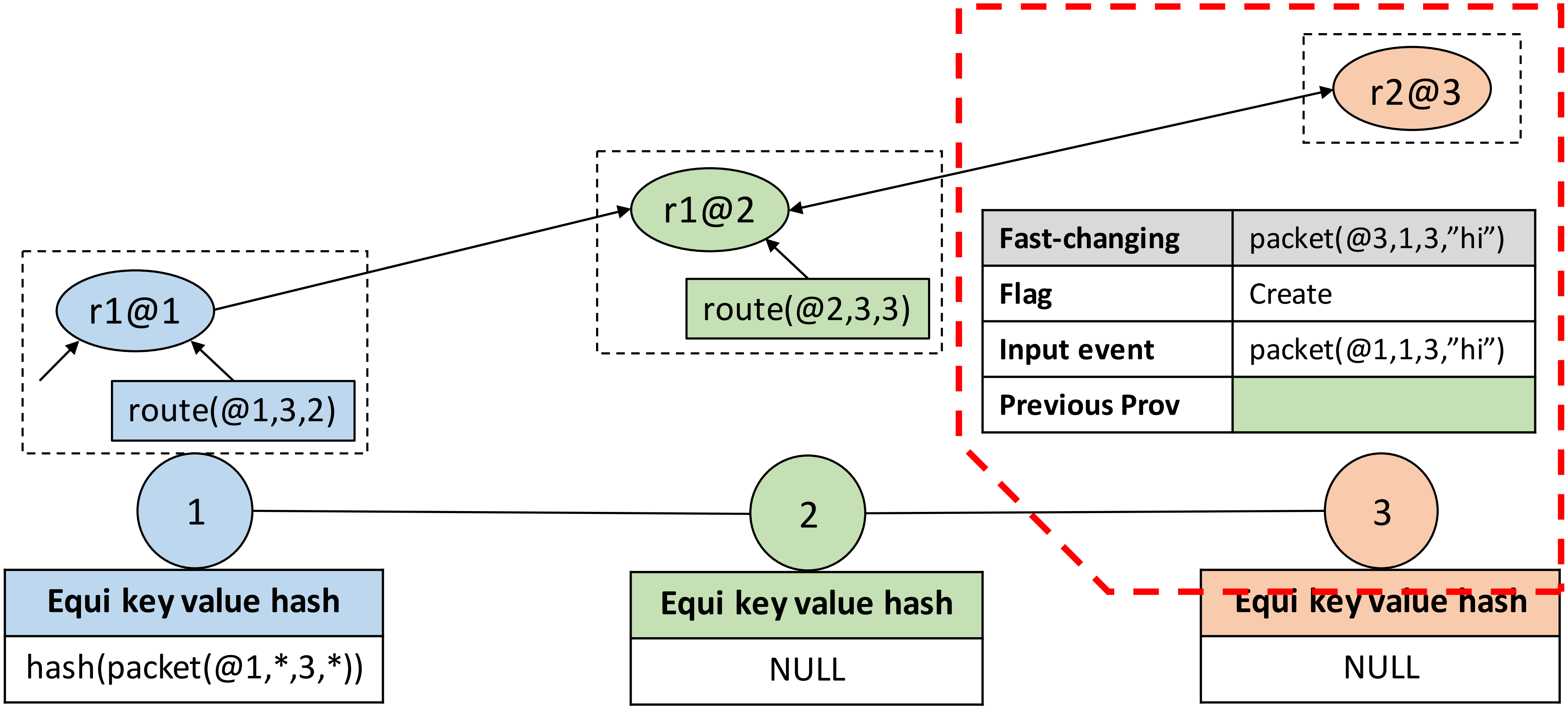
Creating a new provenance tree

Stage 2: Provenance maintenance

Packet Forwarding

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.



Fast-changing	packet(@3,1,3,"hi")
Flag	Create
Input event	packet(@1,1,3,"hi")
Previous Prov	🟩

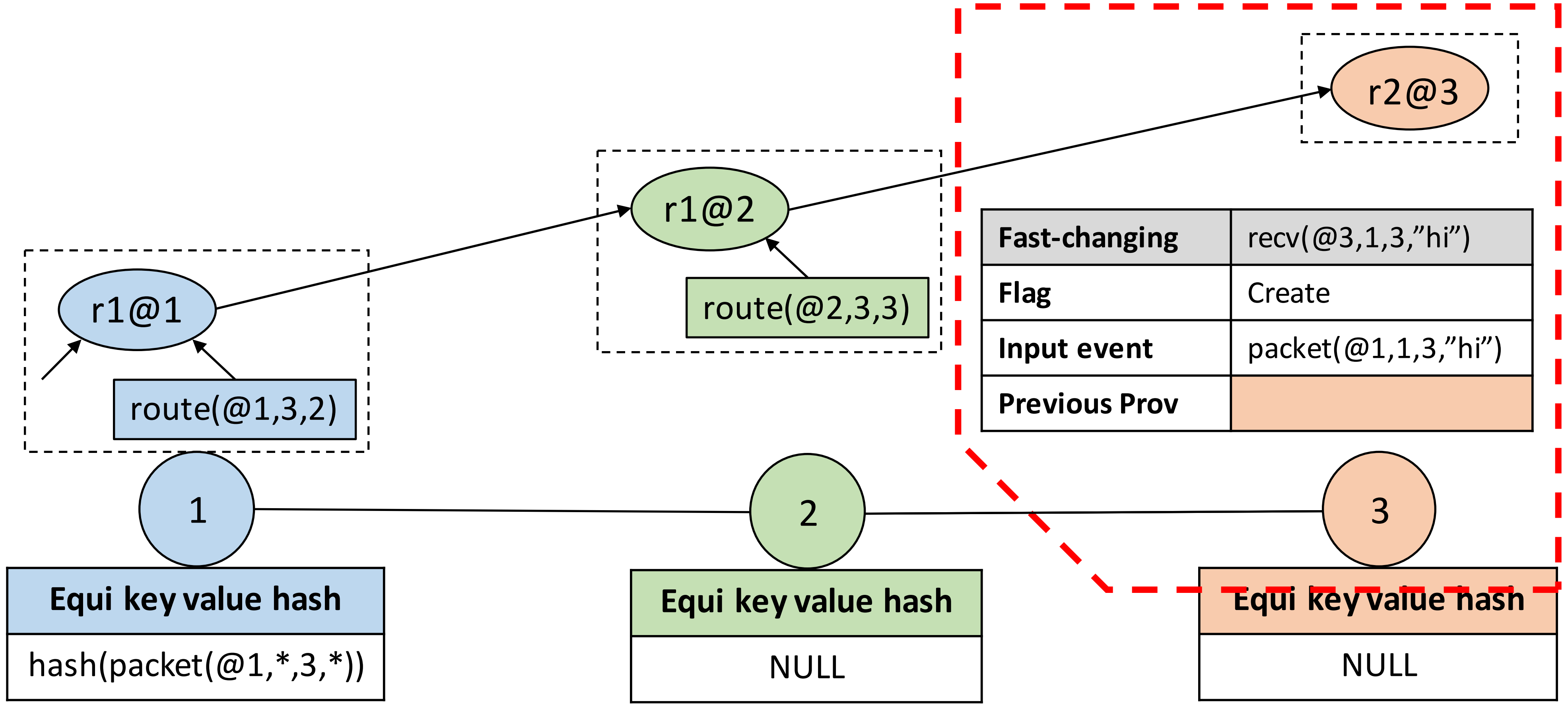
Creating a new provenance tree

Stage 3: Provenance Association

<u>Packet Forwarding</u>

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.



Fast-changing	recv(@3,1,3,"hi")
Flag	Create
Input event	packet(@1,1,3,"hi")
Previous Prov	

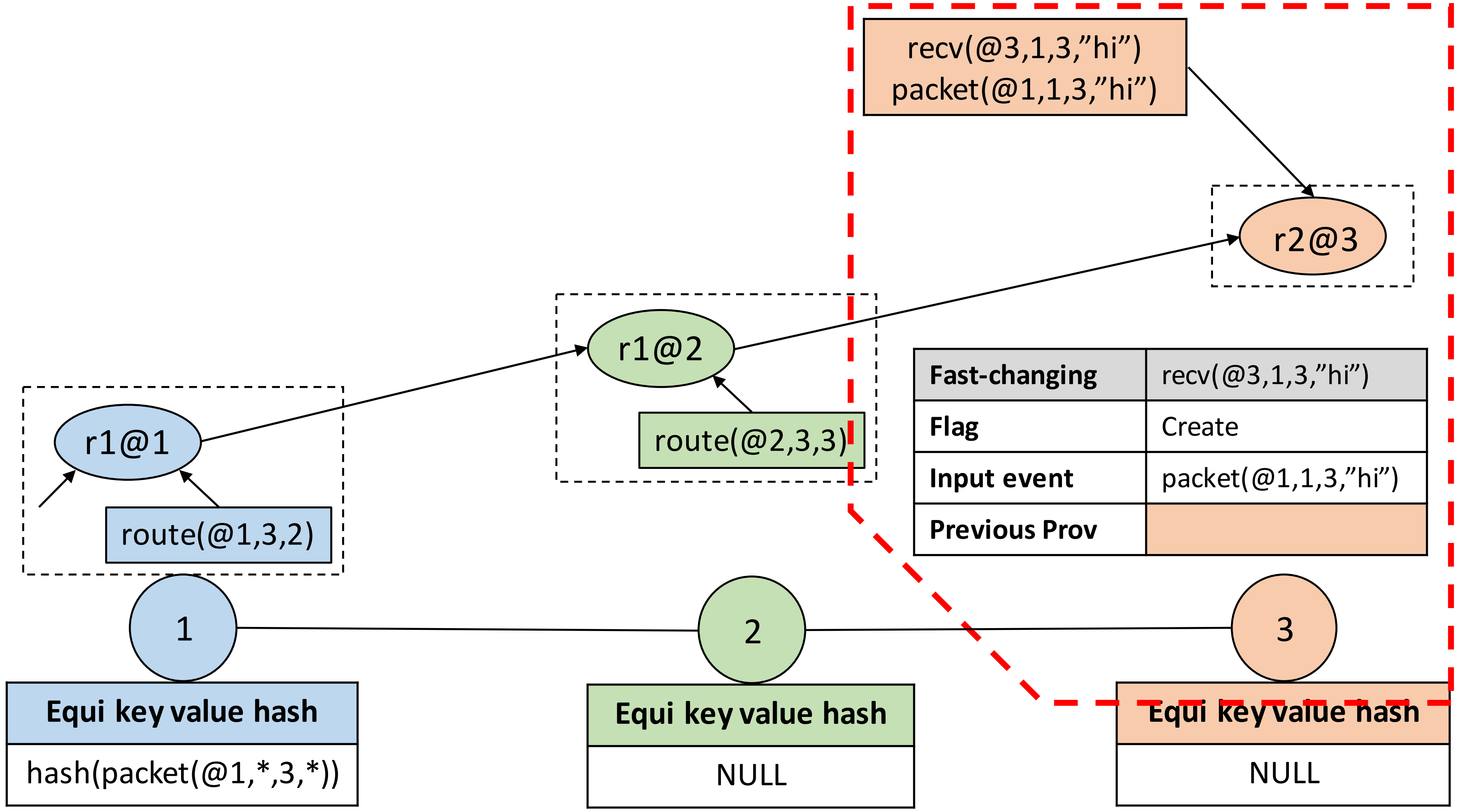
Creating a new provenance tree

Stage 3: Associate output tuple to its stored provenance tree

<u>Packet Forwarding</u>

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.



Fast-changing	recv(@3,1,3,"hi")
Flag	Create
Input event	packet(@1,1,3,"hi")
Previous Prov	

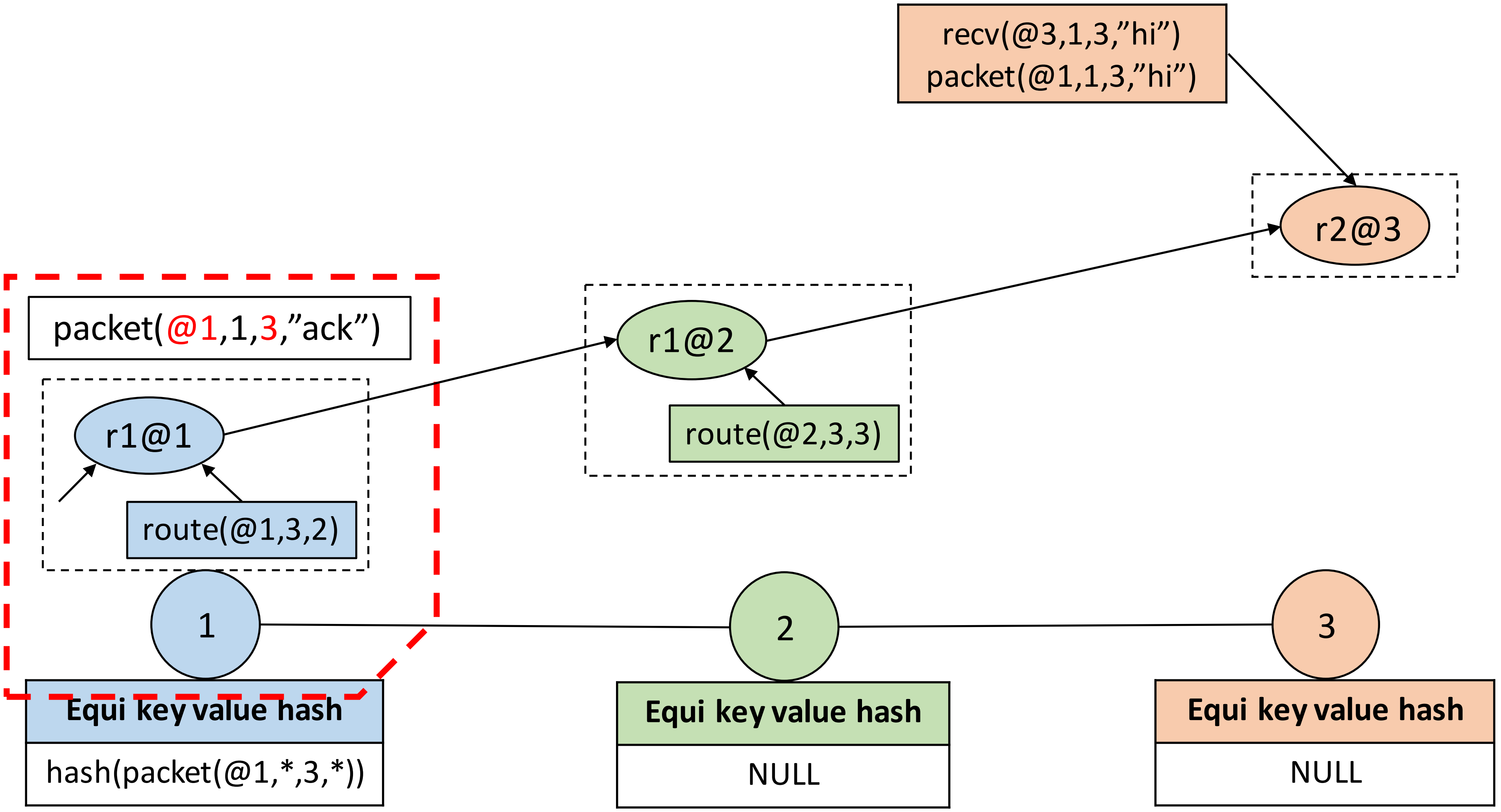
No provenance stored

Stage 1: Equivalence Key Checking

<u>Packet Forwarding</u>

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

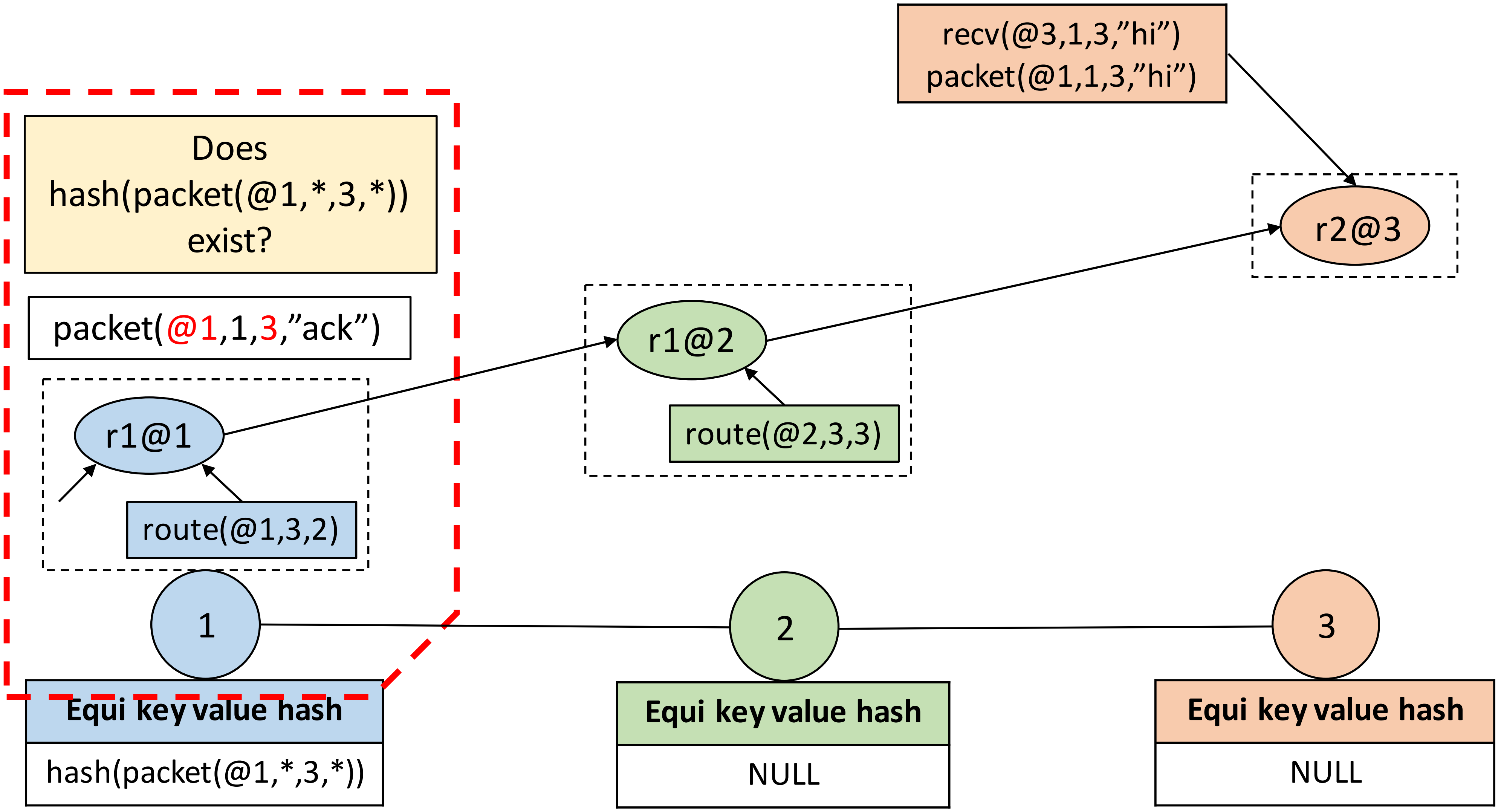
r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.



No provenance stored

Stage 1: Equivalence Key Checking

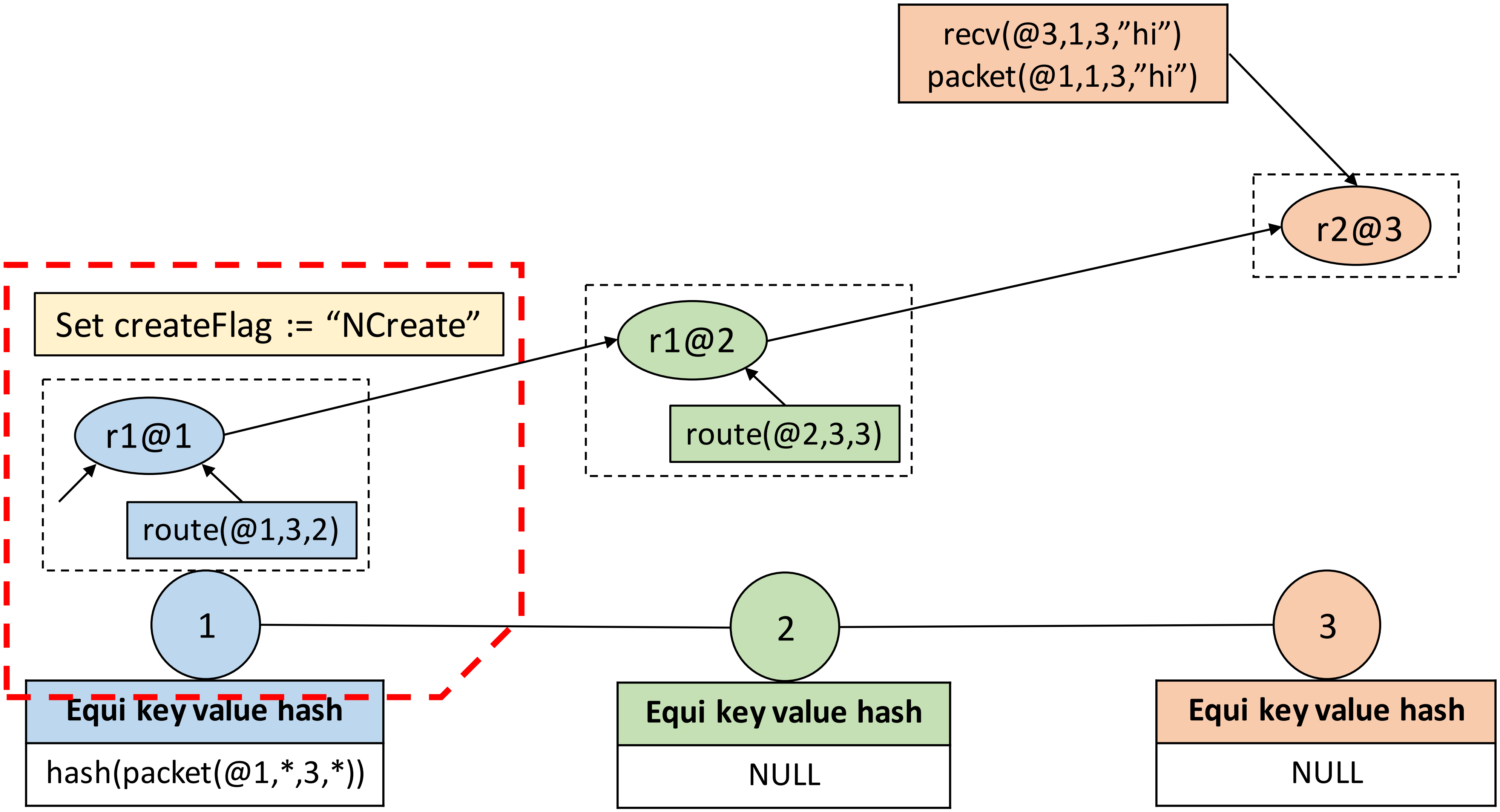
Packet Forwarding
r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).
r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.



No provenance stored

Stage 1: Equivalence Key Checking

Packet Forwarding
r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).
r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.



No provenance stored

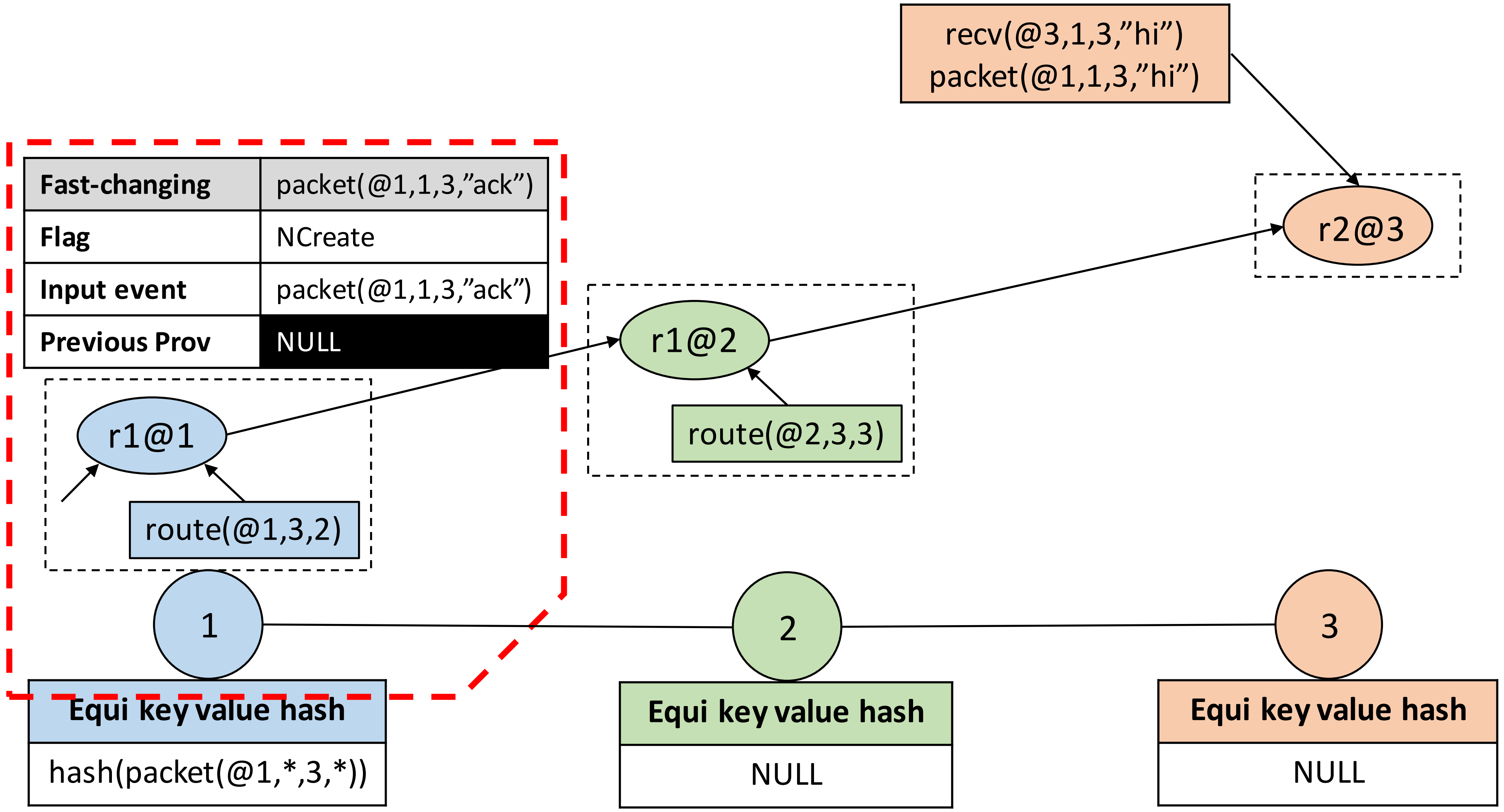
Stage 1: Equivalence Key Checking

Packet Forwarding

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.

Fast-changing	packet(@1,1,3,"ack")
Flag	NCreate
Input event	packet(@1,1,3,"ack")
Previous Prov	NULL



Equi key value hash
hash(packet(@1,*,3,*))

Equi key value hash
NULL

Equi key value hash
NULL

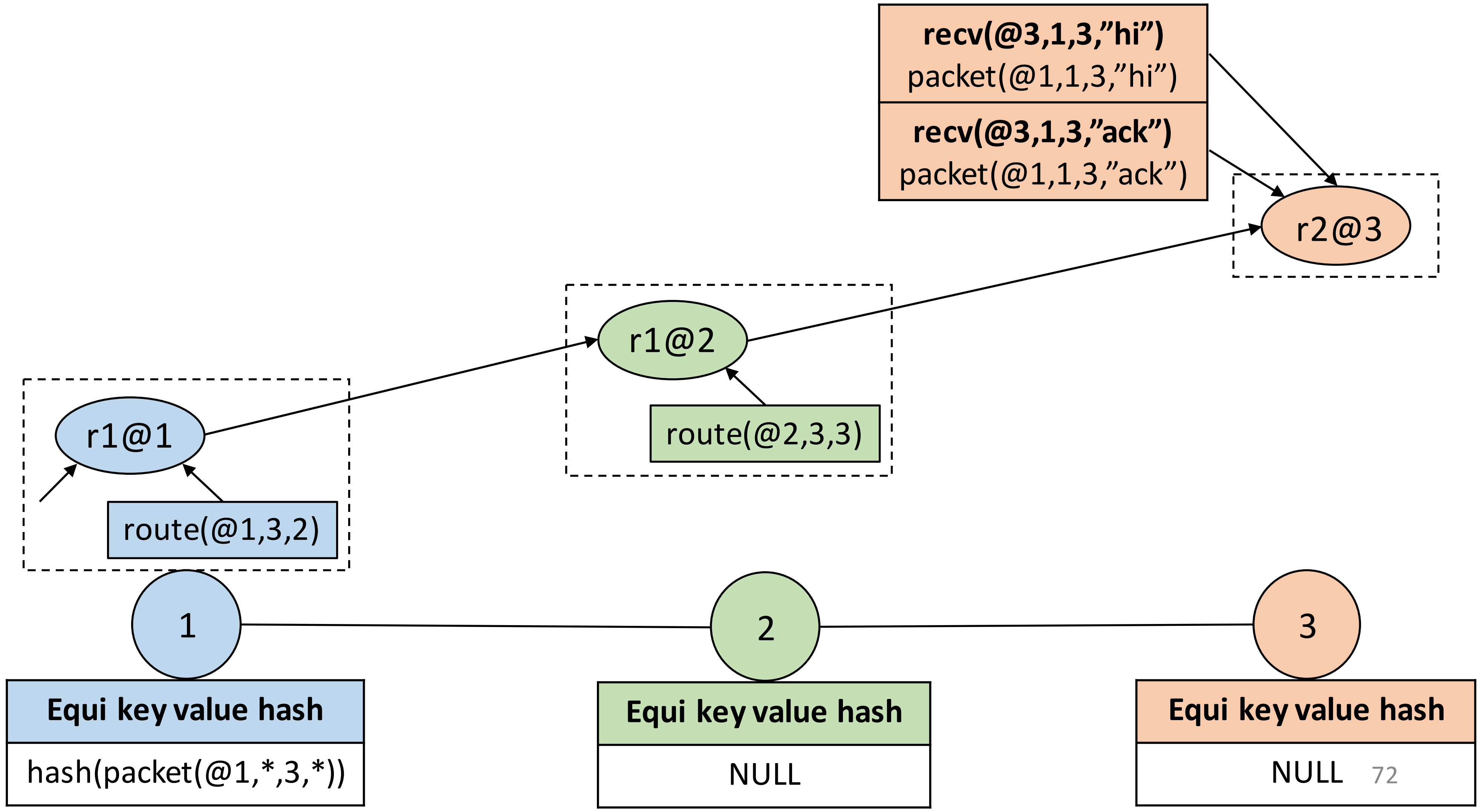
No provenance stored

Stage 3: Associate output tuple to its stored provenance tree

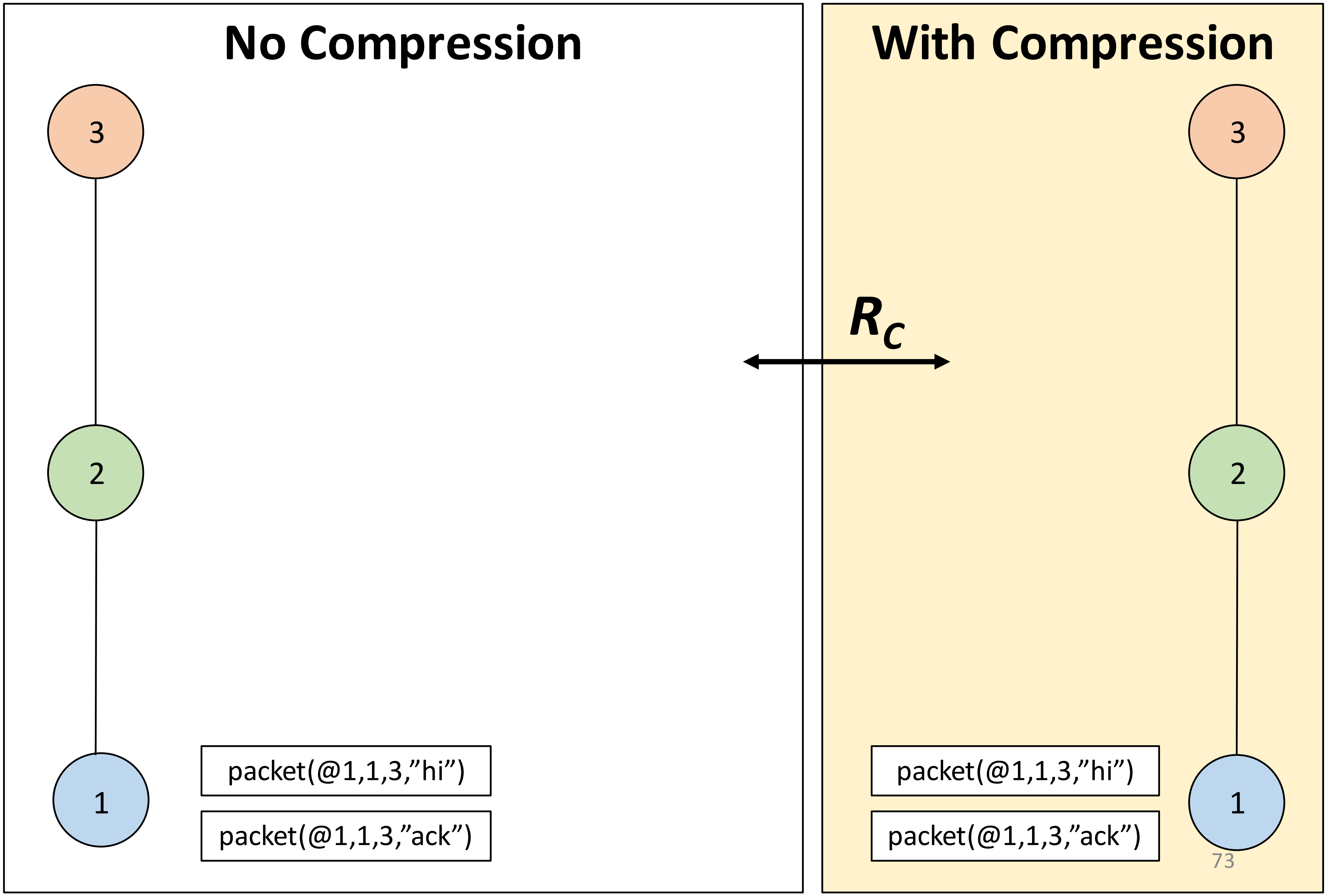
Packet Forwarding

r1 packet(@N,S,D,T) :- packet(@L,S,D,T), route(@L,D,N).

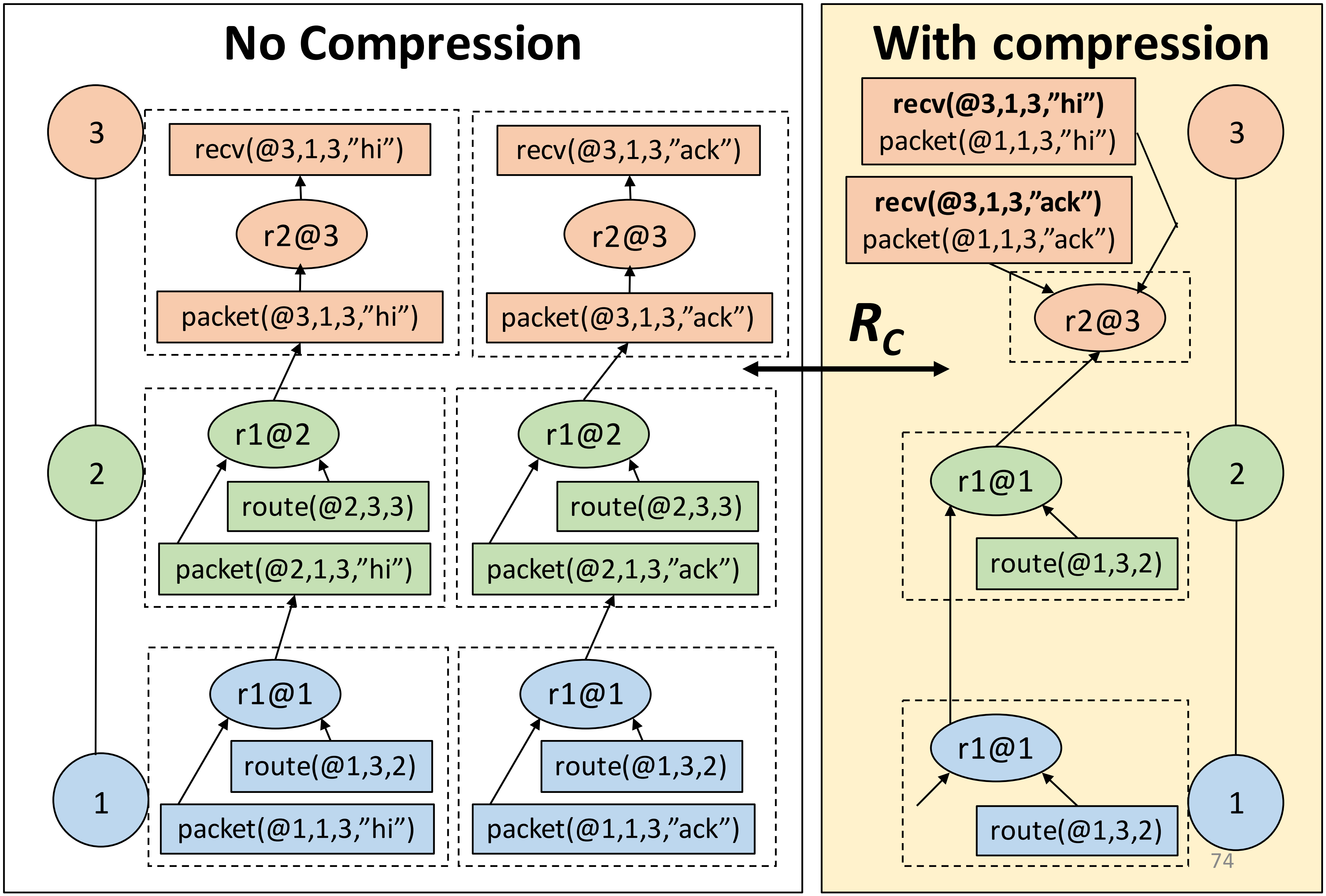
r2 recv(@L,S,D,T) :- packet(@L,S,D,T), D==L.



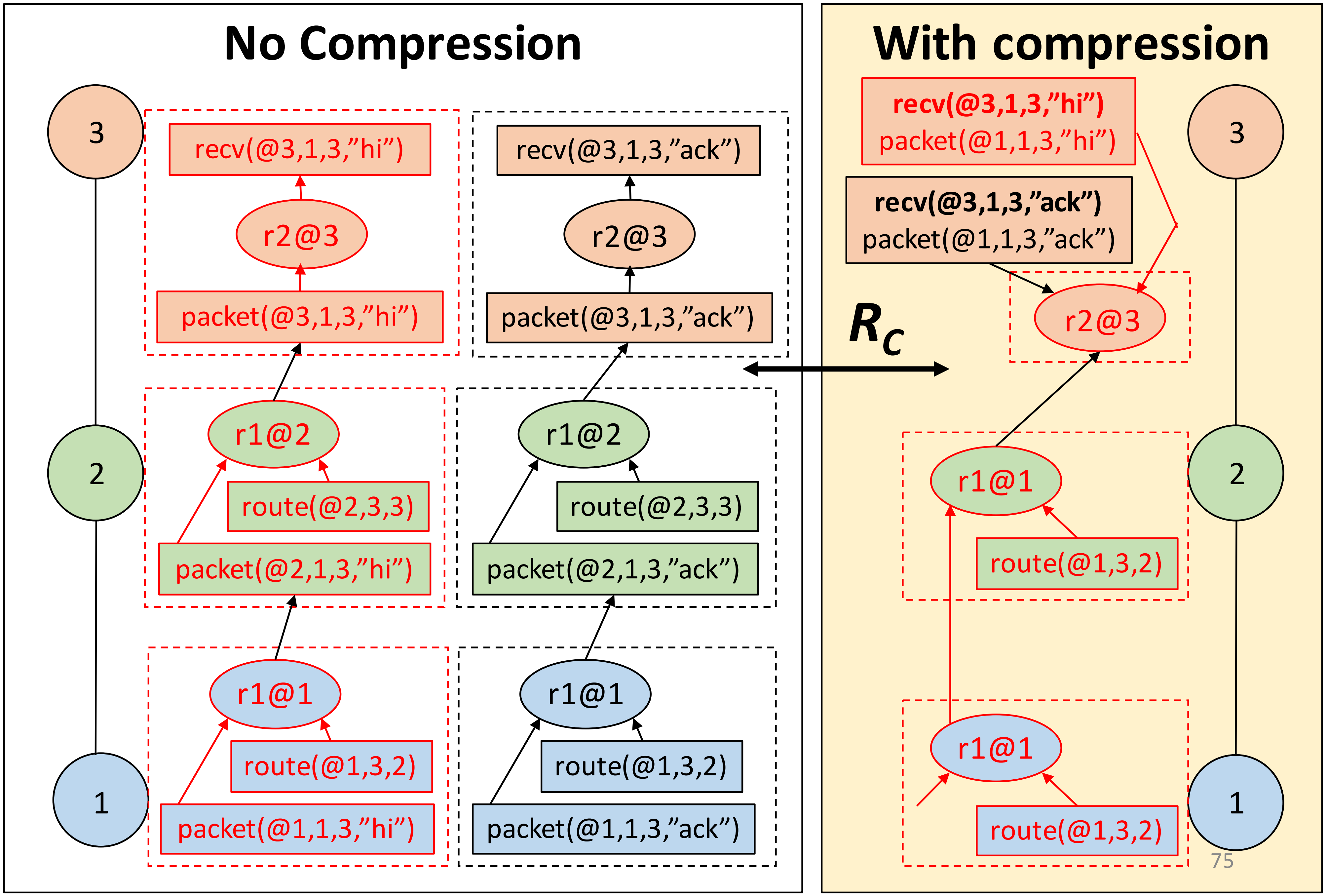
Initial network states



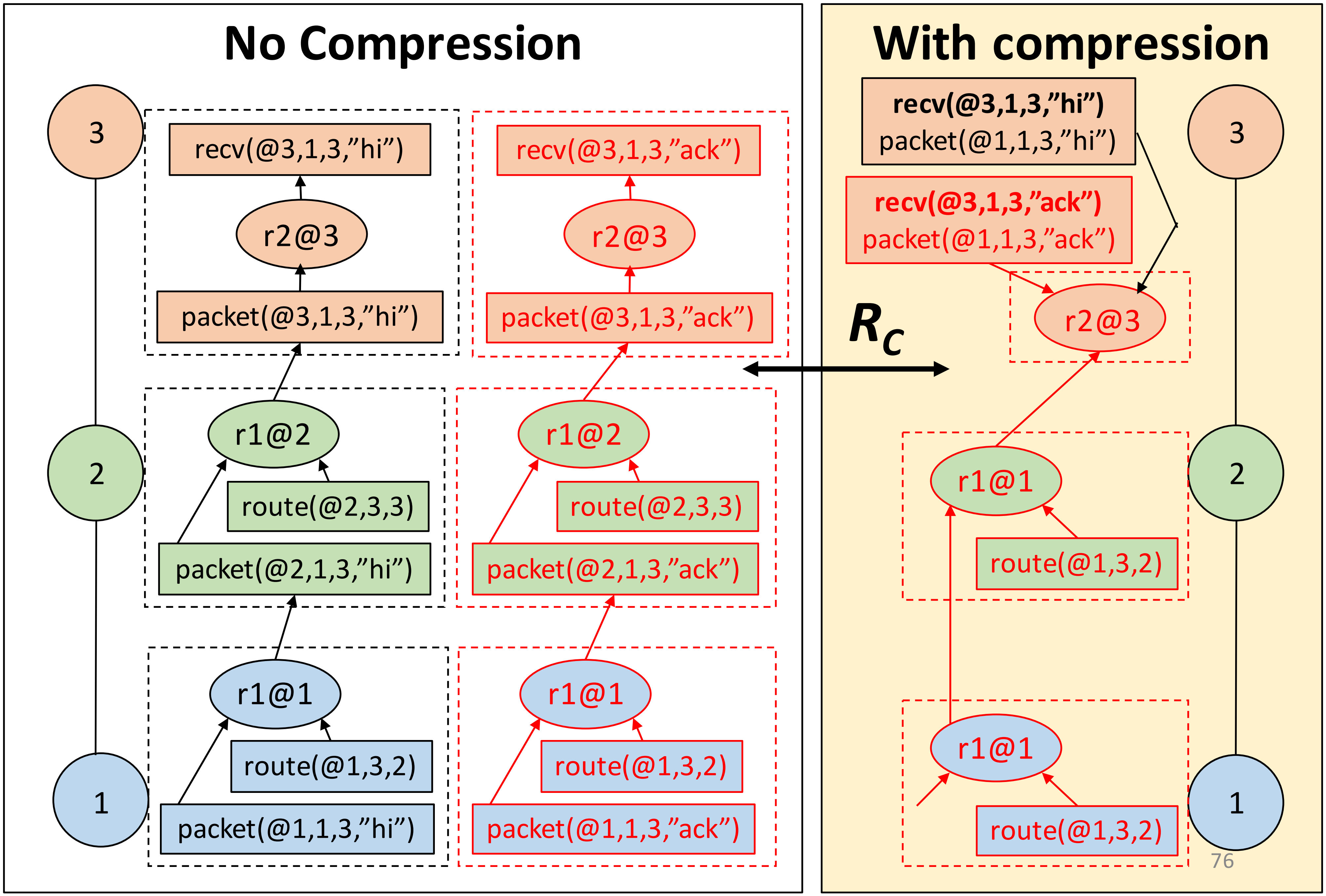
Final network states



Relating network states



Relating network states



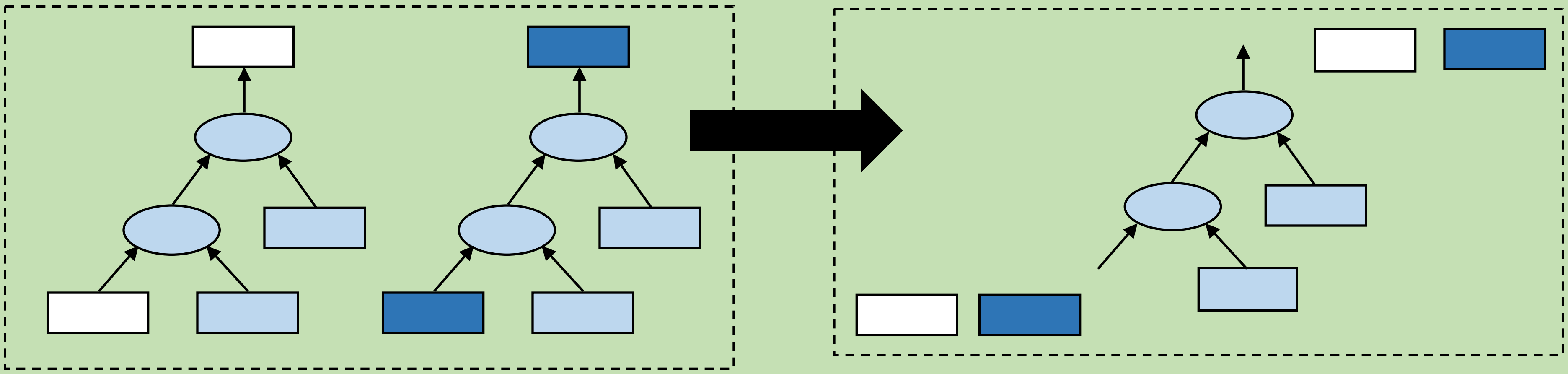
Roadmap

- Background
- Key insights
- Our compression scheme
- ***Conclusion***

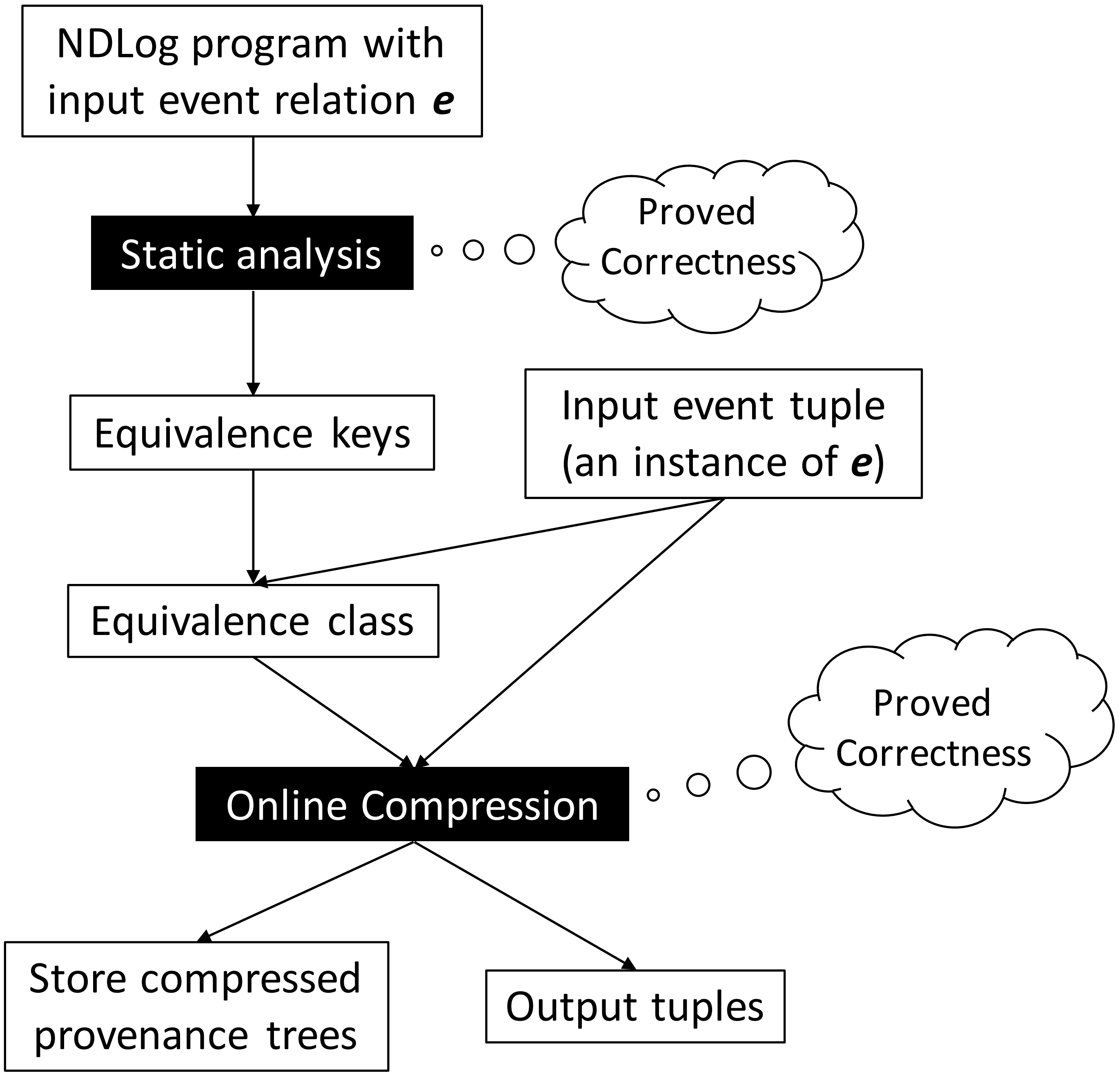
Challenge

Large amount of storage needed to maintain network provenance in a distributed setting

Our Solution



Summary


NDLog program with input event relation ***e***
Static analysis
Proved Correctness
Equivalence keys
Input event tuple (an instance of ***e***)
Equivalence class
Online Compression
Proved Correctness
Store compressed provenance trees
Output tuples