Diagnosing Missing Events in Distributed Systems with Negative Provenance

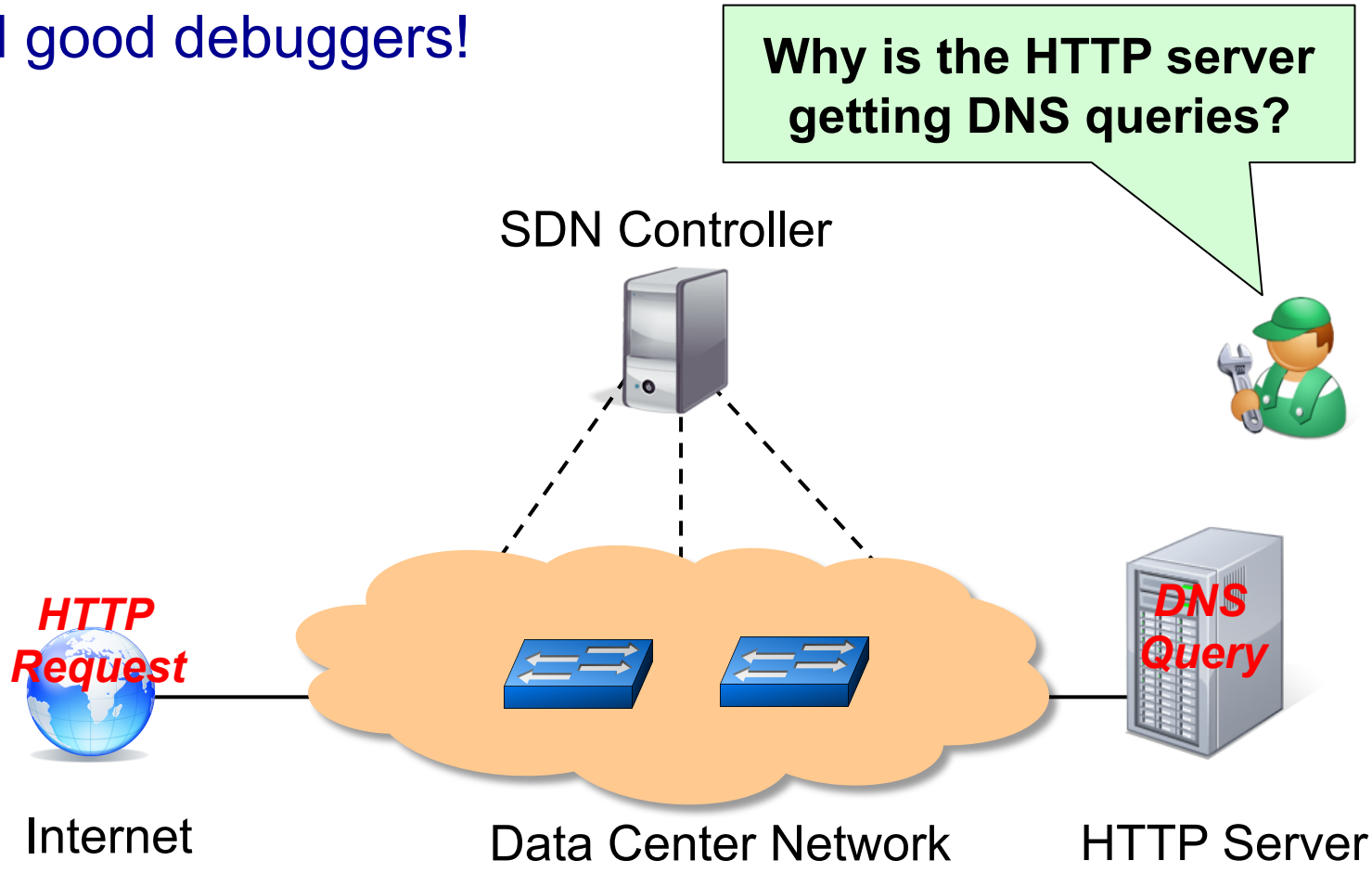
Yang Wu* Mingchen Zhao*

Andreas Haeberlen* Wenchao Zhou+ Boon Thau Loo*

* University of Pennsylvania + Georgetown University

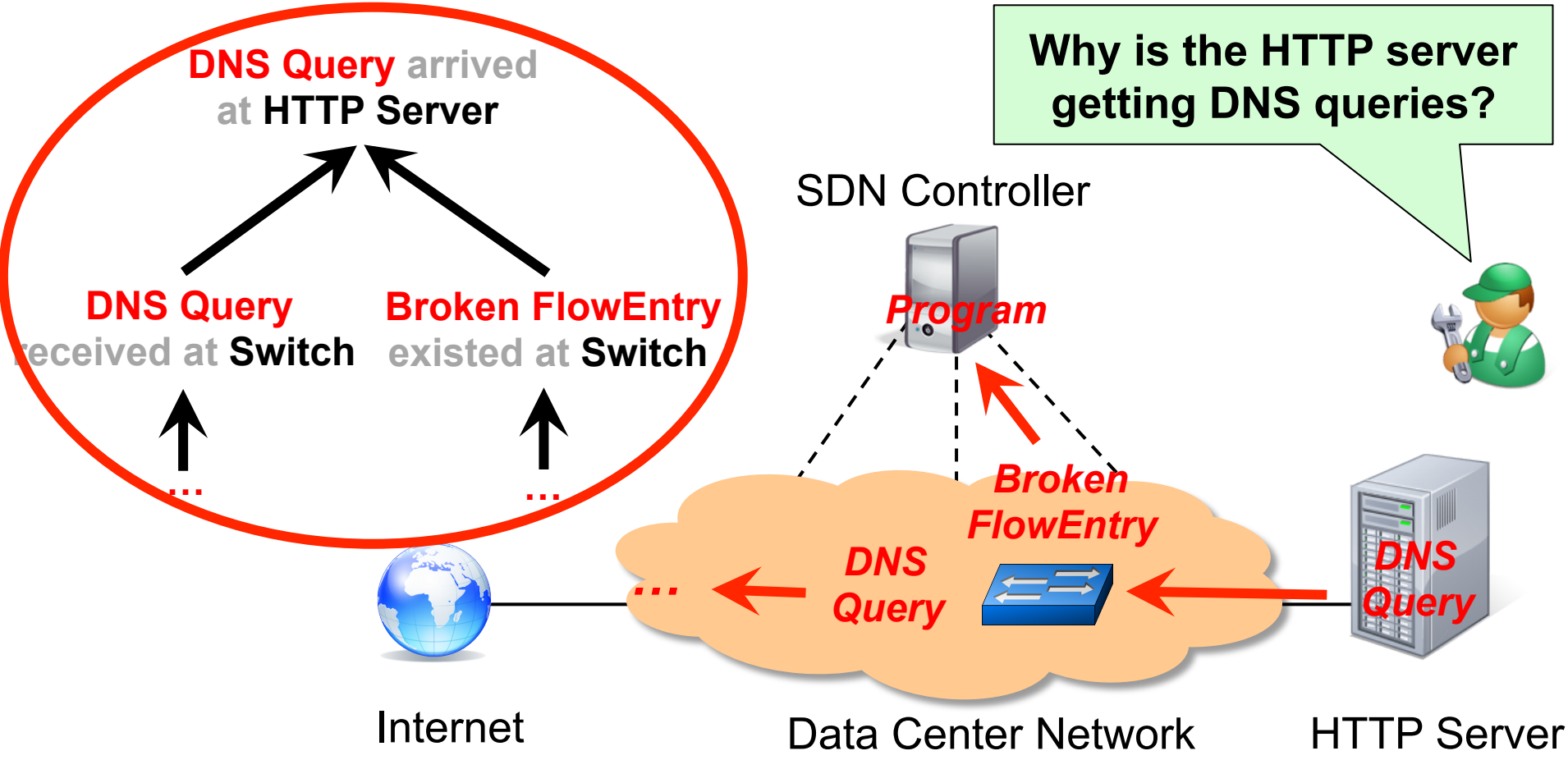
Motivation: Network debugging

- Example: Software Defined Networks
- SDN offers flexibility, but can have bugs
- Need good debuggers!



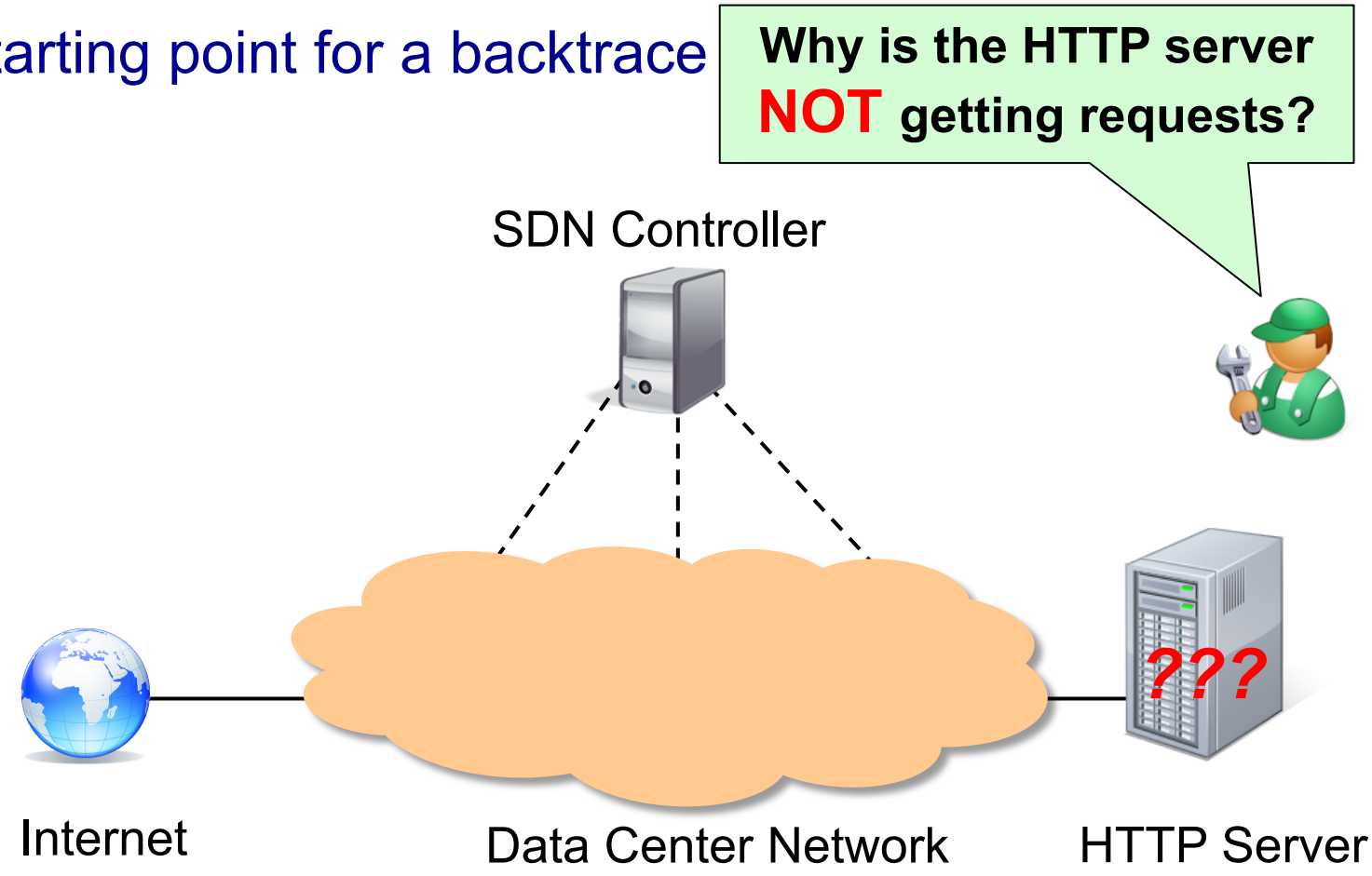
Approach: Provenance

- Existing tools: SNP (SOSP '11), NetSight (NSDI '14)
- They produce "backtraces", or provenance



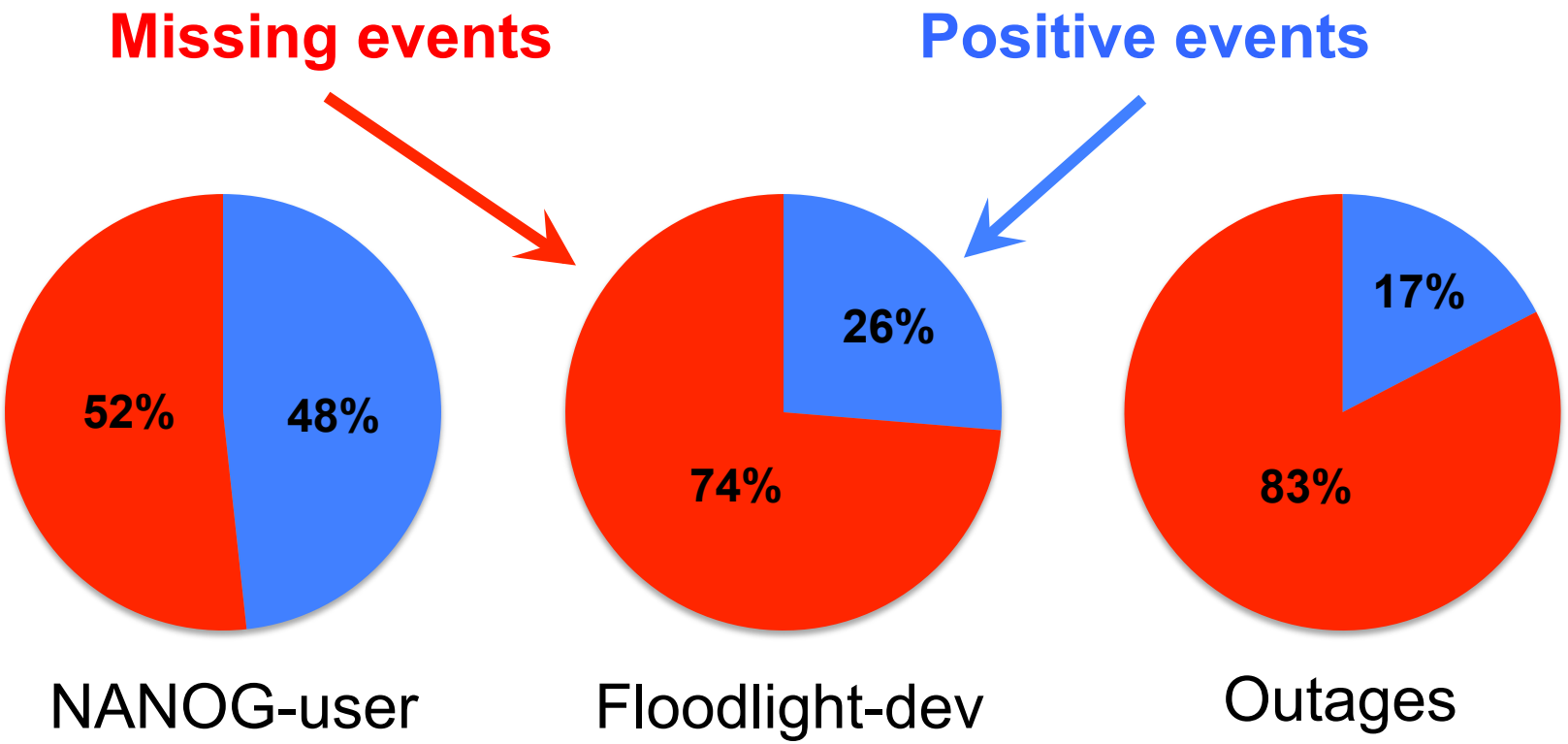
Challenge: Missing events

- What if an expected event does not happen?
- Cannot be handled by existing tools
- No starting point for a backtrace



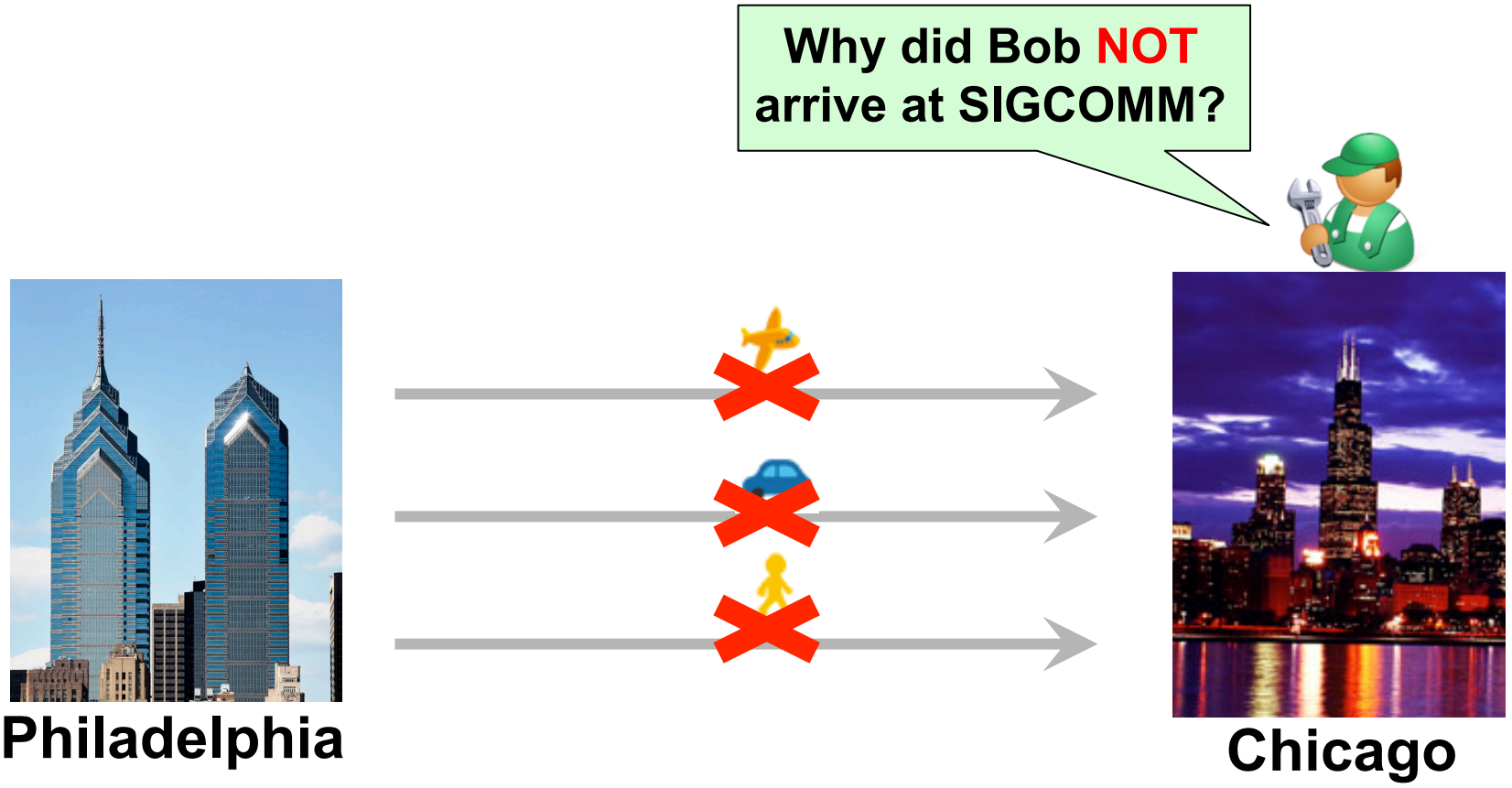
Survey: How common are missing events?

- Missing events are consistently in the majority
- Email threads for missing events are longer

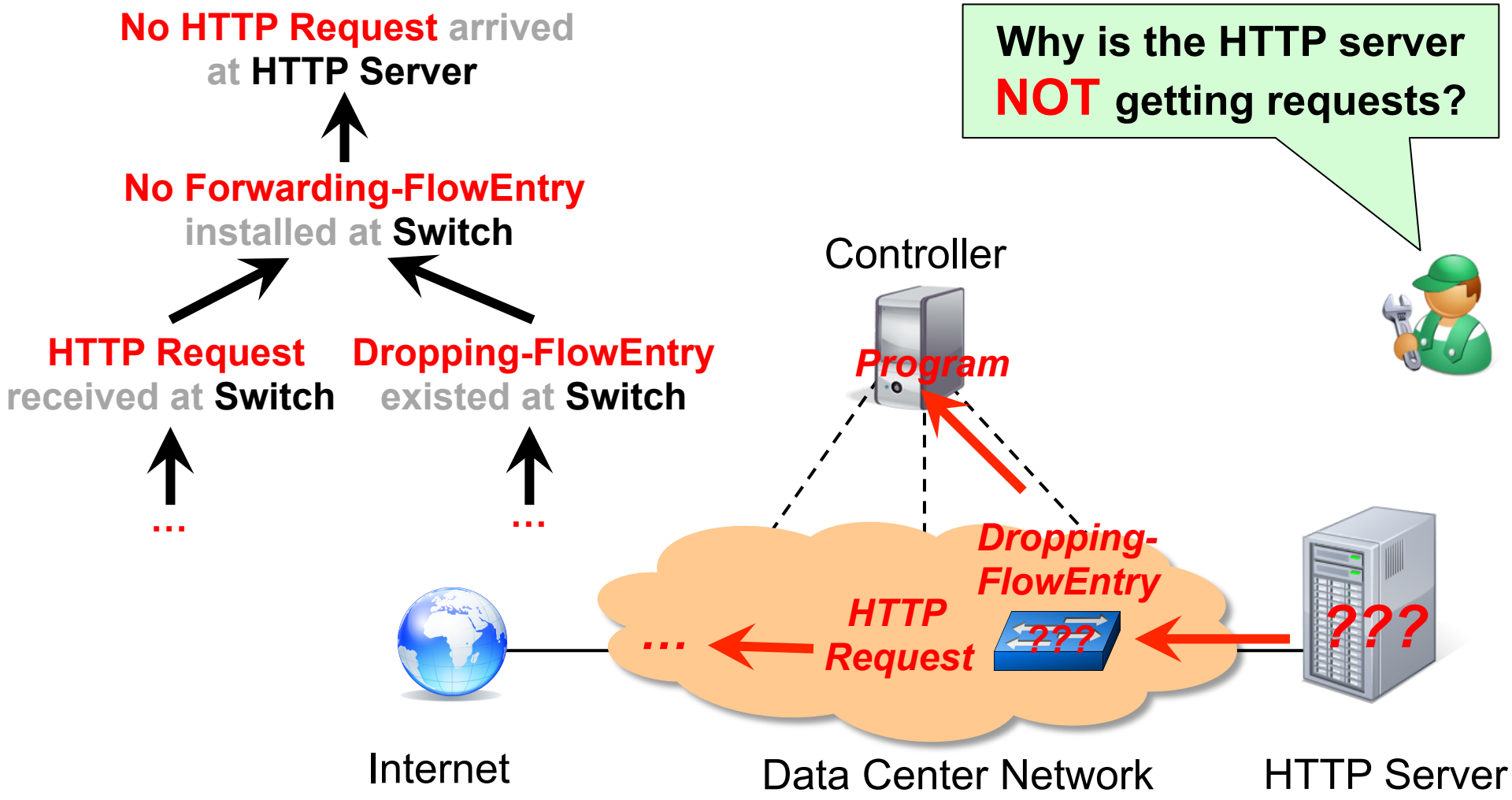


Approach: Counter-factual reasoning

Find all the ways a missing event could have occurred, and show why each of them did not happen.

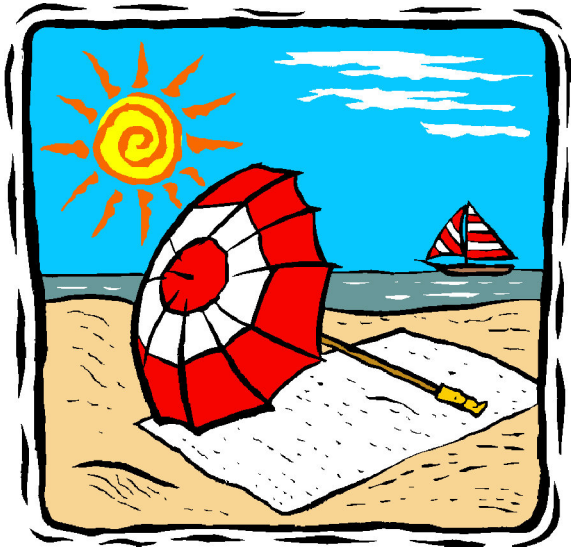


Result: Debugger for missing events



Challenge: Too many possible explanations!

Why did Bob NOT arrive at SIGCOMM?

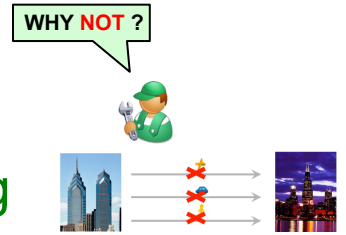


When an event happens, there is one reason.

When an event does not happen, there can be many reasons.

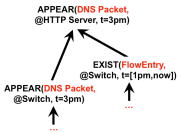
Overview

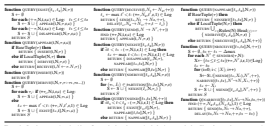
- Goal: Diagnose missing events
- Approach: Counter-factual reasoning
- Challenge: Too many explanations

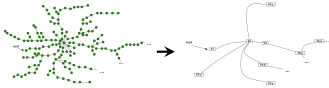


Approach

- Background: Provenance
- Generating Negative Provenance
- Improving readability

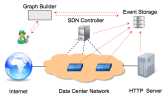


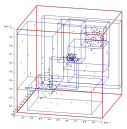




System

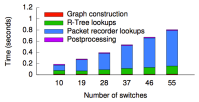
- Y!
- R-tree indexing

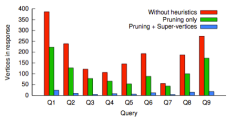




Evaluation

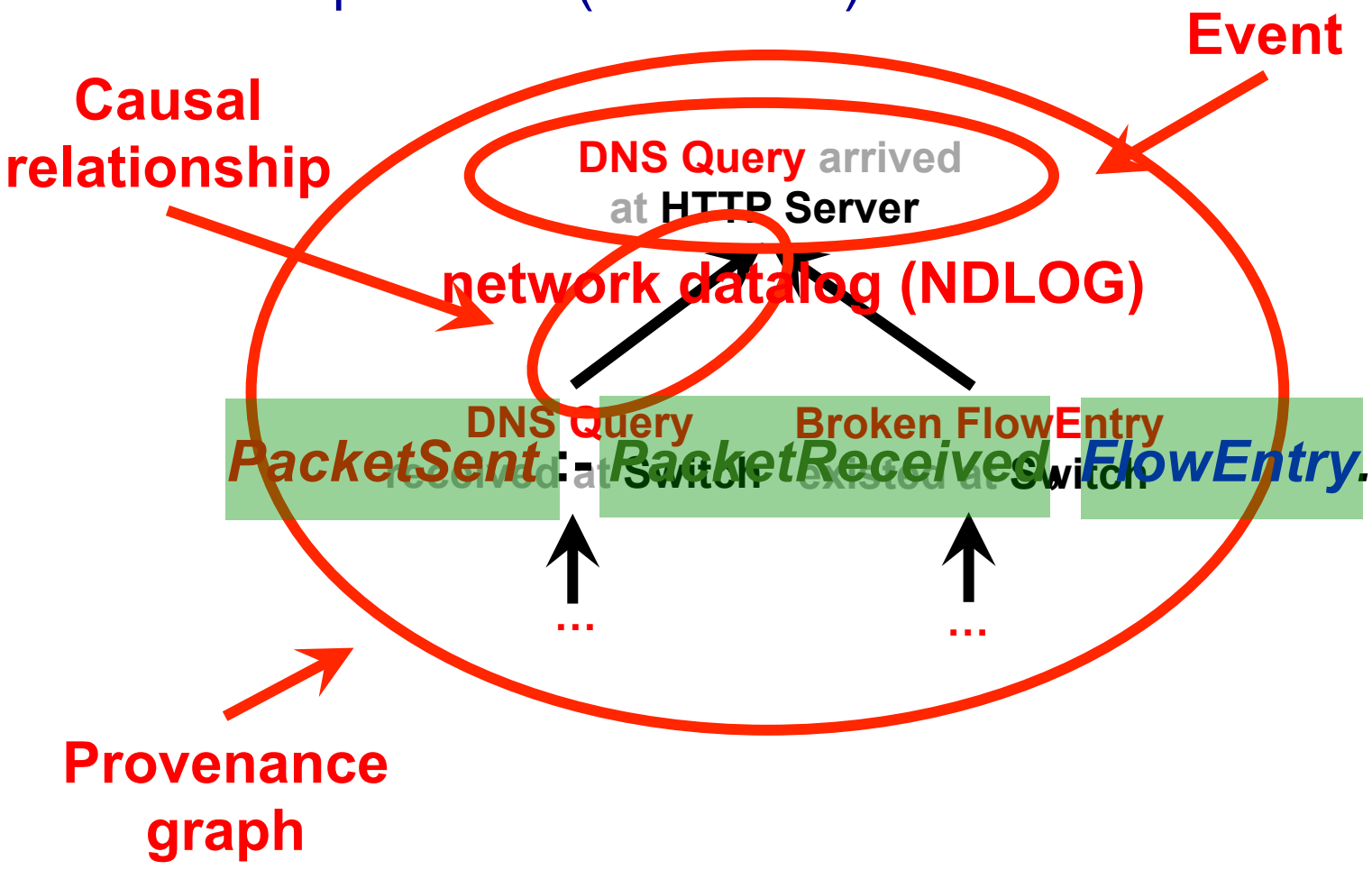
- Experiments
- Query speed
- Size reduction
- Usability





Background: Provenance

- Captures causality between events
- Example: SNP (SOSP '11)

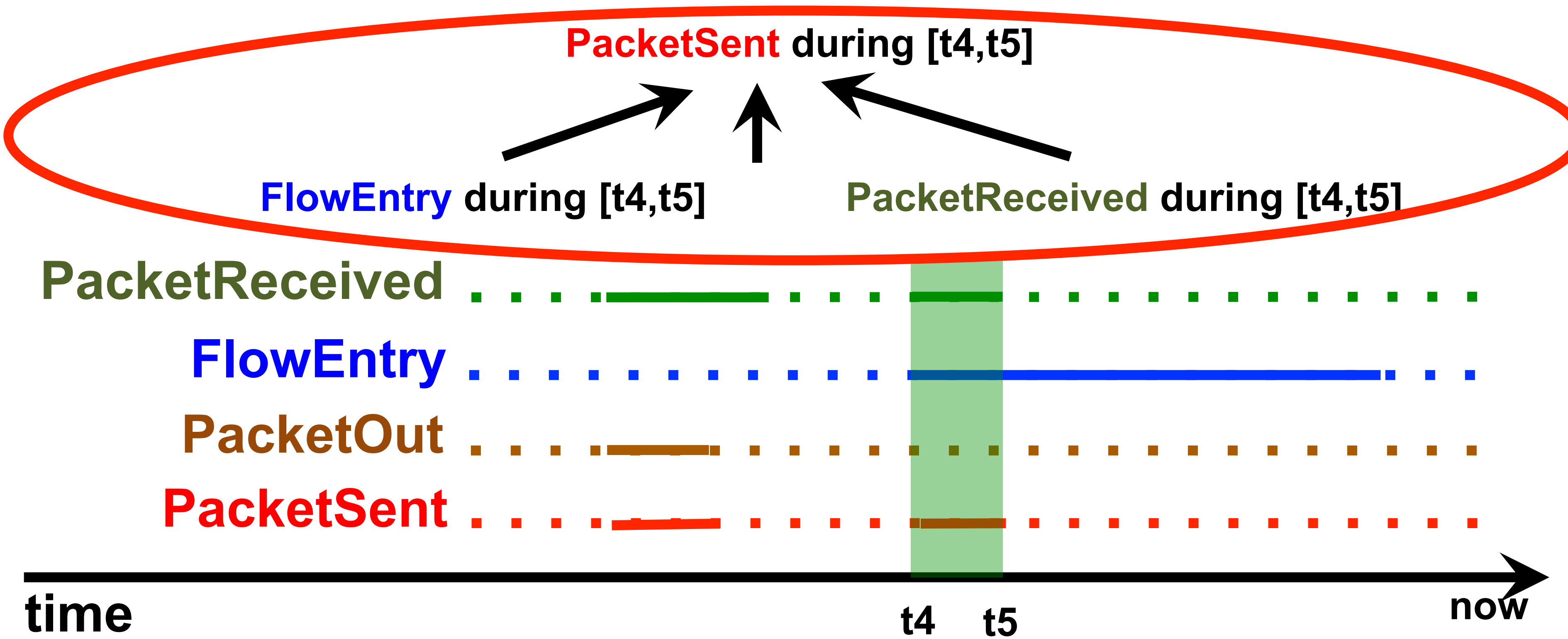


Background: How to generate provenance?

Step [illegible]

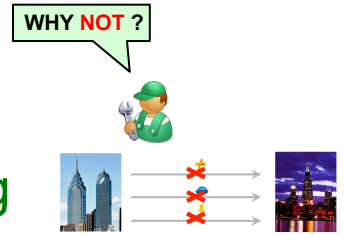
PacketSent **:-** ***PacketReceived*****,** ***FlowEntry*****.**

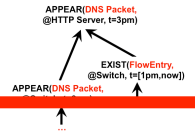
PacketSent **:-** ***PacketOut*****.**



Overview

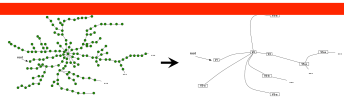
- Goal: Diagnose missing events
- Approach: Counter-factual reasoning
- Challenge: Too many explanations
- Background: Provenance





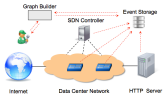
Approach

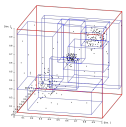
- Generating Negative Provenance
- Improving readability



System

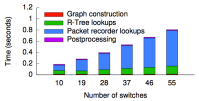
- Y!
- R-tree indexing

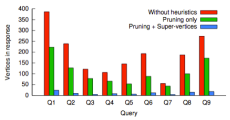




Evaluation

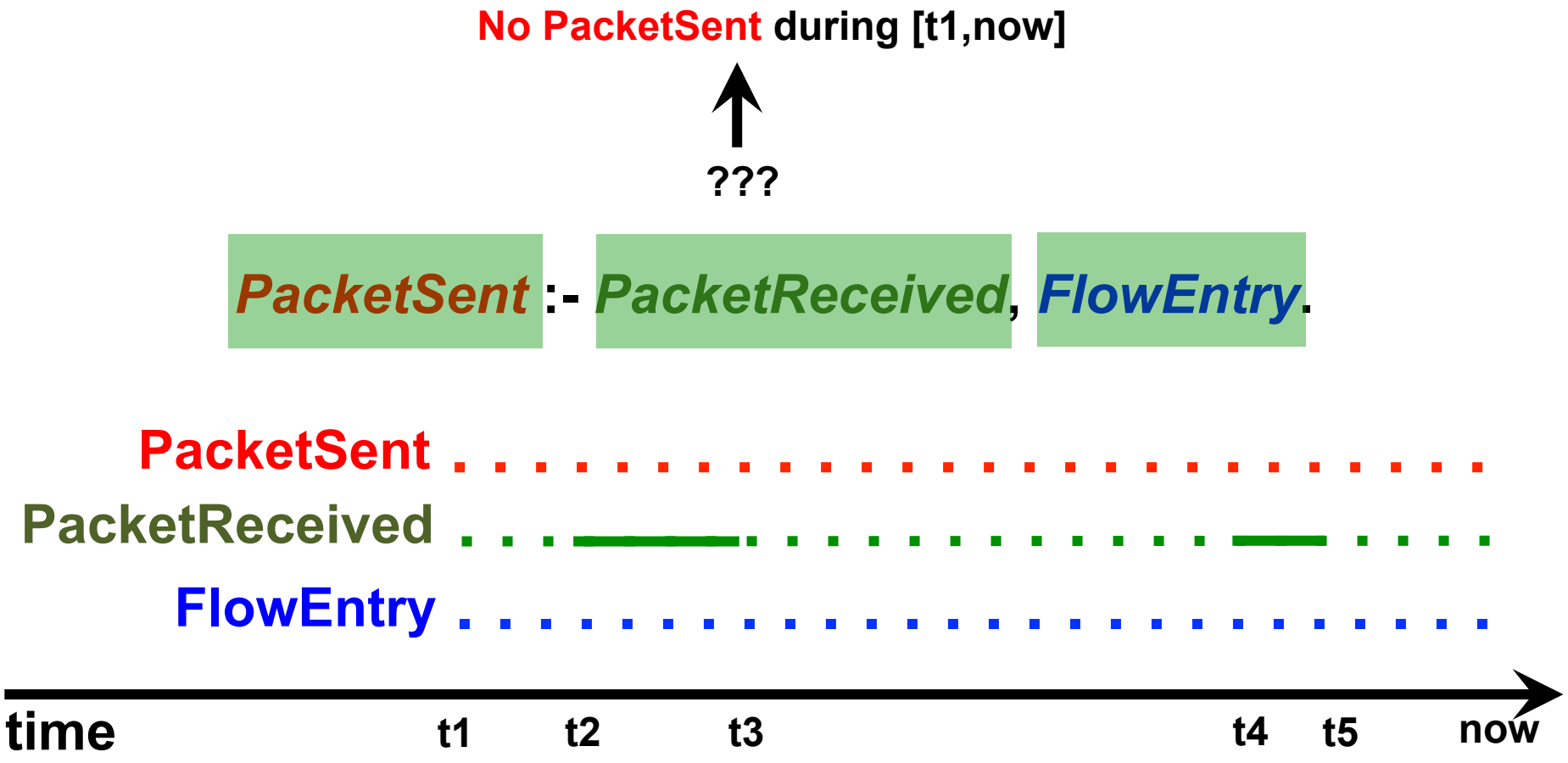
- Experiments
- Query speed
- Size reduction
- Usability





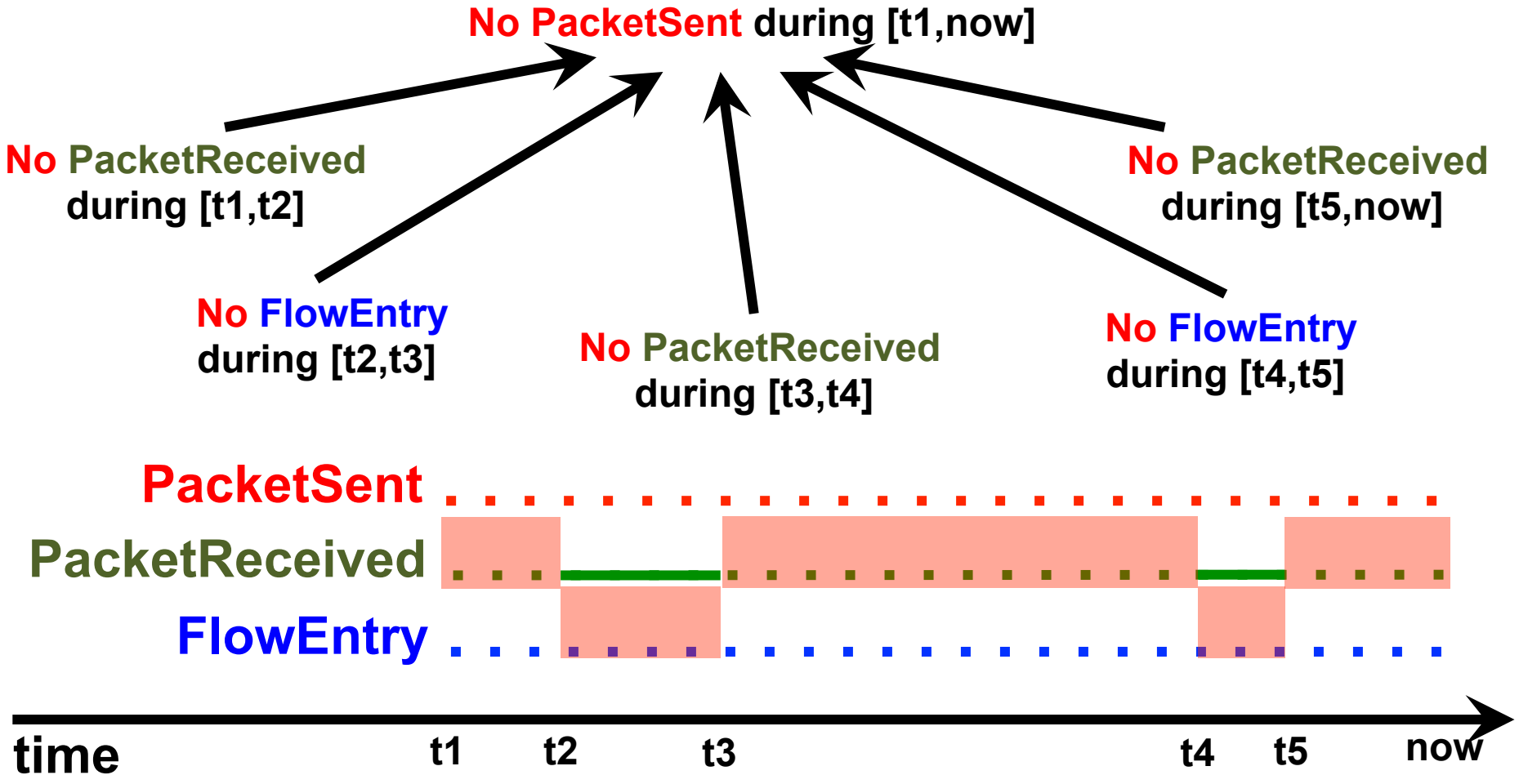
Generating negative provenance graphs

- Goal: Explain why something does not exist
- Use missing preconditions to explain missing events



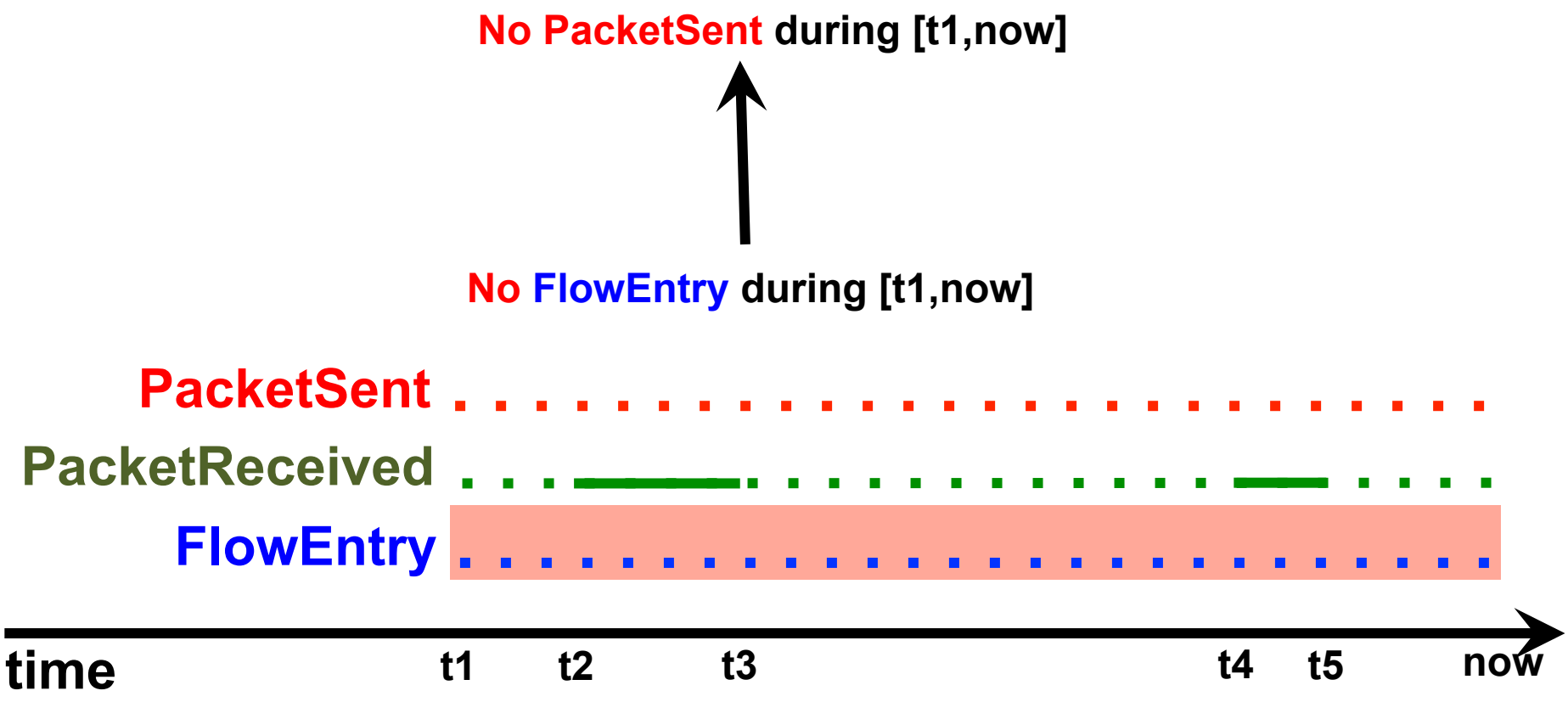
Generating negative provenance graphs

- Explanation can be unnecessarily complex



Generating negative provenance graphs

- We want simple explanations
- This is hard (Set-Cover)
- But greedy heuristics tend to work well



Generating negative provenance graphs

function QUERY(EXIST($[t_1, t_2]$,N,τ))
  S $\leftarrow \emptyset$
  for each ($+\tau$,N,t,r,c) $\in$ Log: $t_1 \leq t \leq t_2$
    S $\leftarrow$ S $\cup$ { APPEAR(t,N,τ,r,c) }
  for each ($-\tau$,N,t,r,c) $\in$ Log: $t_1 \leq t \leq t_2$
    S $\leftarrow$ S $\cup$ { DISAPPEAR(t,N,τ,r,c) }
  RETURN S

function QUERY(APPEAR(t,N,τ,r,c))
  if BaseTuple(τ) **then**
    RETURN { INSERT(t,N,τ) }
  else if LocalTuple(N,τ) **then**
    RETURN { DERIVE(t,N,τ,r) }
  else RETURN{RECEIVE(t,$N \leftarrow r.N$,τ)}

function QUERY(INSERT(t,N,τ))
  RETURN $\emptyset$

function QUERY(DERIVE(t,N,τ,τ:-τ_1,τ_2...))
  S $\leftarrow \emptyset$
  for each τ_i: **if** ($+\tau_i$,N,t,r,c) $\in$ Log:
    S $\leftarrow$ S $\cup$ { APPEAR(t,N,τ_i,c) }
  else
    $t_x \leftarrow$ max $t' < t$: ($+\tau$,N,t',r,1) $\in$ Log
    S $\leftarrow$ S $\cup$ { EXIST($[t_x$,t],N,τ_i,c) }
  RETURN S

function QUERY(RECEIVE(t,$N_1 \leftarrow N_2$,$+\tau$))
  $t_s \leftarrow$ max $t' < t$: ($+\tau$,N_2,t',r,1) $\in$ Log
  RETURN { SEND(t_s,$N_1 \rightarrow N_2$,$+\tau$),
    DELAY(t_s,$N_2 \rightarrow N_1$,$+\tau$,$t - t_s$) }

function QUERY(SEND(t,$N \rightarrow N'$,$+\tau$))
  FIND ($+\tau$,N,t,r,c) $\in$ Log
  RETURN { APPEAR(t,N,τ,r) }

function QUERY(NEXIST($[t_1$,t_2],N,τ))
  if $\exists t < t_1$: ($-\tau$,N,t,r,1) $\in$ Log **then**
    $t_x \leftarrow$ max $t < t_1$: ($-\tau$,N,t,r,1) $\in$ Log
    RETURN { DISAPPEAR(t_x,N,τ),
      NAPPEAR((t_x,t_2],N,τ) }
  else RETURN { NAPPEAR([0,t_2],N,τ) }

function QUERY(NDERIVE($[t_1$,t_2],N,τ,r))
  S $\leftarrow \emptyset$
  for $(\tau_i, I_i) \in$ PARTITION($[t_1$,t_2],N,τ,r)
    S $\leftarrow$ S $\cup$ { NEXIST(I_i,N,τ_i) }
  RETURN S

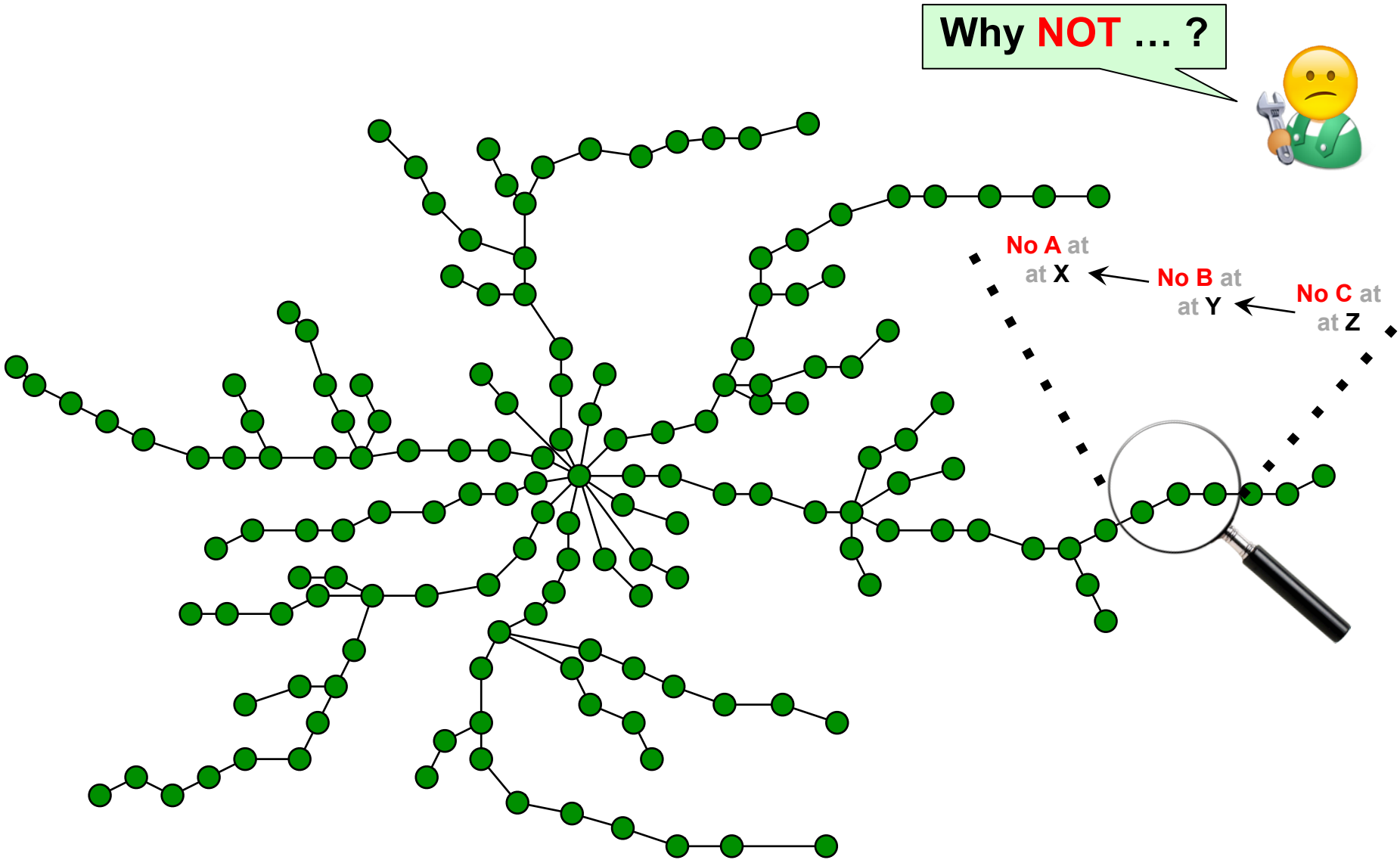
function QUERY(NSEND($[t_1$,t_2],N,$+\tau$))
  if $\exists t_1 < t < t_2$: ($-\tau$,N,t,r,1) $\in$ Log **then**
    RETURN { EXIST($[t_1$,t],N,τ),
      NAPPEAR((t,t_2],N,τ) }
  else RETURN { NAPPEAR($[t_1$,t_2],N,τ) }

function QUERY(NAPPEAR($[t_1$,t_2],N,τ))
  if BaseTuple(τ) **then**
    RETURN { NINSERT($[t_1$,t_2],N,τ) }
  else if LocalTuple(N,τ) **then**
    RETURN $\bigcup_{r \in \text{Rules(N):Head}(r)=\tau}$
      { NDERIVE($[t_1$,t_2],N,τ,r) }
  else RETURN {NRECEIVE($[t_1$,t_2],N,$+\tau$)}

function QUERY(NRECEIVE($[t_1$,t_2],N,$+\tau$))
  S $\leftarrow \emptyset$, $t_0 \leftarrow t_1 - \Delta$max
  for each $N' \in$ SENDERS(τ,N):
    X$\leftarrow \{t_0 \leq t \leq t_2 | (+\tau, N', \text{t,r,1}) \in \text{Log}\}$
    $t_x \leftarrow t_0$
    for (i=0; i< $|X|$; i++)
      S$\leftarrow$S$\cup${NSEND((t_x,X_i),N',$+\tau$),
      NARRIVE((t_1,t_2),$N' \rightarrow N$,X_i,$+\tau$)}
      $t_x \leftarrow X_i$
    S $\leftarrow$ S $\cup$ { NSEND($[t_x$,t_2],N',$+\tau$) }
  RETURN S

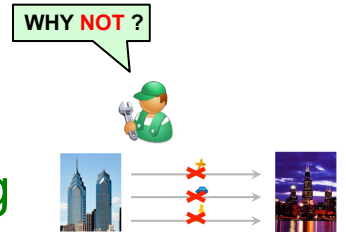
function Q(NARRIVE($[t_1$,t_2],$N_1 \rightarrow N_2$,t_0,$+\tau$))
  FIND ($+\tau$,N_2,t_3,(N_1,t_0),1) $\in$ Log
  RETURN { SEND(t_0,$N_1 \rightarrow N_2$,$+\tau$),
    DELAY(t_0,$N_1 \rightarrow N_2$,$+\tau$,$t_3 - t_0$) }

Challenge: Explanation is complicated!



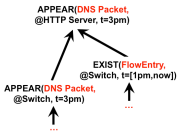
Overview

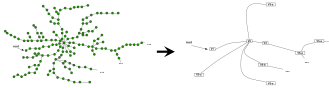
- Goal: Diagnose missing events
- Approach: Counter-factual reasoning
- Challenge: Too many explanations



Approach

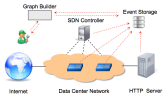
- Background: Provenance
- Generating Negative Provenance
- Improving readability

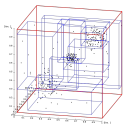




System

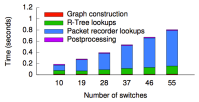
- Y!
- R-tree indexing

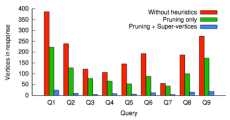




Evaluation

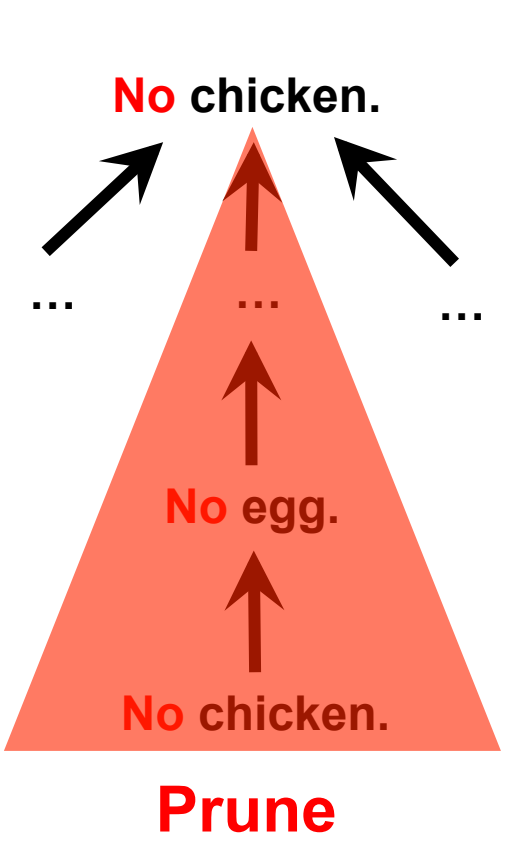
- Experiments
- Query speed
- Explanation size reduction
- Explanation usability

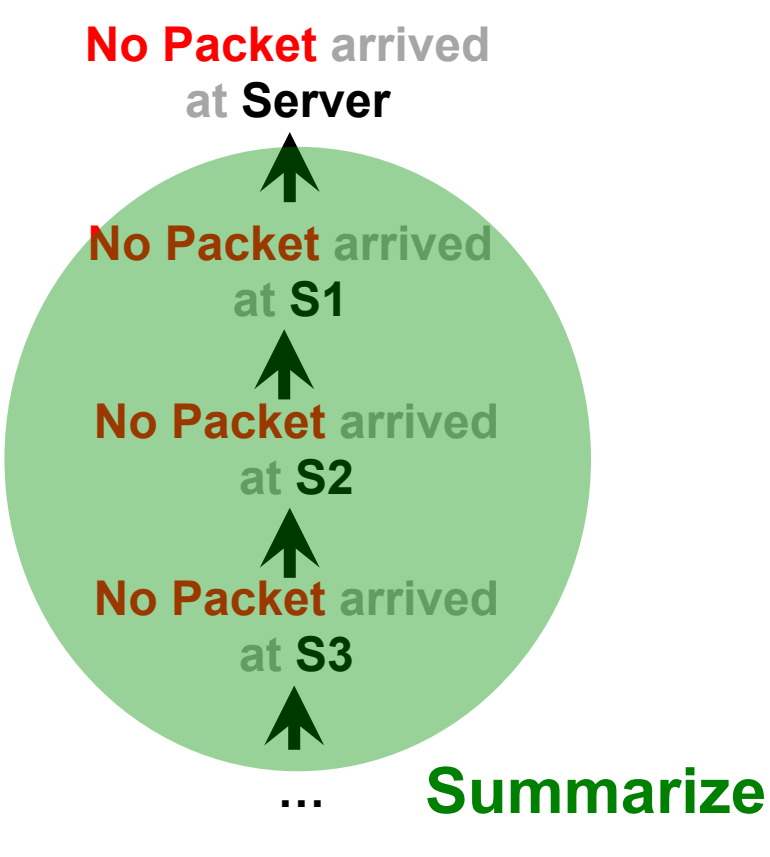




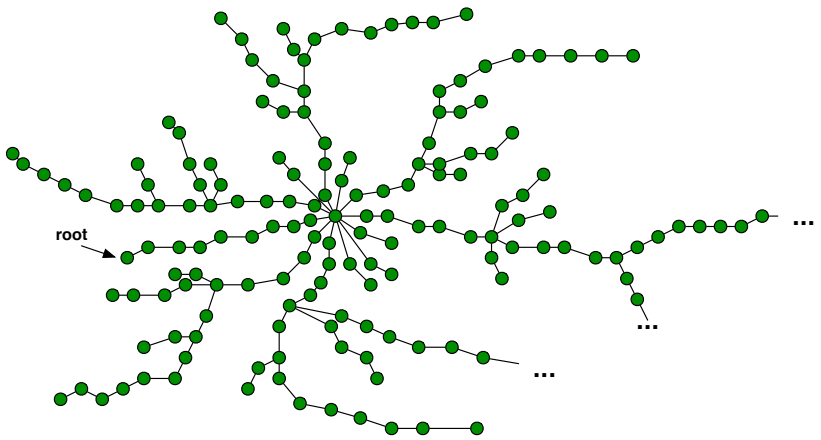
Readability: How to simplify the provenance?

- Heuristic #1: Prune logical inconsistencies
- Heuristic #2: Summarize transient event chains





Readability: Other heuristics



Prune logical inconsistencies.

Prune failed assertions.

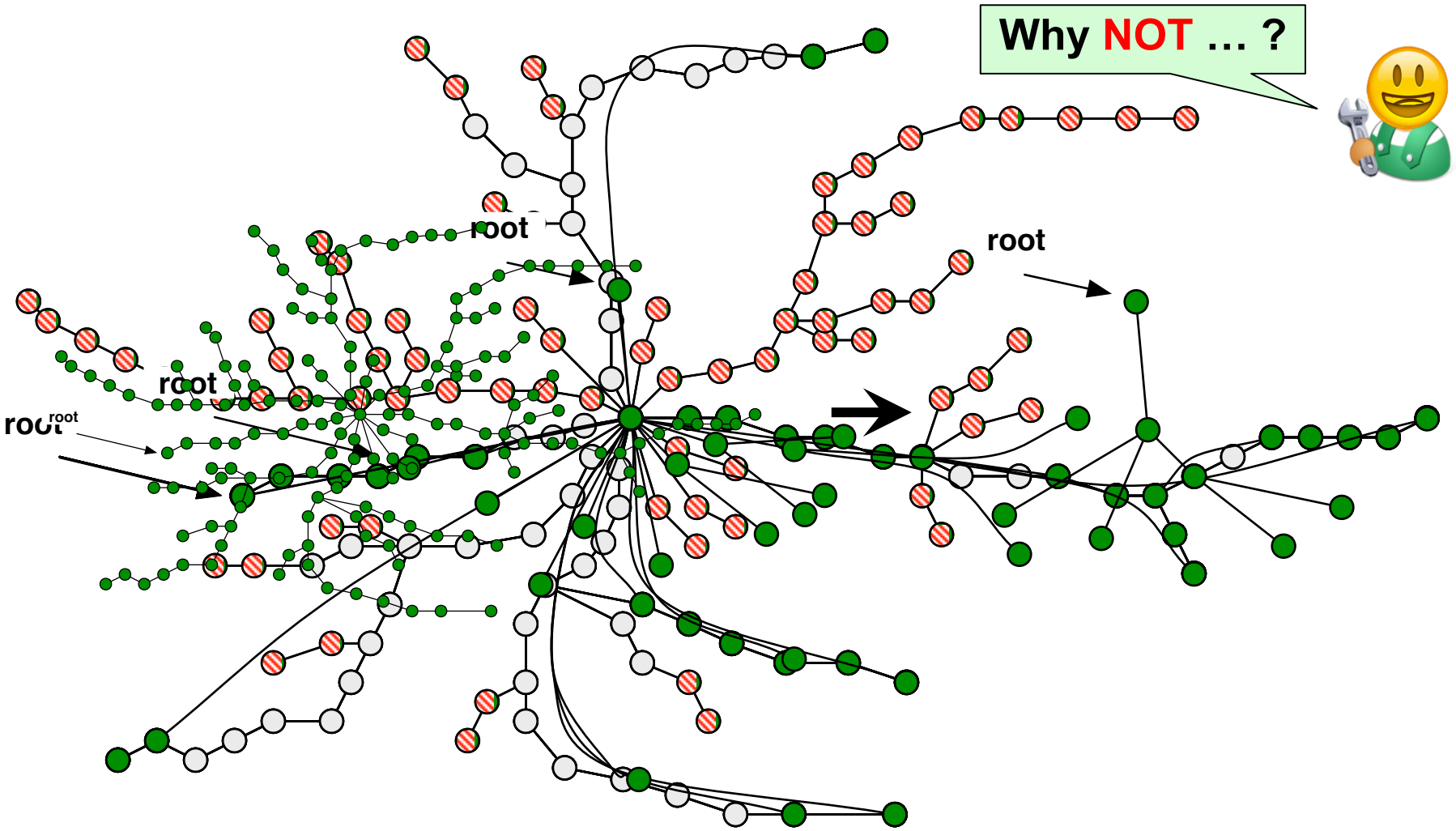
Branch coalescing.

Application-specific invariants.

Summarize transient event chains.

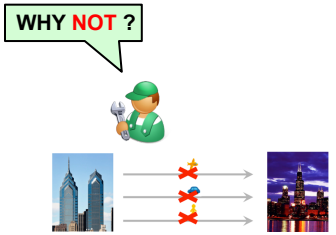
Summarize super-vertex.

Readability: Concise explanations



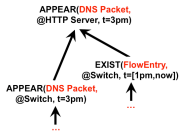
Overview

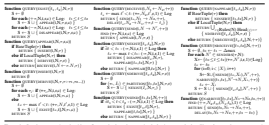
- Goal: Diagnose missing events
- Approach: Counter-factual reasoning
- Challenge: Too many explanations

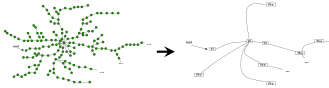


Approach

- Background: Provenance
- Generating Negative Provenance
- Improving readability

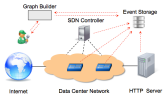


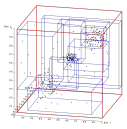




System

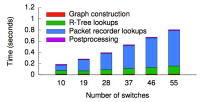
- Y!
- R-tree indexing

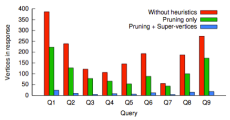




Evaluation

- Experiments
- Query speed
- Explanation size reduction
- Explanation usability



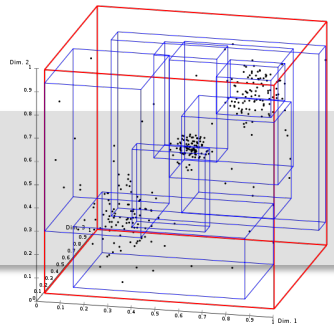


System: Y!

General: Works for any NDLOG program (not just SDN)

Supports general programs: Pyretic frontend frenetic >>

Uses R-tree to speed up queries

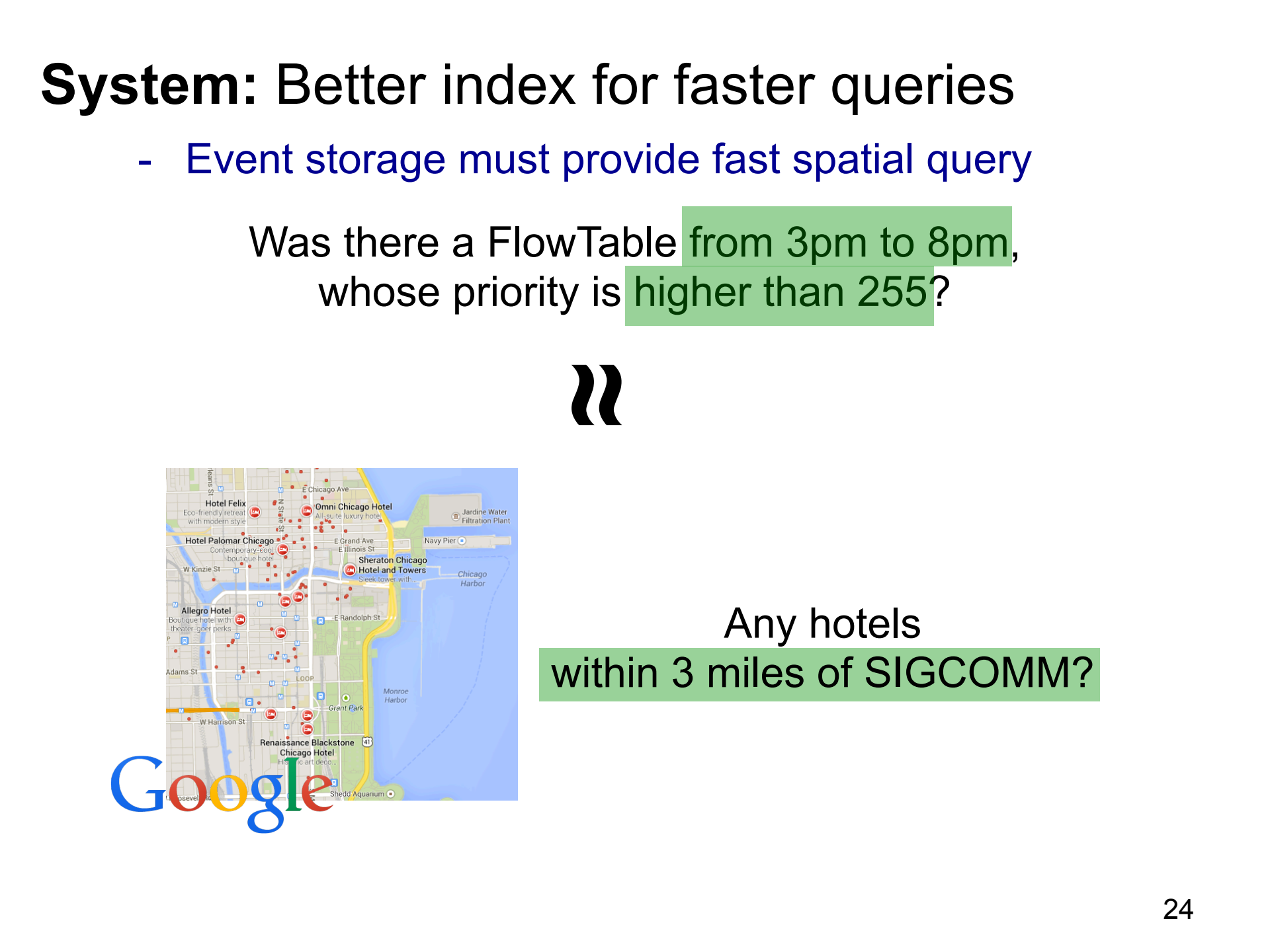


More details are in the paper

System: Better index for faster queries

- Event storage must provide fast spatial query

Was there a FlowTable from 3pm to 8pm,
whose priority is higher than 255?

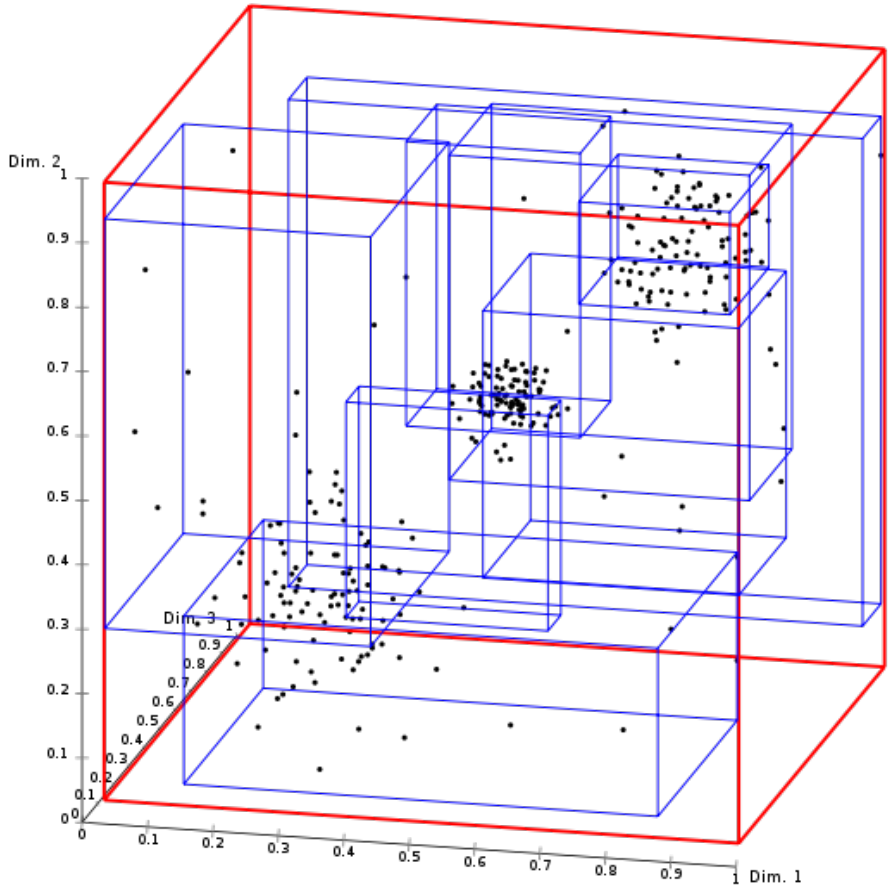


Google

Any hotels
within 3 miles of SIGCOMM?

System: R-tree for faster queries

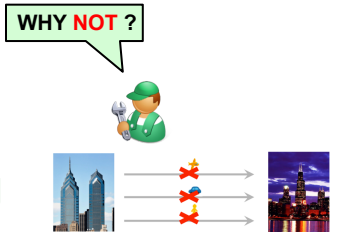
- R-tree: Designed to handle high-dimensional queries
- Basic idea: Multi-dimensional boxes as indexes



Used material from Wikipedia.

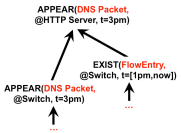
Overview

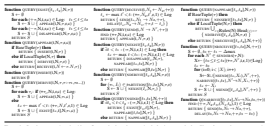
- Goal: Diagnose missing events
- Approach: Counter-factual reasoning
- Challenge: Too many explanations

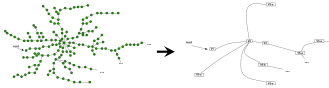


Approach

- Background: Provenance
- Generating Negative Provenance
- Improving readability

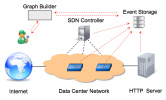


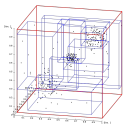




System

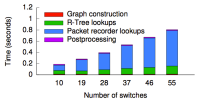
- Y!
- R-tree indexing

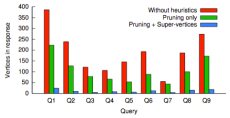




Evaluation

- Experiments
- Query speed
- Size reduction
- Usability





Evaluation: Setup

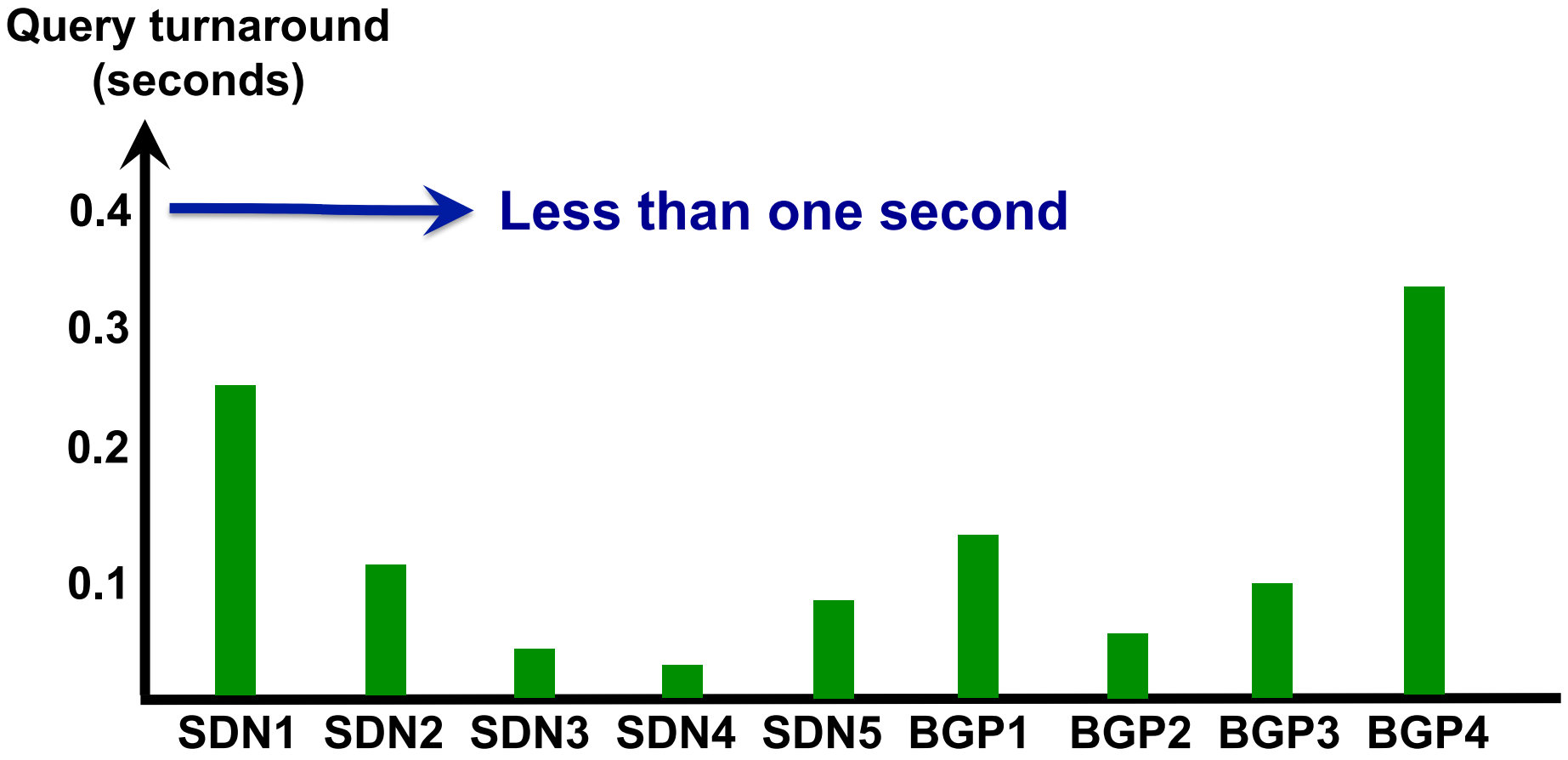
- Two case studies: SDN and BGP
- Simulation stack: RapidNet + Mininet + Trema
- Buggy scenarios reproduced from literature and survey
  - SDN1: Broken flow entry
  - SDN2: MAC spoofing
  - SDN3: Incorrect ACL
  - SDN4: Ping traceback
  - SDN5: Internal access
  - BGP1: Off-path change
  - BGP2: Black hole
  - BGP3: Link failure
  - BGP4: Bogon List

Evaluation: Questions

Are negative provenance graphs concise?
Are negative provenance graphs useful?
What is the query turnaround time?
What is the runtime storage overhead?
Will Y! slow down the distributed system?
How runtime storage overhead scales?
How query turnaround time scales?
How readability heuristics scales?

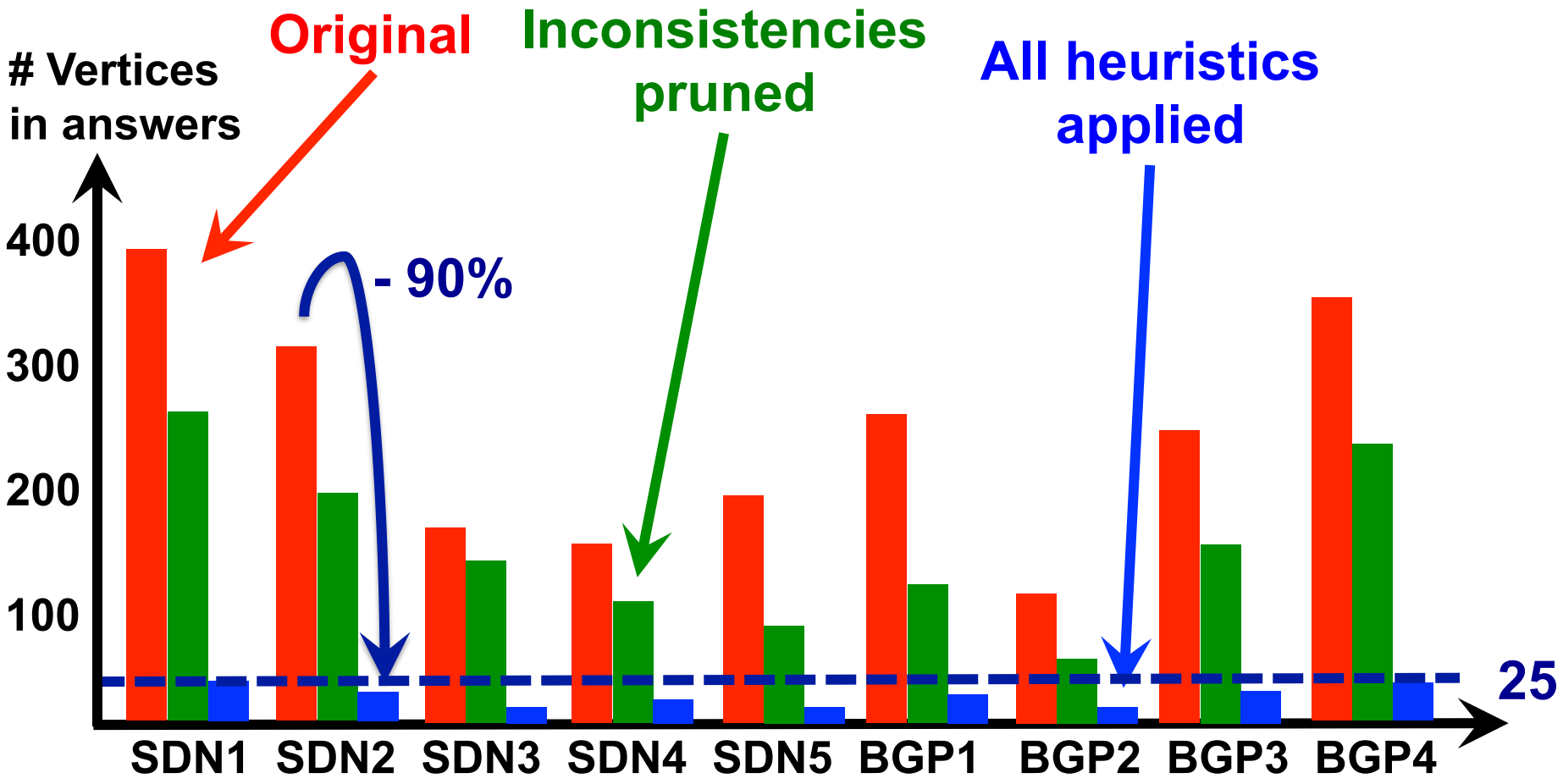
Evaluation: Time to answer a query

- Query turnaround less than one second

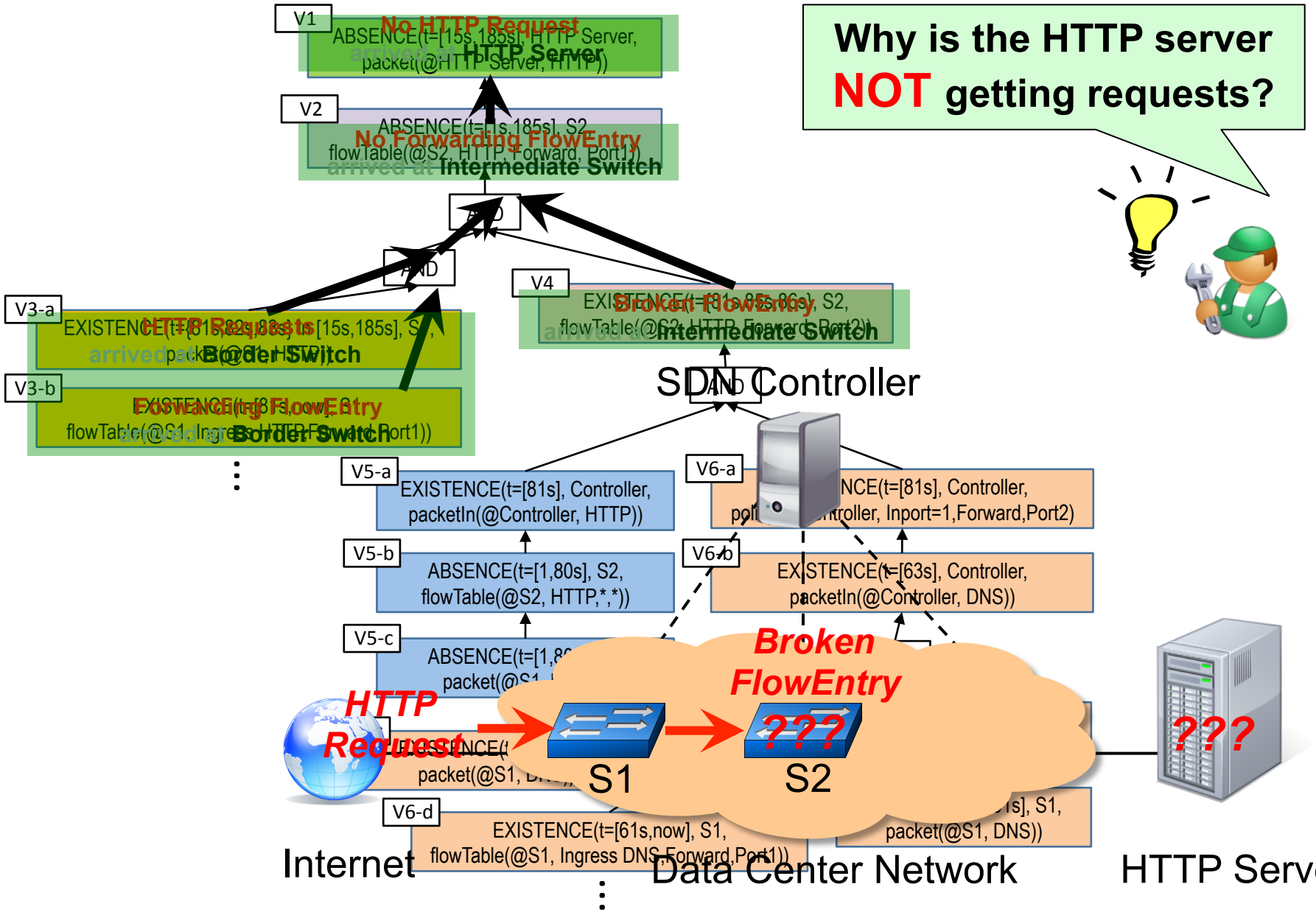


Evaluation: Size of the returned answer

- Heuristics reduce size of the provenance by over 90%
- No answers had more than 25 vertices



Evaluation: How useful are the answers?



- **Goal: Diagnose events with negative symptoms**

  Example: Why is the HTTP server not getting any requests?

- **Approach: Negative Provenance**

  Uses counterfactual reasoning to find all the ways in which the missing event could have occurred. Then Explains why each did not come to pass.

- **Challenge: Explanation can be very large**

  Uses a combination of several heuristics to remove redundancy and improve readability.

- **Implementation: Y!**

  Can be applied to any distributed system.
  Supports both positive and negative provenance.

- **Two case studies: SDN and BGP**

  Provenance is readable and can be computed quickly.

More information: http://snp.cis.upenn.edu/