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!

Why is the HTTP server getting DNS queries?
Approach: Provenance

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

Why is the HTTP server getting DNS queries?

DNS Query arrived at HTTP Server

DNS Query received at Switch

Broken FlowEntry existed at Switch

SDN Controller

Program

Broken FlowEntry

DNS Query

Internet

Data Center Network

HTTP Server
Challenge: Missing events

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

Why is the HTTP server NOT getting requests?
Survey: How common are missing events?

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

Missing events:
- NANOG-user: 52% missing, 48% positive
- Floodlight-dev: 74% missing, 26% positive

Positive events:
- Outages: 83% positive, 17% missing
Approach: Counter-factual reasoning

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

Why did Bob NOT arrive at SIGCOMM?
Result: Debugger for missing events

Why is the HTTP server NOT getting requests?

No HTTP Request arrived at HTTP Server

No Forwarding-FlowEntry installed at Switch

HTTP Request received at Switch

Dropping-FlowEntry existed at Switch

HTTP Request received at Switch

Dropping-FlowEntry existed at Switch

Internet

Data Center Network

Controller

Program

HTTP Request

Dropping-FlowEntry

HTTP Server

???
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?

PacketSent : - PacketReceived, FlowEntry.

PacketSent : - PacketOut.

Step 3: Select events from distributed systems
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

No PacketSent during [t1,now]

PacketSent :- PacketReceived, FlowEntry.

PacketSent
PacketReceived
FlowEntry

time t1 t2 t3 t4 t5 now
Generating negative provenance graphs

- Explanation can be unnecessarily complex

No PacketSent during [t1,now]

No PacketReceived during [t1,t2]
No FlowEntry during [t2,t3]
No PacketReceived during [t3,t4]
No FlowEntry during [t4,t5]
No PacketReceived during [t5,now]

PacketSent
PacketReceived
FlowEntry

Time

t1  t2  t3  t4  t5  now
Generating negative provenance graphs

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

No PacketSent during [t1,now]

No FlowEntry during [t1,now]

PacketSent
PacketReceived
FlowEntry
Generating negative provenance graphs

function QUERY(exist([t_1, t_2], N, t, r))
S ← ∅
for each (+t, N, t, r, c) ∈ Log: t_1 ≤ t ≤ t_2
S ← S ∪ {appear(t, N, t, r, c)}
for each (−t, N, t, r, c) ∈ Log: t_1 ≤ t ≤ t_2
S ← S ∪ {disappear(t, N, t, r, c)}
return S

function QUERY(appear(t, N, t, 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 ← τ. N, τ)}

function QUERY(insert(t, N, τ))
return ∅

function QUERY(derive(t, N, τ, −τ_1, τ_2...))
S ← ∅
for each τ_i: (+τ_i, N, t, r, c) ∈ Log:
S ← S ∪ {appear(t, N, τ_i, c)}
else
t_x ← max t' < t: (+τ, N, t', r, 1) ∈ Log
S ← S ∪ {exist([t_x, t], N, τ_i, c)}
return S

function QUERY(receive(t, N_1 ← N_2, +τ))
t_s ← max t' < t: (+τ, N_2, t', r, 1) ∈ Log
return {send(t_s, N_1 → N_2, +τ), delay(t_s, N_2 → N_1, +τ, t - t_s)}

function QUERY(send(t, N → N′, +τ))
find (+τ, N, t, r, c) ∈ Log
return {appear(t, N, τ, r)}

function QUERY(nexist([t_1, t_2], N, τ))
if ∃τ < t_1 : (−τ, N, t, r, 1) ∈ Log then
t_x ← max t < t_1: (−τ, N, t, r, 1) ∈ Log
return {disappear(t_x, N, τ), nappear([t_x, t], N, τ)}
else return {nappear([0, t], N)}

function QUERY(nderive([t_1, t_2], N, τ, r))
S ← ∅
for (τ_i, I_i) ∈ partition([t_1, t_2], N, τ, r)
S ← S ∪ {nexist(I_i, N, τ_i)}
return S

function QUERY(nsend([t_1, t_2], N, +τ))
if ∃t_1 < t < t_2 : (−τ, N, t, r, 1) ∈ Log then
return {exist([t_1, t], N, τ), nappear([t, t], N, τ)}
else return {nappear([t_1, t_2], N, τ)}

function QUERY(narrive([t_1, t_2], N, −τ, +τ))
if BaseTuple(τ) then
return {ninsert([t_1, t_2], N, τ)}
else if LocalTuple(N, τ) then
return ∪_r ∈ Rules(N).Head(τ) = τ
    {nderive([t_1, t_2], N, τ, r)}
else return {nreceive([t_1, t_2], N, +τ)}

function QUERY(nreceive([t_1, t_2], N, +τ))
S ← ∅, t_0 ← t_1 - Δmax
for each N′ ∈ senders(τ, N):
X ← {t_0 ≤ t ≤ t_2|(+τ, N′, t, r, 1) ∈ Log}
t_x ← t_0
for (i=0; i < |X|; i++)
    S ← S ∪ {nsend((t_x, X_i), N′, +τ), narrive([t_1, t_2], N′ → N, X_i, +τ)}
    t_x ← X_i
S ← S ∪ {nsend([t_1, t_2], N′, +τ)}
return S

function Q(narrive([t_1, t_2], N_1 → N_2, t_0, +τ))
find (+τ, N_2, t_3, (N_1, t_0, 1) ∈ Log
return {send(t_0, N_1 → N_2, +τ), delay(t_0, N_1 → N_2, +τ, t_3 - t_0)}
Challenge: Explanation is complicated!

Why NOT ... ?
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
Explanation size reduction
Explanation usability
Readability: How to simplify the provenance?

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

Prune

No chicken.

No egg.

No chicken.

No Packet arrived at Server

No Packet arrived at S1

No Packet arrived at S2

No Packet arrived at S3

... Summarize
Readability: Other heuristics

- Prune logical inconsistencies.
- Prune failed assertions.
- Branch coalescing.
- Application-specific invariants.
- Summarize transient event chains.
- Summarize super-vertex.
Readability: Concise explanations

Why NOT ... ?
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
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?

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.
Goal: Diagnose missing events

Approach: Counter-factual reasoning

Challenge: Too many explanations

Background: Provenance

Generating Negative Provenance

Improving readability

R-tree indexing

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

Heuristics reduce size of the provenance by over 90%
No answers had more than 25 vertices
Evaluation: How useful are the answers?

Why is the HTTP server NOT getting requests?

No HTTP Request arrived at HTTP Server

No Forwarding FlowEntry arrived at Intermediate Switch

HTTP Requests arrived at Border Switch

Forwarding FlowEntry arrived at Border Switch

Broken FlowEntry arrived at Intermediate Switch

No HTTP Request arrived at HTTP Server

No Forwarding FlowEntry arrived at Intermediate Switch

HTTP Requests arrived at Border Switch

Forwarding FlowEntry arrived at Border Switch

Broken FlowEntry arrived at Intermediate Switch

No HTTP Request arrived at HTTP Server

No Forwarding FlowEntry arrived at Intermediate Switch

HTTP Requests arrived at Border Switch

Forwarding FlowEntry arrived at Border Switch

Broken FlowEntry arrived at Intermediate Switch

No HTTP Request arrived at HTTP Server

No Forwarding FlowEntry arrived at Intermediate Switch

HTTP Requests arrived at Border Switch

Forwarding FlowEntry arrived at Border Switch

Broken FlowEntry arrived at Intermediate Switch

No HTTP Request arrived at HTTP Server

No Forwarding FlowEntry arrived at Intermediate Switch

HTTP Requests arrived at Border Switch

Forwarding FlowEntry arrived at Border Switch

Broken FlowEntry arrived at Intermediate Switch
- **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/](http://snp.cis.upenn.edu/)