USC Viterbi
School of Engineering
Department of Computer Science

Sibyl: A Practical Internet Route Oracle

Ethan Katz-Bassett (University of Southern California)

with:
Pietro Marchetta (University of Napoli Federico II),
Matt Calder, Yi-Ching Chiu (USC), Italo Cunha (UFMG),
Harsha Madhyastha (Michigan), Vasileios Giotsas (CAIDA)

Supported By: Google NSF

Sibyl: A Practical Internet Route Oracle

Ethan Katz-Bassett (University of Southern California)

with:
Pietro Marchetta (University of Napoli Federico II),
Matt Calder, Yi-Ching Chiu (USC), Italo Cunha (UFMG),
Harsha Madhyastha (Michigan), Vasileios Giotsas (CAIDA)

Supported By: Google NSF

Traceroute Widely Used by Operators and Researchers

"The number one go-to tool is traceroute."

NANOG Network operators troubleshooting tutorial, 2009.

- Lots of use cases:
  - topology and AS relationships
  - route performance and inflation
  - location of congestion
  - outages
  - prefix hijacks
  - etc

Traceroute Widely Used by Operators and Researchers

“The number one go-to tool is traceroute.”

NANOG Network operators troubleshooting tutorial, 2009.

- Lots of use cases:
  - topology and AS relationships
  - route performance and inflation
  - location of congestion
  - outages
  - prefix hijacks
  - etc
- Lots of vantage points:
  - PlanetLab, Ark
  - RIPE Atlas, BISmark
  - traceroute servers
  - MobiPerf, Dasu
  - RIPE RIS, RouteViews
  - etc

Traceroute Is Extremely Limited

“The number one go-to tool is traceroute.”

NANOG Network operators troubleshooting tutorial, 2009.

- Lots of use cases:
  - topology and AS relationships
  - route performance and inflation
  - location of congestion
  - outages
  - prefix hijacks
  - etc
- Lots of vantage points:
  - PlanetLab, Ark
  - RIPE Atlas, BISmark
  - traceroute servers
  - MobiPerf, Dasu
  - RIPE RIS, RouteViews
  - etc

- But traceroute only supports one query:
  “What is the path from vantage point s to destination d?”

YouTube
0:00 / 6:66

How I’d Like to Use Vantage Points

Query for:

- Path from a certain network

What’s the path from AT&T mobile in LA to YouTube?

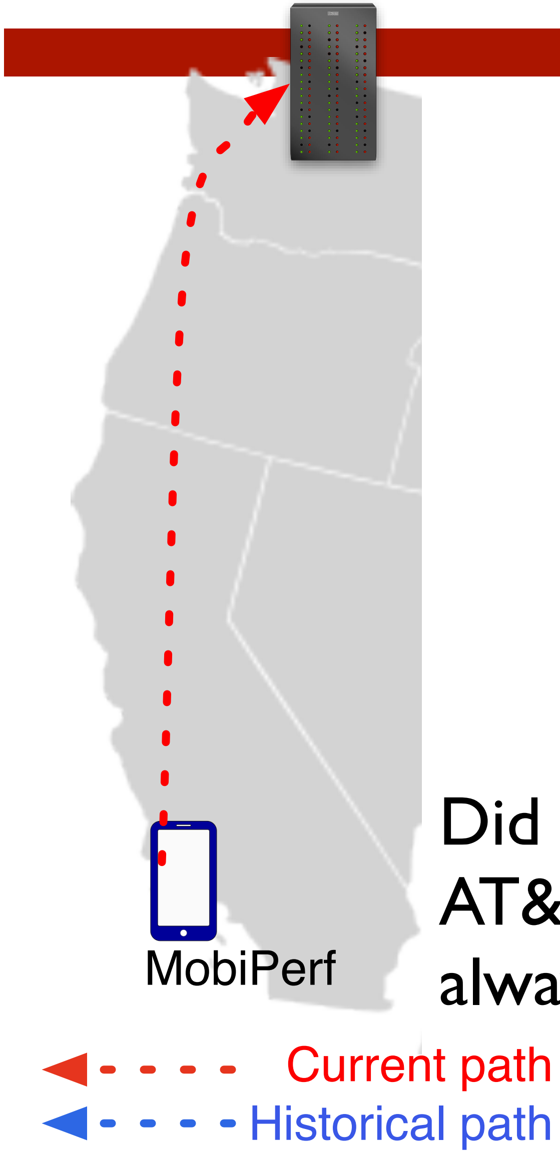
◀- - - - Current path

◀- - - - Historical path

How I'd Like to Use Vantage Points

Query for:

- Path from a certain network
- Historical path

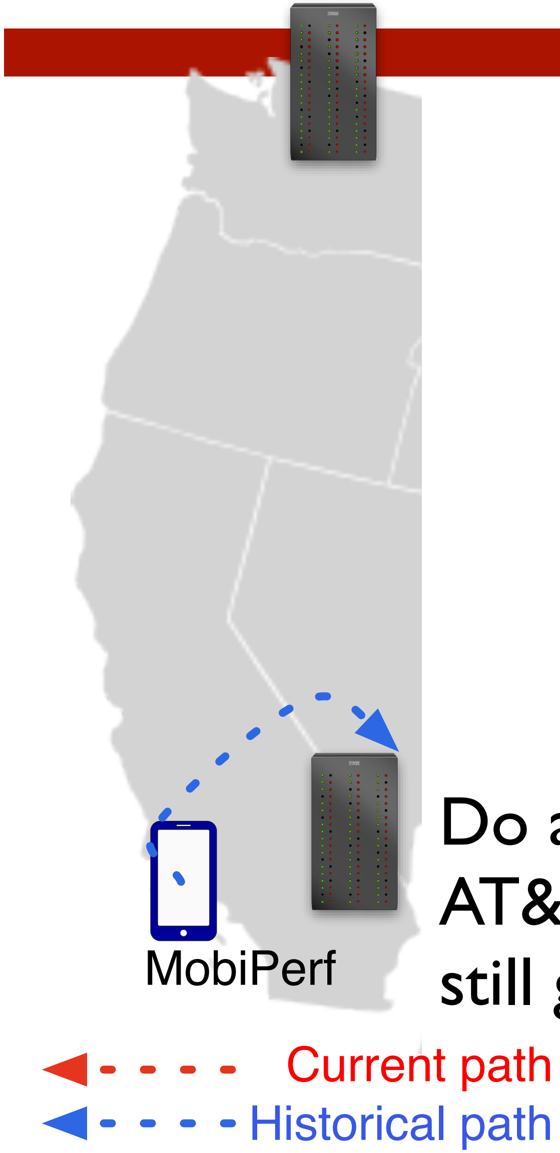


Did paths from AT&T mobile in LA always go to Seattle?

How I'd Like to Use Vantage Points

Query for:

- Path from a certain network
- Historical path
- Paths through series of hops

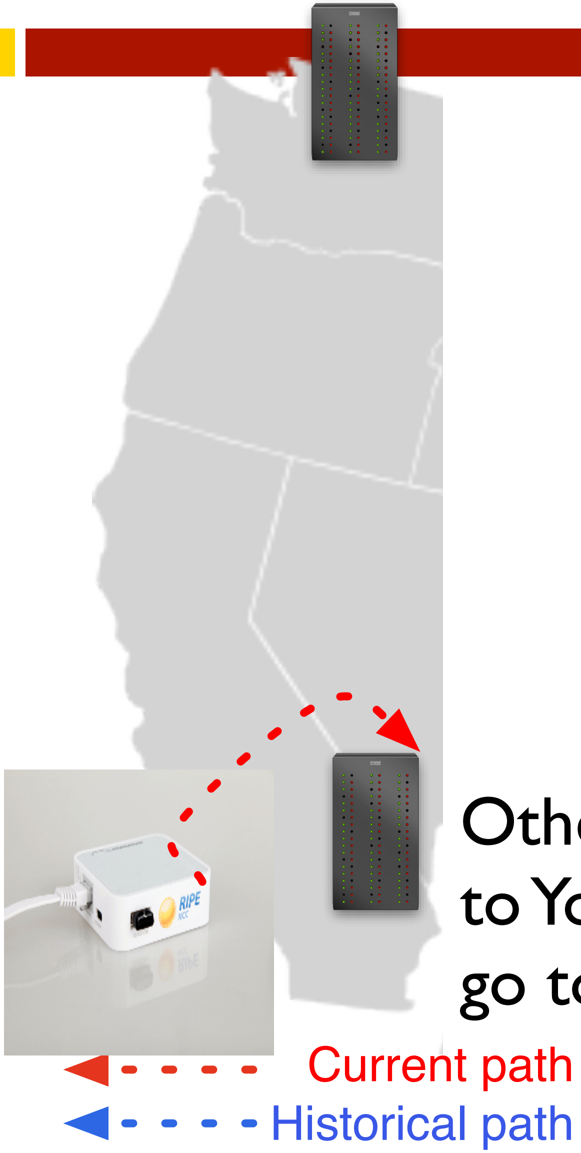


Do any paths through AT&T in LA to YouTube still go to LA server?

How I’d Like to Use Vantage Points

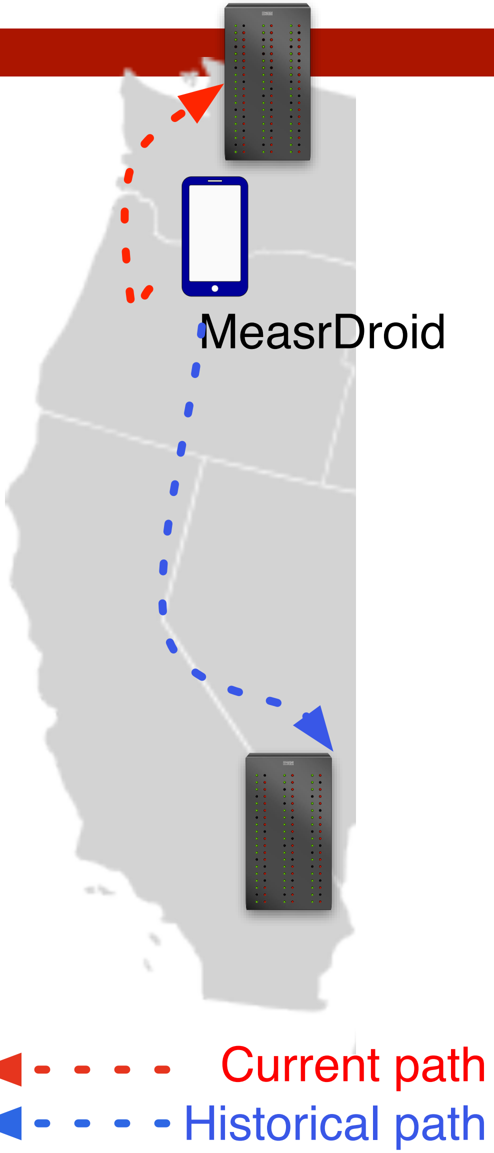
Query for:

- Path from a certain network
- Historical path
- Paths through series of hops
- Relationship between historical path and current path



Other paths that went to YouTube LA and now go to YouTube Seattle?

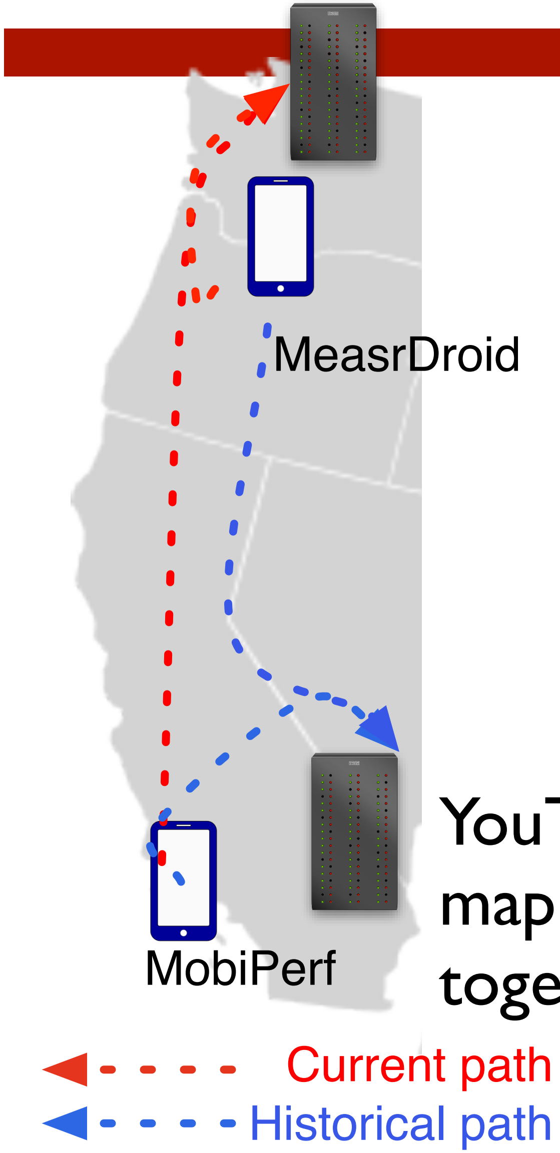
How I’d Like to Use Vantage Points



Query for:

- Path from a certain network
- Historical path
- Paths through series of hops
- Relationship between historical path and current path

How I’d Like to Use Vantage Points

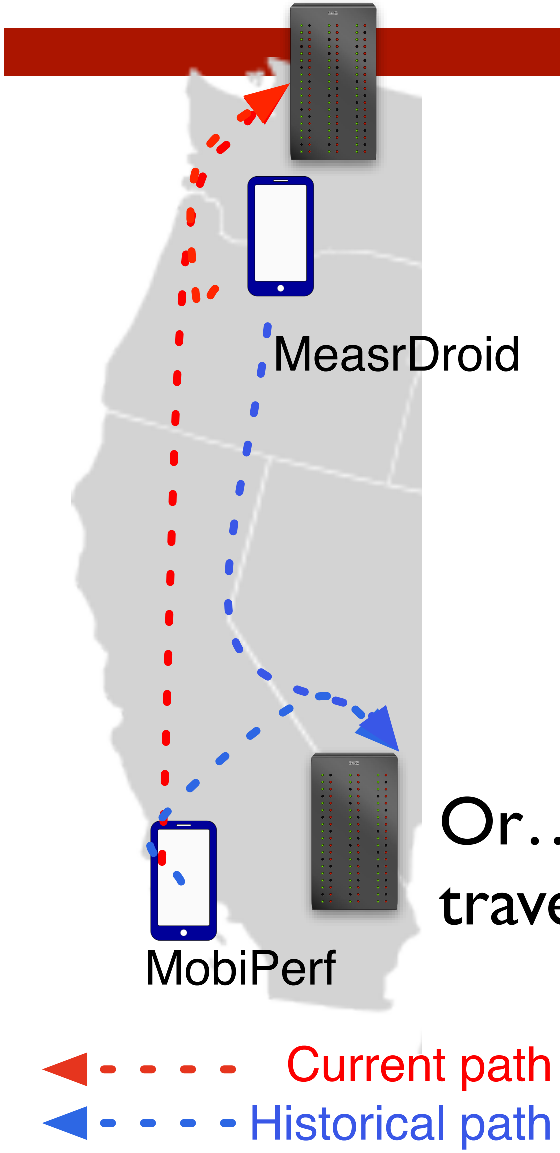


Query for:

- Path from a certain network
- Historical path
- Paths through series of hops
- Relationship between historical path and current path

YouTube appears to map these clients together.

How I’d Like to Use Vantage Points

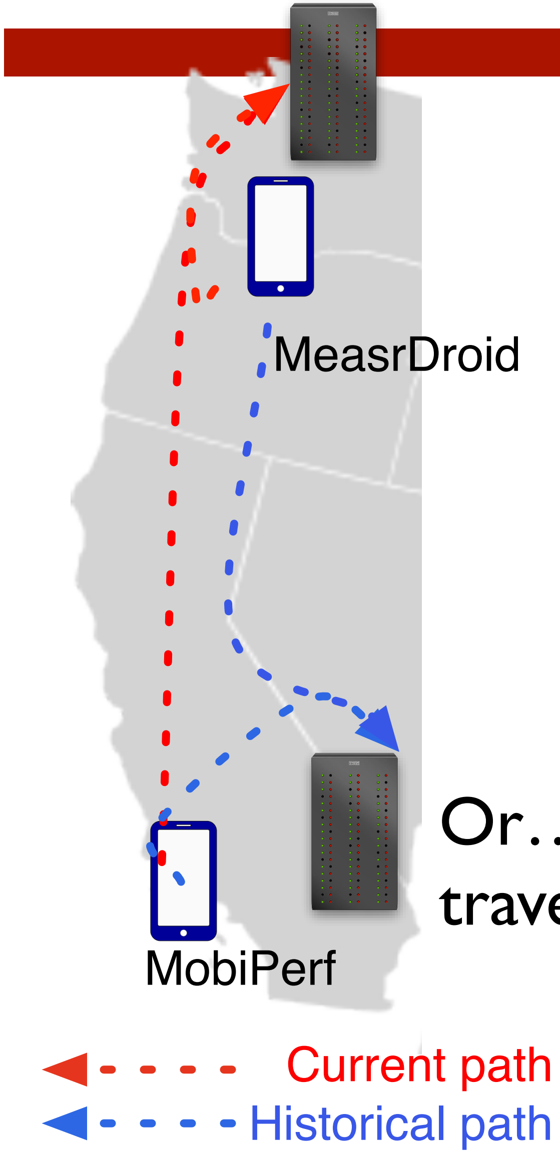


Query for:

- Path from a certain network
- Historical path
- Paths through series of hops
- Relationship between historical path and current path

Or…other paths traversing AT&T-NTT?

How I’d Like to Use Vantage Points

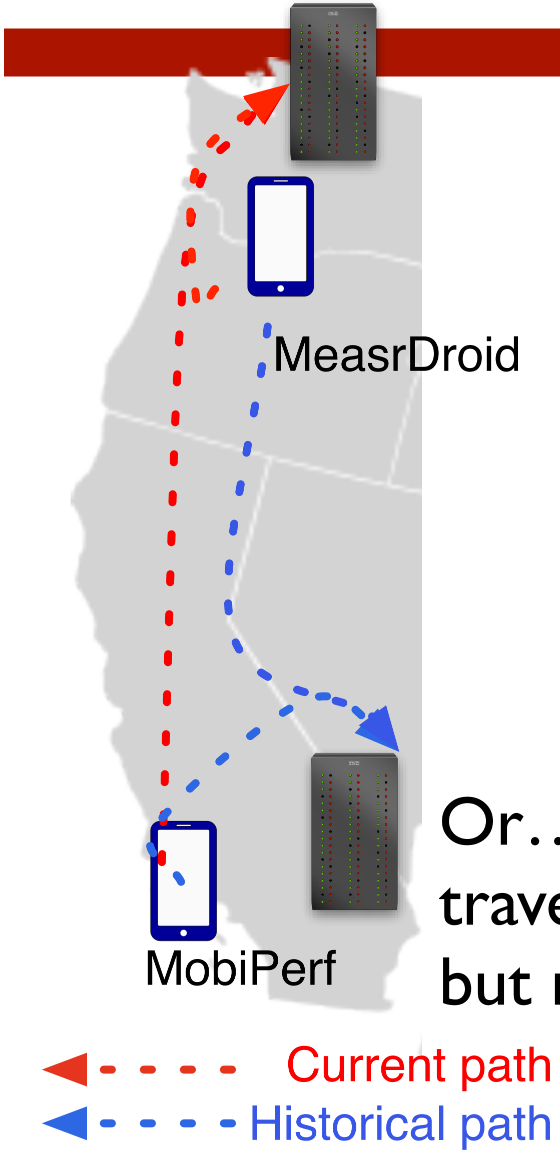


Query for:

- Path from a certain network
- Historical path
- Paths through series of hops
- Relationship between historical path and current path

Or…other paths traversing GTT-NTT?

How I’d Like to Use Vantage Points

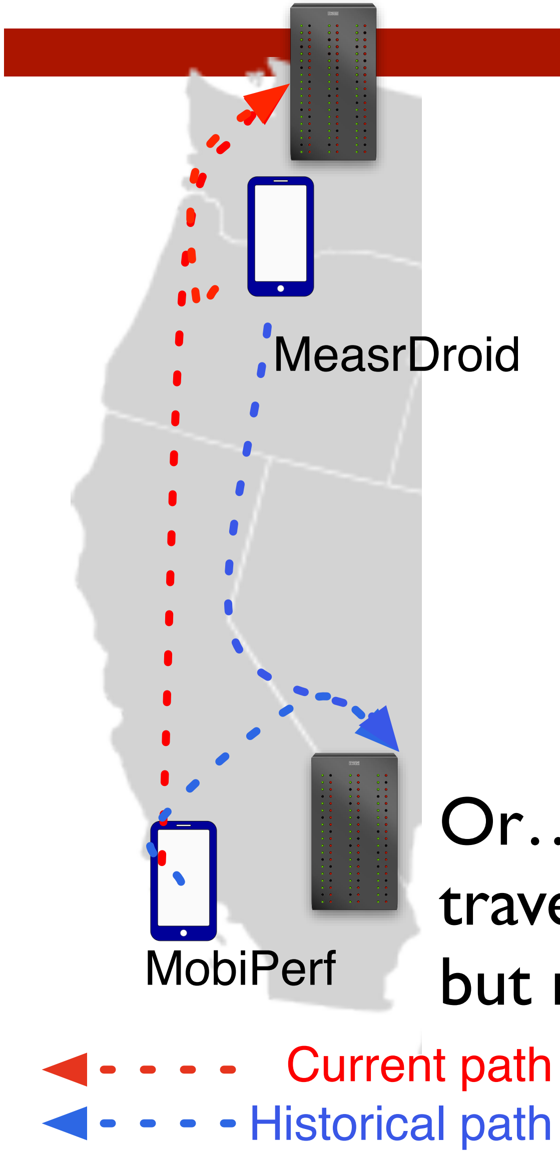


Query for:

- Path from a certain network
- Historical path
- Paths through series of hops
- Relationship between historical path and current path

Or…other paths traversing NTT but not AT&T or GTT?

How I'd Like to Use Vantage Points



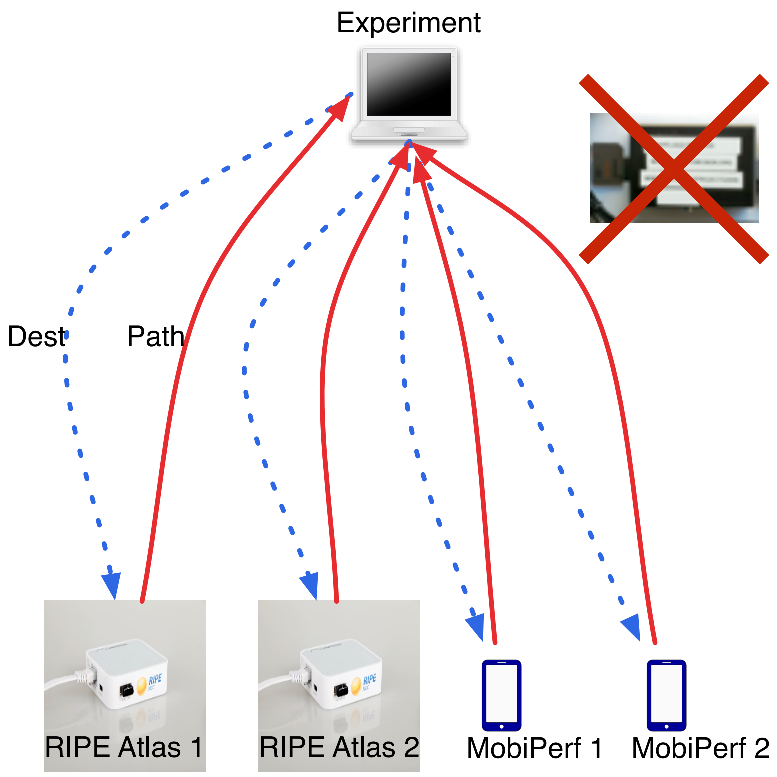
Query for:

- Path from a certain network
- Historical path
- Paths through series of hops
- Relationship between historical path and current path

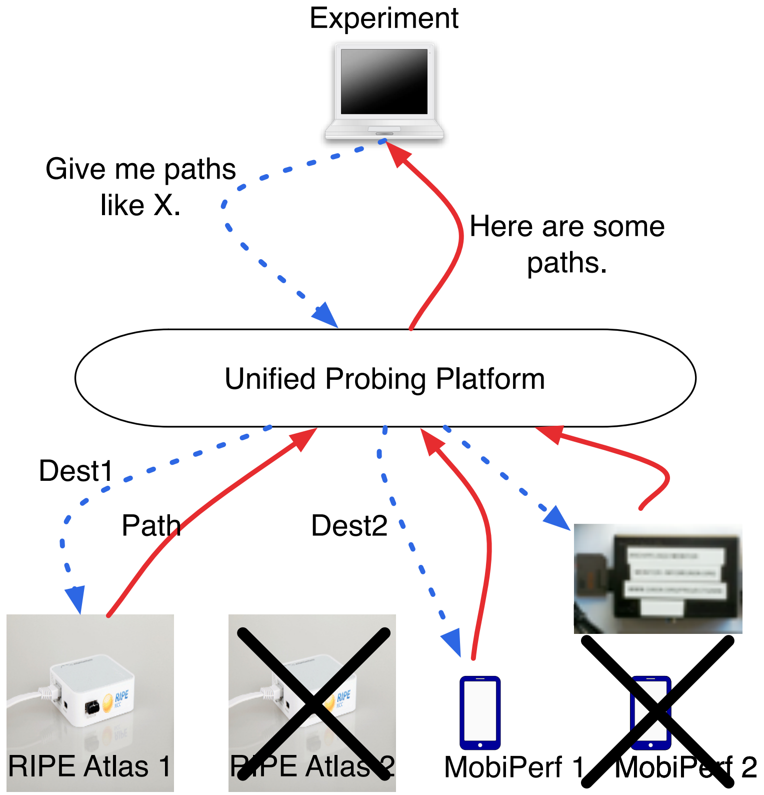
Or…other paths traversing NTT but not LA or SEA?

2014 Measurement vs 2016 Measurement

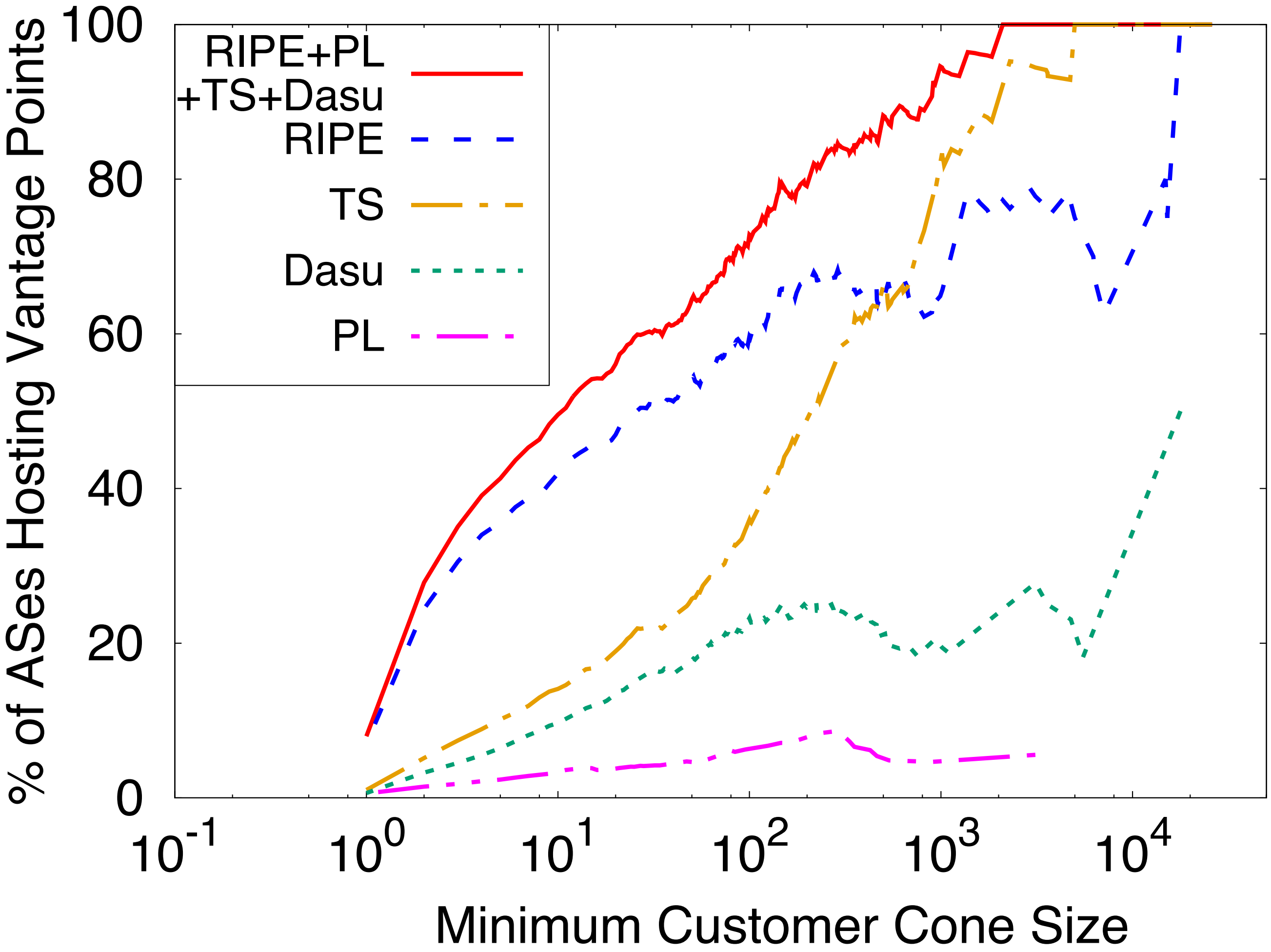
- What I do



- What I want to do

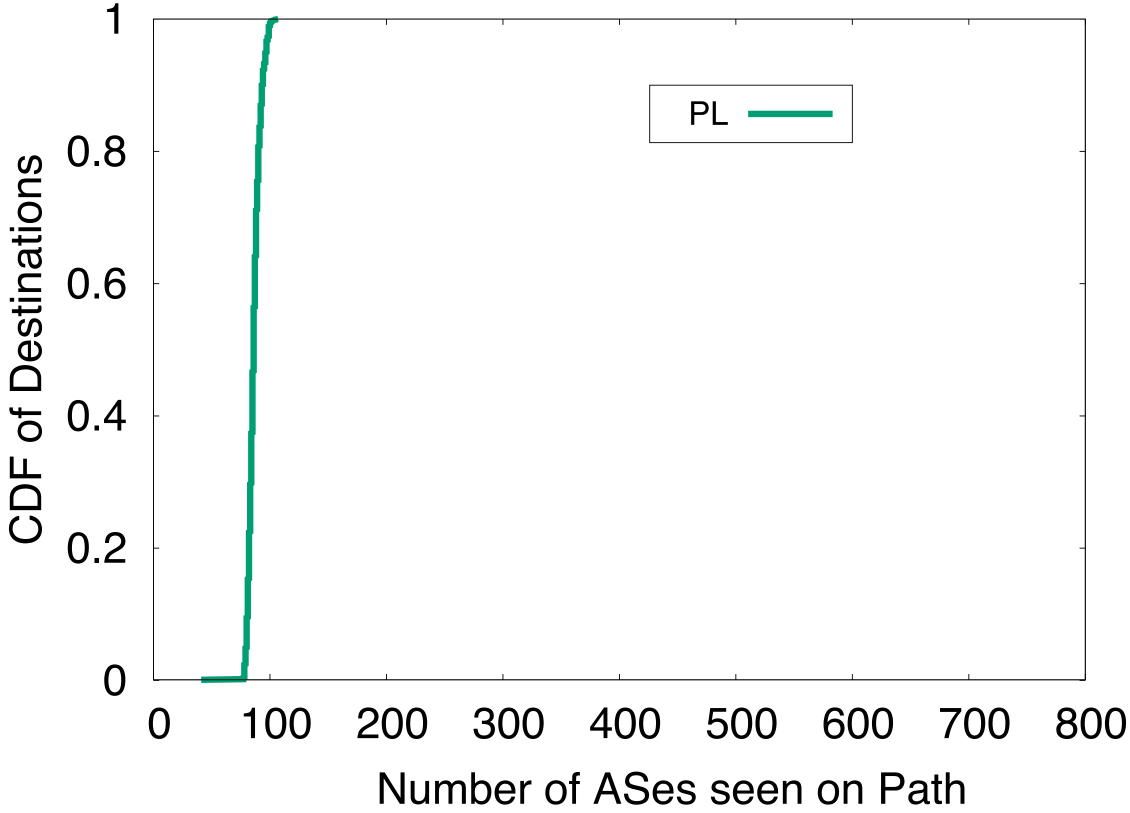


Benefit of Combining Platforms



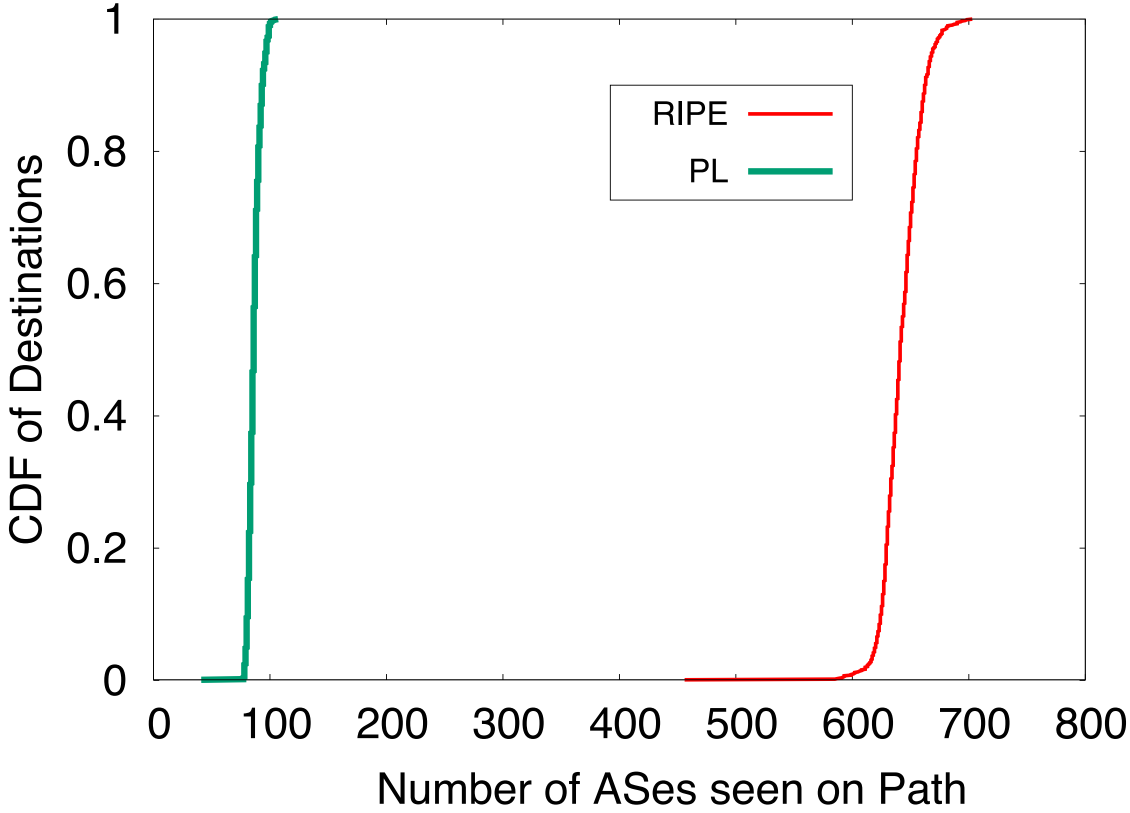
- Combining platforms improves coverage

Challenge of Combining Platforms



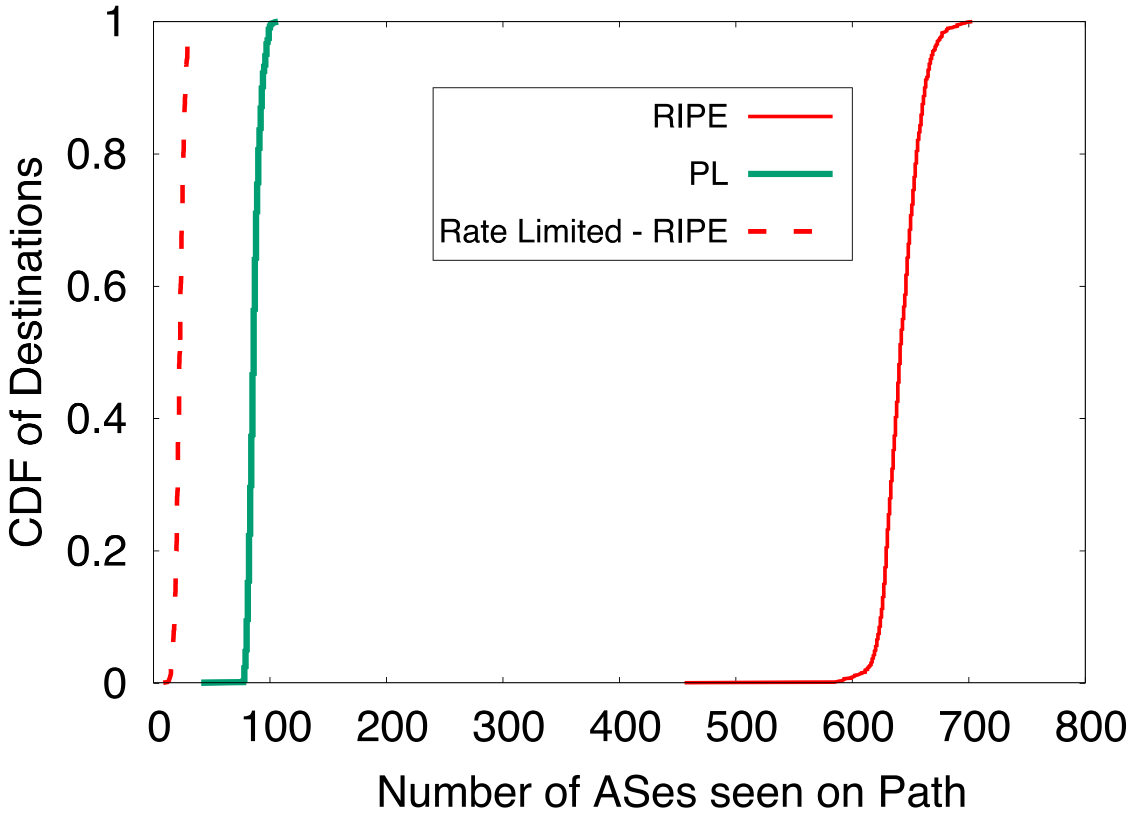
- Combining platforms improves coverage

Challenge of Combining Platforms



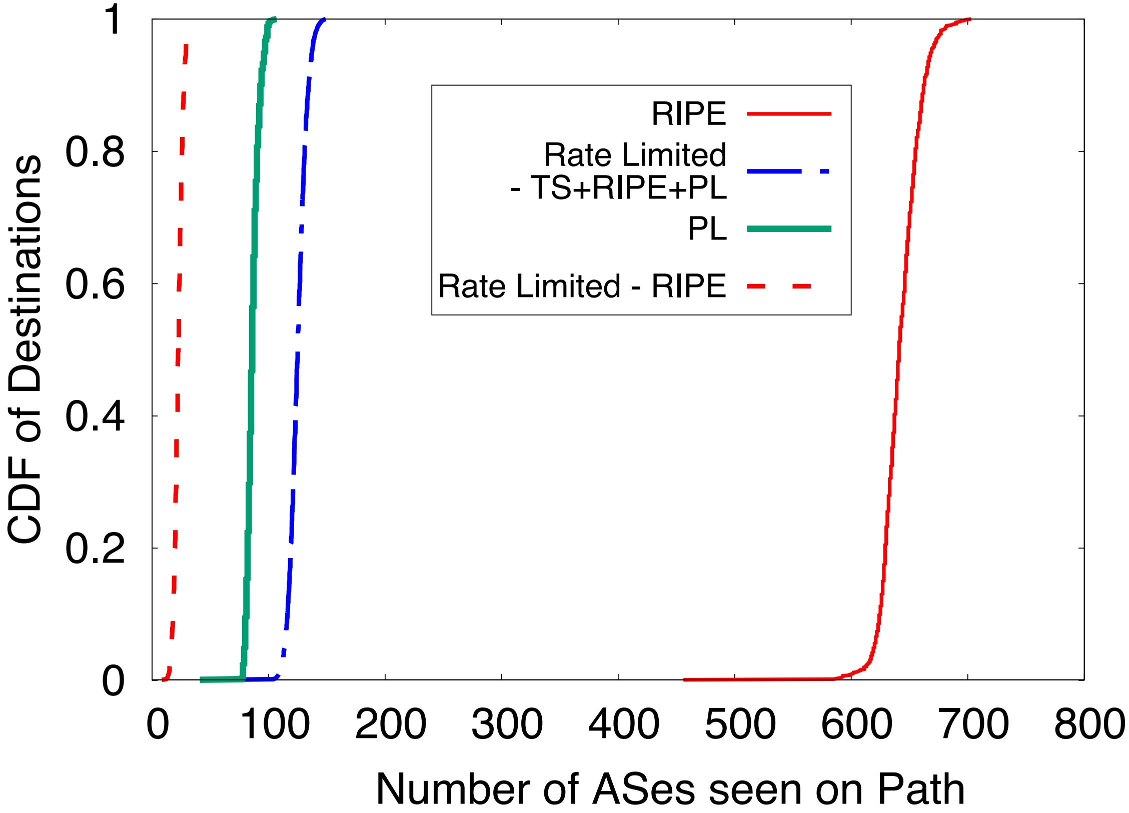
- Combining platforms improves coverage

Challenge of Combining Platforms



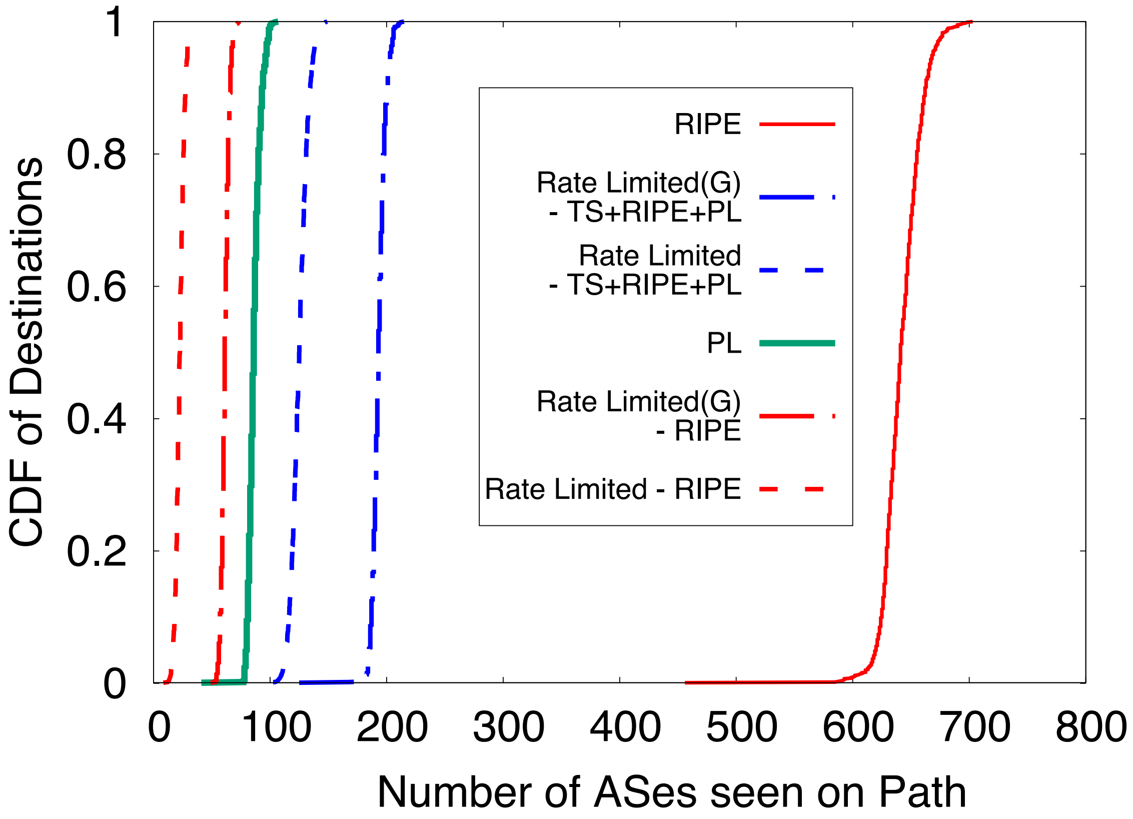
- Combining platforms improves coverage
- … but exhaustive probing is infeasible

Challenge of Combining Platforms



- Combining platforms improves coverage
- … but exhaustive probing is infeasible

Challenge of Combining Platforms



- Combining platforms improves coverage
- … but exhaustive probing is infeasible
- Rate limits mean you have to be smart about what to issue

Goals

- Take advantage of diverse vantage points
- Efficient use of probing budgets
  - High rate, low diversity vantage points like Ark and PL
  - Low rate, high diversity vantage points like Atlas and LG
- Support rich queries

Service queries:

- as if we had traceroutes from all vantage points to all Internet destinations
- even though probing budgets are very restricted

Related Work

- IXPs: Mapped?, RocketFuel, Lord of the Links, Reverse Traceroute, etc
  - Issued measurements likely to traverse particular links
- iPlane
  - Predicted source/destination paths
- TopHat
  - Unified historical and current
  - Multiple testbeds?
  - Does it exist anymore?
- MPlane
- Srikanth's talk
- Others?

Goals

- Take advantage of diverse vantage points
- Efficient use of probing budgets
  - High rate, low diversity vantage points like Ark and PL
  - Low rate, high diversity vantage points like Atlas and LG
- Support rich queries

Service queries:

- as if we had traceroutes from all vantage points to all Internet destinations
- even though probing budgets are very restricted

Optimize Use of Probing Budget

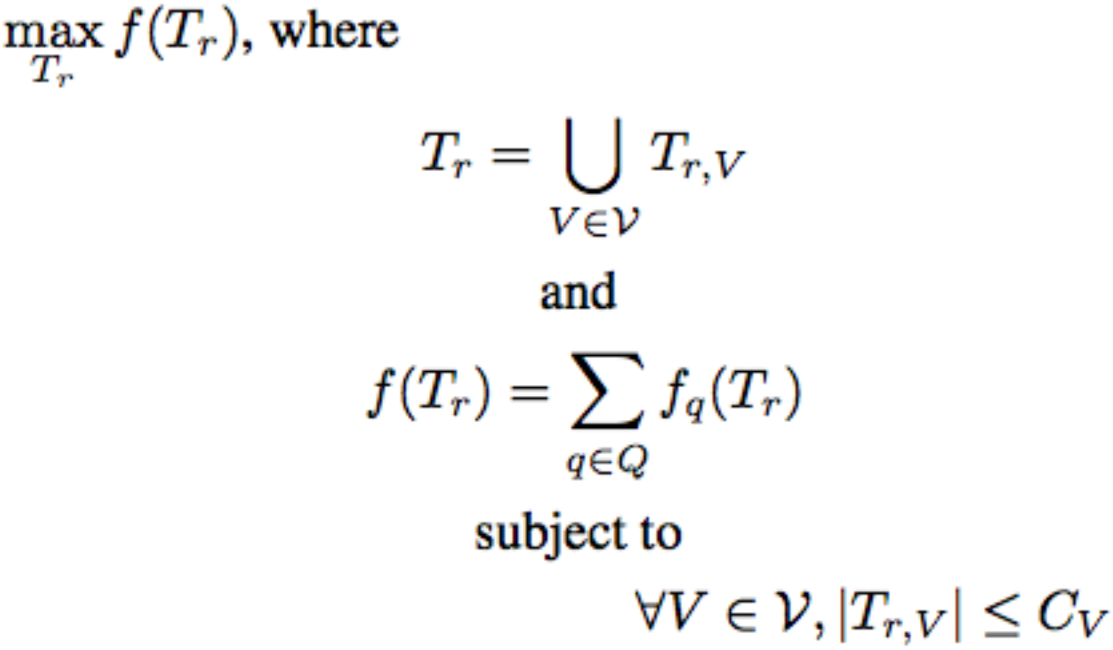
Goal:

- In each round, allocate probing budget to best serve queries

Optimize Use of Probing Budget

Goal:

- In each round, allocate probing budget to best serve queries



$$\max_{T_r} f(T_r), \text{ where}$$

$$T_r = \bigcup_{V \in \mathcal{V}} T_{r,V}$$

and

$$f(T_r) = \sum_{q \in Q} f_q(T_r)$$

subject to

$$\forall V \in \mathcal{V}, |T_{r,V}| \leq C_V$$

Optimize Use of Probing Budget

Goal:

- In each round, allocate probing budget to best serve queries

$$\max_{T_r} f(T_r), \text{ where}$$

$$T_r = \bigcup_{V \in \mathcal{V}} T_{r,V}$$

and

$$f(T_r) = \sum_{q \in Q} f_q(T_r)$$

subject to

$$\forall V \in \mathcal{V}, |T_{r,V}| \leq C_V$$

- Max utility of traceroutes

Optimize Use of Probing Budget

Goal:

- In each round, allocate probing budget to best serve queries

$$\max_{T_r} f(T_r), \text{ where}$$

$$T_r = \bigcup_{V \in \mathcal{V}} T_{r,V}$$

and

$$f(T_r) = \sum_{q \in Q} f_q(T_r)$$

subject to

$$\forall V \in \mathcal{V}, |T_{r,V}| \leq C_V$$

- Max utility of traceroutes
- T_r: set of traceroutes in the round is union of those from each platform *V*

Optimize Use of Probing Budget

Goal:

- In each round, allocate probing budget to best serve queries

$$\max_{T_r} f(T_r), \text{ where}$$

$$T_r = \bigcup_{V \in \mathcal{V}} T_{r,V}$$

and

$$f(T_r) = \sum_{q \in Q} f_q(T_r)$$

subject to

$$\forall V \in \mathcal{V}, |T_{r,V}| \leq C_V$$

- Max utility of traceroutes
- *T_r*: set of traceroutes in the round is union of those from each platform *V*
- Utility of set of traceroutes *T_r* is sum of utility in matching each query *q*

Optimize Use of Probing Budget

Goal:

- In each round, allocate probing budget to best serve queries

$$\max_{T_r} f(T_r), \text{ where}$$

$$T_r = \bigcup_{V \in \mathcal{V}} T_{r,V}$$

$$\text{and}$$

$$f(T_r) = \sum_{q \in Q} f_q(T_r)$$

$$\text{subject to}$$

$$\forall V \in \mathcal{V}, |T_{r,V}| \leq C_V$$

- Max utility of traceroutes
- T_r: set of traceroutes in the round is union of those from each platform *V*
- Utility of set of traceroutes T_r is sum of utility in matching each query *q*
- Each platform *V* has a rate limit budget C_V

Challenges in Optimization

Goal:

- In each round, allocate probing budget to best serve queries

Need to solve (rest of talk):

- What is query language?
- How to find paths to match queries, given we can't issue every traceroute (or know the results before issuing)?
- How to solve the optimization efficiently?

Query language

- How to express what paths you care about?
- Sibyl in its current state:
  - Regular expressions over hops
    - All evaluation with AS-level hops, but internal representation is PoPs
  - Supports two types of queries:
    - Existence query: Give me (at least) one path that matches
      - □ A path that goes through Sprint on the way to USC?
        Sprint-.*-USC$
    - Diversity query: Give me as diverse a set of matching paths as possible
      - □ Suspected problem on peering between GTT and Level3?
        \(^.*-(GTT-Level3|Level3-GTT)-.*$\)
      - □ Actually, NTT-GTT too! Other GTT peers?
        \(^.*-[^NTT-Level3]-GTT-[^NTT-Level3]-.*$\)
- ***Thoughts?***

Challenges in Optimization

Goal:

- In each round, allocate probing budget to best serve queries

Need to solve (rest of talk):

- *What is query language? Regular expressions over hops.*

- How to find paths to match queries, given we can’t issue every traceroute (or know the results before issuing)?

- How to solve the optimization efficiently?

Finding Paths to Match Queries

1. Find candidate (v_1 to d_2) by splicing existing traceroutes

2. Predict probability that traceroute (v_1 to d_2) matches q

Finding Paths to Match Queries

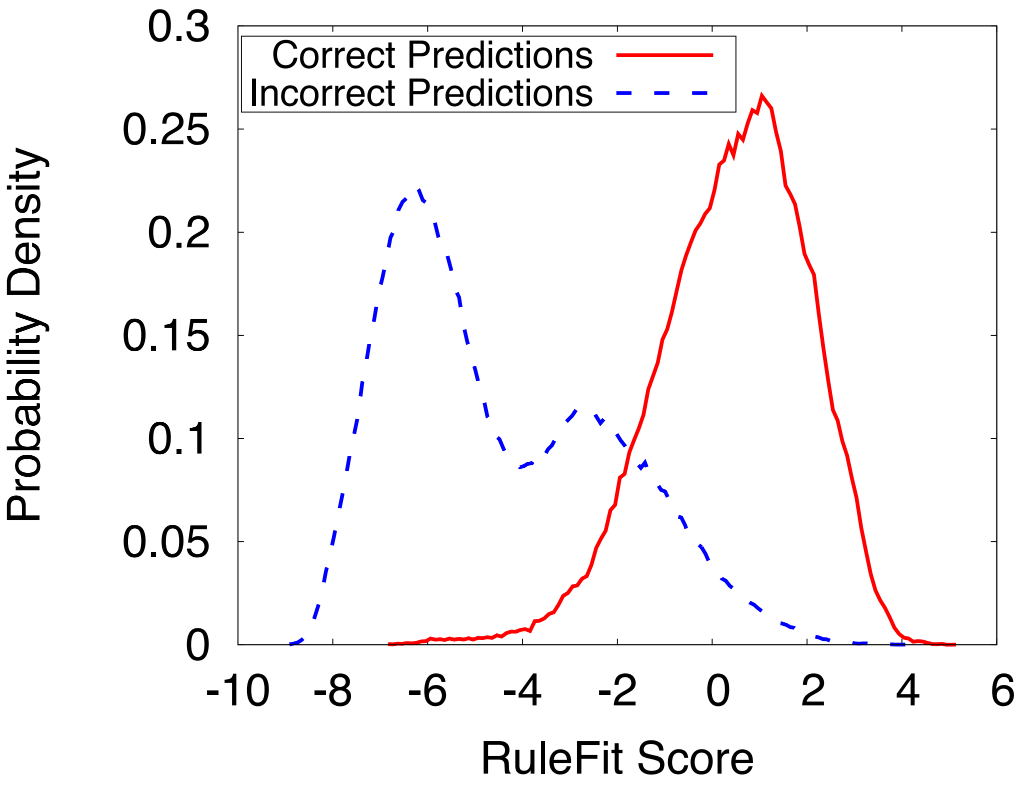
1. Find candidate (v_1 to d_2) by splicing existing traceroutes
   1. Find historical traceroute t_1 from a vantage point v_1 to destination d_1 that matches a prefix q_p of q
   2. Find historical traceroute t_2 from a vantage point v_2 to destination d_2 that matches a prefix q_p of q s.t.:
      1. t_1 and t_2 intersect at a common PoP p
      2. $v_1 \ldots p \ldots d_2 = q_p q_s = q$
   3. Nominate traceroute (v_1 to d_2) as a candidate to match q
2. Predict probability that traceroute (v_1 to d_2) matches q

Finding Paths to Match Queries

1. Find candidate (v_1 to d_2) by splicing existing traceroutes
   a. Find historical traceroute t_1 from a vantage point v_1 to destination d_1 that matches a prefix q_p of q
   b. Find historical traceroute t_2 from a vantage point v_2 to destination d_2 that matches a prefix q_p of q s.t.:
      i. t_1 and t_2 intersect at a common PoP p
      ii. $v_1 \ldots p \ldots d_2 = q_p q_s = q$
   c. Nominate traceroute (v_1 to d_2) as a candidate to match q
2. Predict probability that traceroute (v_1 to d_2) matches q
   a. Generate all possible spliced paths from v_1 to d_2
   b. Rate how likely each spliced path is to be correct
   c. Probability of matching q is sum of likelihood of the spliced paths that match

How Likely is a Spliced Path Correct?

- Train system to recognize likely vs unlikely splices
  - Features include:
    - AS relationships at splice point
    - AS path similarity with other possible splices
    - AS path inflation vs shortest prediction
- System scores spliced path based on features



- Evaluation shows a positive score from system usually means the prediction is correct

Challenges in Optimization

Goal:

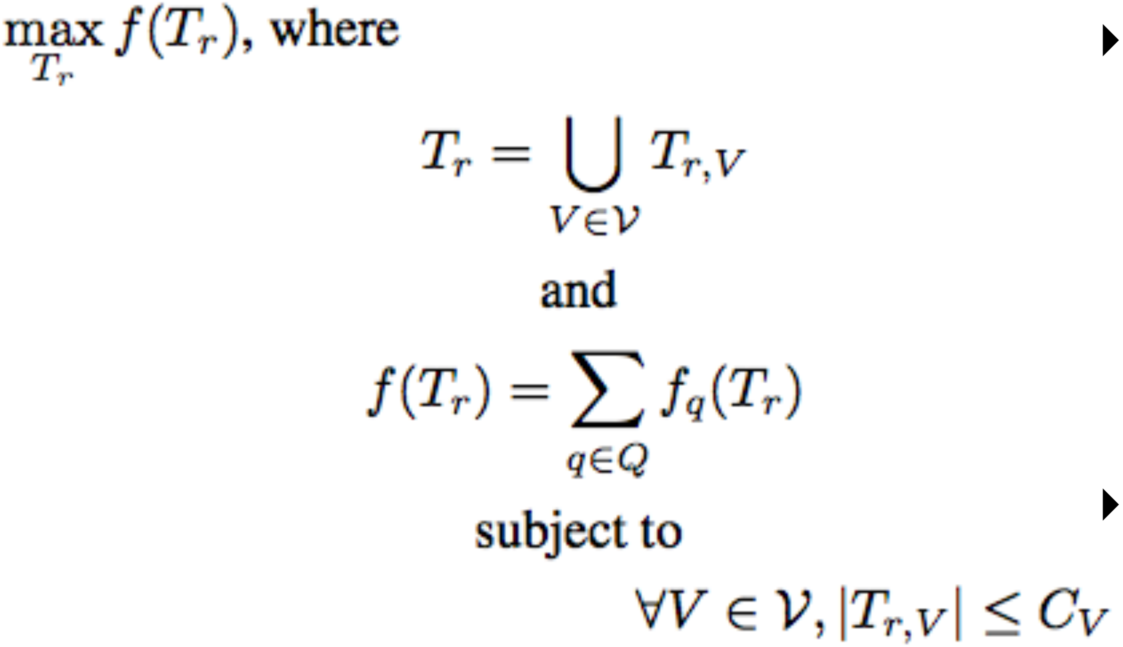
- In each round, allocate probing budget to best serve queries

Need to solve (rest of talk):

- *What is query language? Regular expressions over hops.*
- *How to find paths to match queries, given we can't issue every traceroute (or know the results before issuing)? Splice existing traceroutes to predict new ones, learn which predictions are good.*
- How to solve the optimization efficiently?

Optimize Use of Probing Budget

- In each round, allocate probing budget to best serve queries
- Solve greedily
  - Greedy heuristic known to perform well for submodular function subject to partition constraints



$$\max_{T_r} f(T_r), \text{ where}$$

$$T_r = \bigcup_{V \in \mathcal{V}} T_{r,V}$$

and

$$f(T_r) = \sum_{q \in Q} f_q(T_r)$$

subject to

$$\forall V \in \mathcal{V}, |T_{r,V}| \leq C_V$$

- Submodular function: function on sets that exhibits diminishing returns
- Partition constraints: each traceroute subject to at most one constraint

Challenges in Optimization

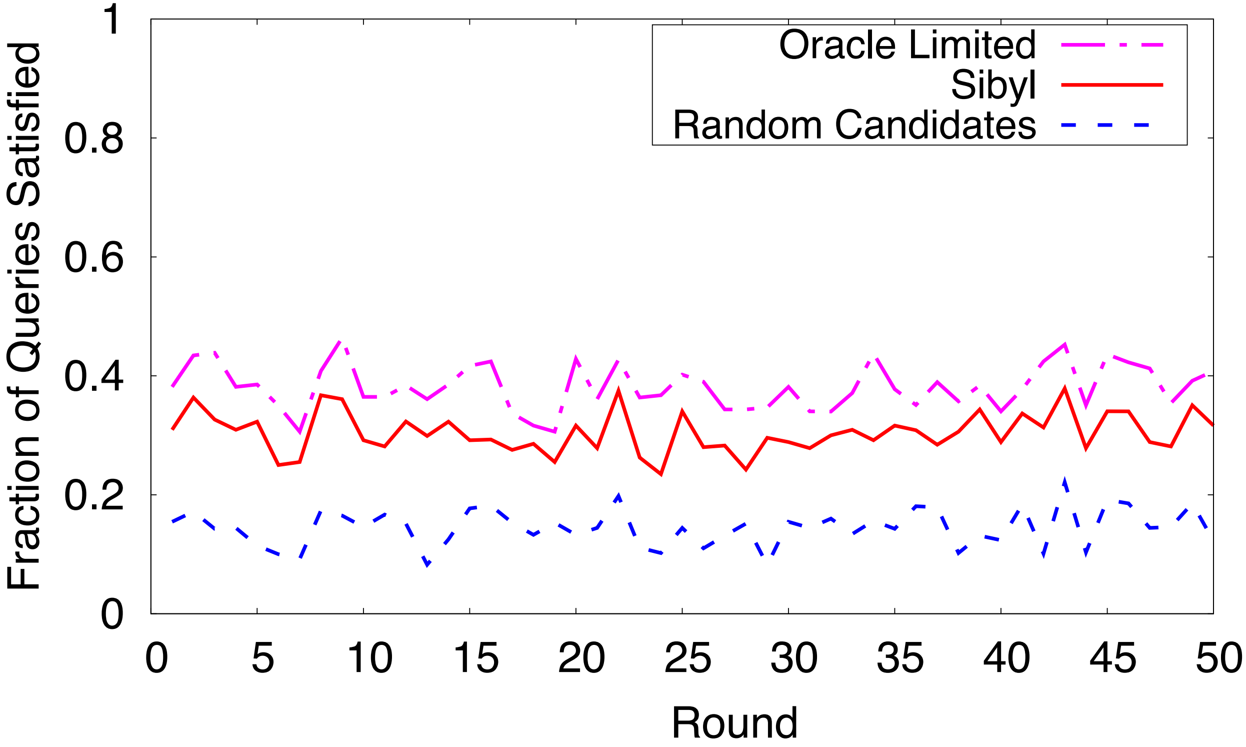
Goal:

- In each round, allocate probing budget to best serve queries

Need to solve (rest of talk):

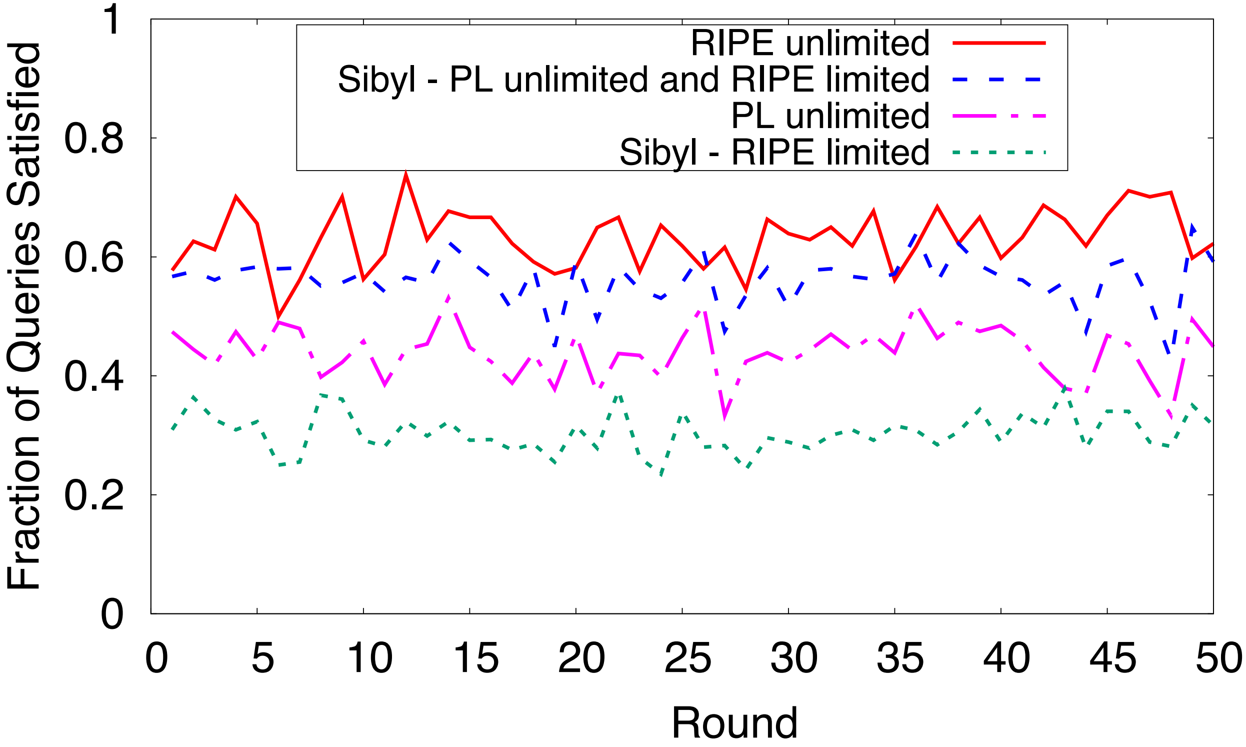
- *What is query language? Regular expressions over hops.*
- *How to find paths to match queries, given we can't issue every traceroute (or know the results before issuing)? Splice existing traceroutes to predict new ones, learn which predictions are good.*
- *How to solve the optimization efficiently? Greedily.*

Evaluation: Does Sibyl efficiently allocate probing budget? (I)



- **Prediction is effective**: Sibyl satisfies 80% as many queries as an Oracle that knows all paths but is subject to rate limits
- **Important to assess likelihood:** Sibyl satisfies 107% more than Randomly selecting among spliced candidates

Evaluation: Does Sibyl efficiently allocate probing budget? (II)



- By combining diverse but rate limited RIPE Atlas vantage points with the smaller number of PlanetLab vantage points, Sibyl matches nearly as many queries as RIPE Atlas without rate limits

Conclusion

- Lots of route vantage points exist,
  but our interface to them is extremely limited
  - Little unification of historical and live measurements
  - Each platform is independent
  - Can only ask one question:
    "*What is the path from here to there?*"
- Sibyl: A unified platform for Internet path queries
  - Goal: Get relevant paths with properties of interest, without limiting queries by testbed, source, or destination
  - Approach: Predict paths based on previously measured ones, and optimize budget use based on predictions