Network Tomography:

Inverse Methods for Network State Monitoring from End-to-End Measurements

Ting He

Associate Professor, CSE@Penn State

Students: Yilei Lin, Yudi Huang (Penn State), Liang Ma (Imperial College)

Collaborators: Tom La Porta (Penn State), Don Towsley (Umass), Kin K. Leung (Imperial College), Ananthram Swami (ARL)

Overview: What is network tomography

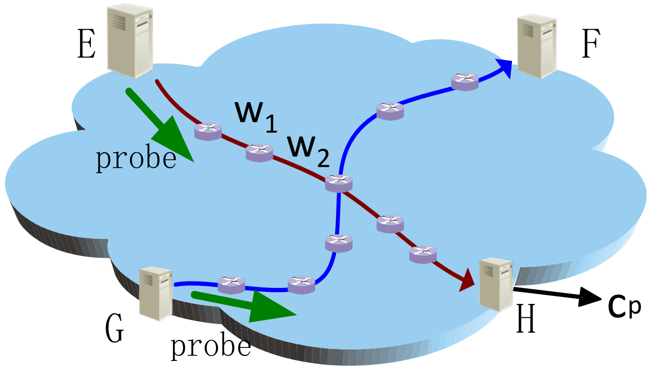
> Tomography refers to imaging by sections or sectioning, through the use of any kind of penetrating wave.
>
> --- Wikipedia

- **Network tomography:** Using *external observations* to infer *internal network state*

Network state:

w

Measurement:



Q(w)=c

Inference:

Given **Q(.), c, w**=?

A "CT scan" for the network!

Motivation: Why needing network state

- **Network state (e.g., topology, link/node performances) provides useful information for many applications**
  - Routing
  - Caching
  - Service placement
  - Client-server association
  - Load balancing
  - Trouble shooting
  - Overlay management
  - …

Motivation: Why inferring network state

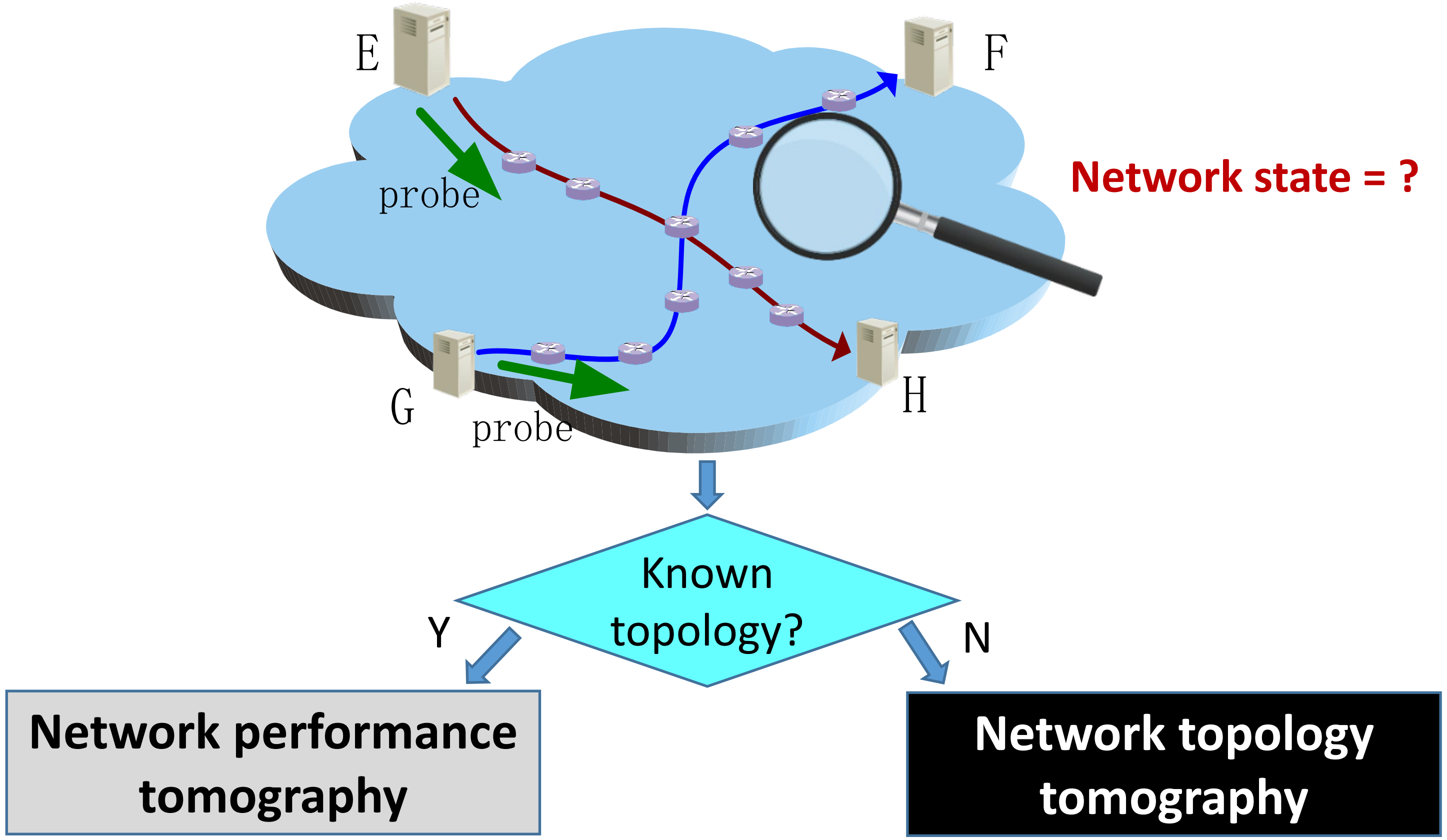
- **Modern networks are increasingly**
  - **Complex**: Expanded functionalities (e.g., network function virtualization, content-based networking)
  - **Heterogeneous**: Different technologies (e.g., cellular/WiFi) and ownerships (e.g., Internet, private-public cloud, coalition)

- **Network state is not always observable**
  - Use protocols to collect state information (e.g., SNMP, OpenFlow) → admin privilege
  - Use ICMP to measure internal state (e.g., traceroute) → supportive internal nodes

Q: Is it possible to infer **network state** from **end-to-end measurements**? If so, how?

Overview: Branches of network tomography

- Using *external observations* to infer *internal network state*



Network state = ?

Known topology?

Y

N

Network performance tomography

Network topology tomography

Using *internal observations* to infer *external state*

- e.g., using link loads to infer traffic matrix (**Origin-Destination matrix tomography**)

Network Performance Tomography

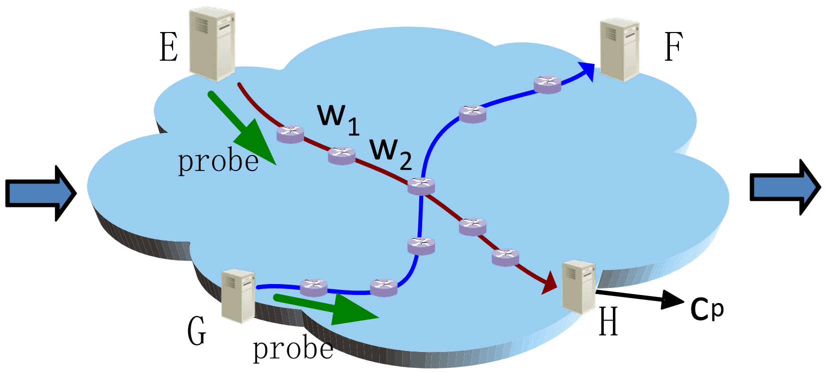
Identifiability and Measurement Design

Problem statement

- **Network performance tomography:** Given routing, inferring *link metrics* from *path measurements*.

Network state: **Measurement:** **State inference:**

w



Given **Q(.), c,**
w=?

Q(w)=c

- Examples: To infer
  - link delay/jitter: **Q(w)** = w_1+…+w_n → additive metric
  - link success rate (loss): **Q(w)** = w_1*…*w_n → additive metric (taking $-\log(\cdot)$)
    - assuming link independence
  - link availability (failure): **Q(w)** = w_1*…*w_n → Boolean metric
    - 1 – available, 0 – failed
  - link bandwidth: **Q(w)** = min(w_1,…,w_n) → min metric

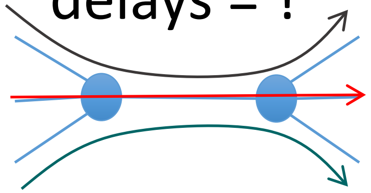
Existing approaches

- Deterministic approach
  - Model **link metrics as constants**
  - *Best-effort inference* of link metrics from path metrics

- Statistical approach
  - Model **link metrics as random variables**
  - *Best-effort inference* of link parameters from realizations of path metrics

Measure average path delays

Average link delays = ?

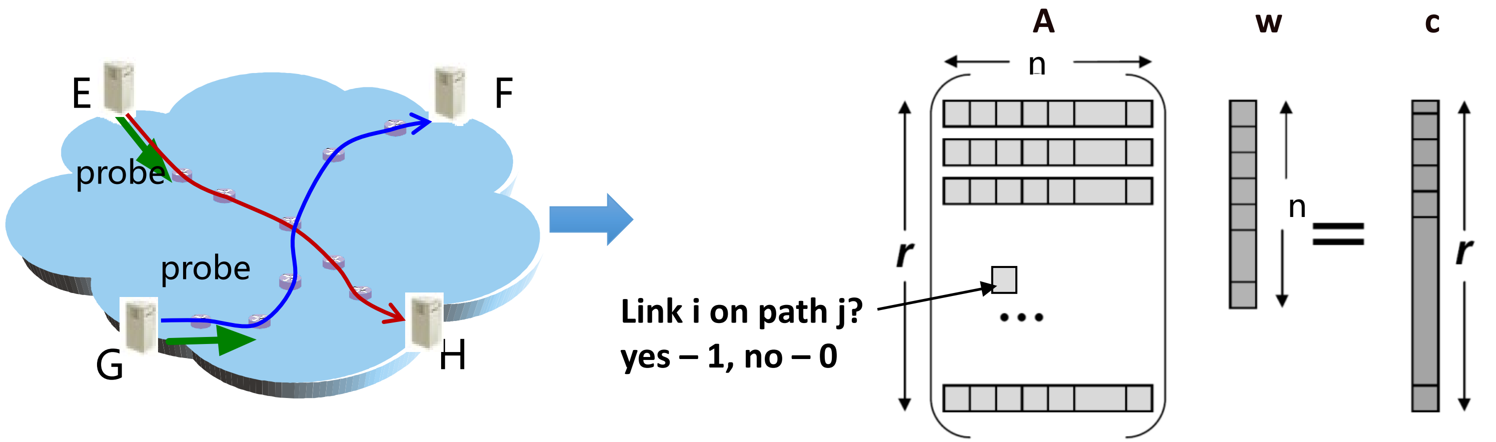


Measure realizations of path delays

Assumption: Q(.) is invertible, a.k.a. network state is identifiable.

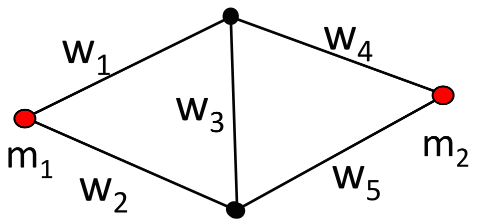
Challenge: Lack of identifiability

Assume: constant, additive link metrics (e.g., mean delay/jitter, log delivery ratio).



Given measurement matrix **A** and path metrics **c**, link metrics **w** =?

- Example:



$w_1+w_4=c_1$

$w_2+w_5=c_2$

$w_1+w_3+w_5=c_3$

$w_2+w_3+w_4=c_4$

$$\Rightarrow \underset{\mathbf{A}}{\begin{pmatrix} 1&0&0&1&0 \\ 0&1&0&0&1 \\ 1&0&1&0&1 \\ 0&1&1&1&0 \end{pmatrix}} \underset{\mathbf{w}}{\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{pmatrix}} = \underset{\mathbf{c}}{\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}}$$

w is not identifiable

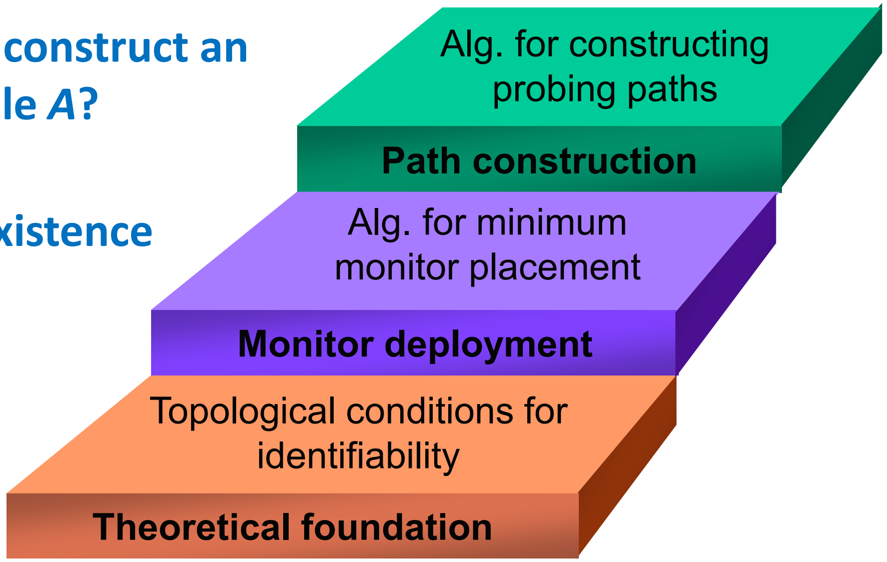
Solution: Measurement design

- Lack of identifiability is common
  - Out of the $O(n^2)$ routing paths between n nodes, only $O(n \log n)$ are linearly independent [Chen04SIGCOMM]
- Fix: Strategically design measurement paths to ensure identifiability

How to construct an invertible *A*?

How to guarantee existence of an invertible *A*?

When is measurement matrix *A* invertible?



Alg. for constructing probing paths

Path construction

Alg. for minimum monitor placement

Monitor deployment

Topological conditions for identifiability

Theoretical foundation

[Chen04SIGCOMM] Chen, et al., “An Algebraic Approach to Practical and Scalable Overlay Network Monitoring,” SIGCOMM 2004.

Problem 1: Identifiability condition

Assumptions

- Known network topology G=(V, L)
- Unknown link metrics that are additive and constant (e.g., mean delay)
- Measurements along arbitrary cycle-free paths between monitors

Objective

Necessary and sufficient conditions for the network state to be identifiable, i.e., all the link metrics are uniquely determined by path metrics

G is identifiable iff rank(A) = #links

- Difficult to verify
- No insight

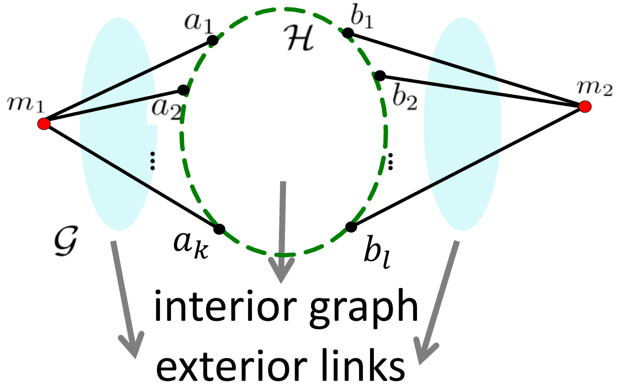
Conditions in terms of topology & monitor placement

Results on identifiability conditions

Negative result:

Theorem. $\mathcal{G}$ $(|\mathcal{L}|>1)$ is **unidentifiable using two monitors**, regardless of the network topology and the monitor placement.

Proof idea: Exterior links are unidentifiable even if all interior link metrics are known.



interior graph

exterior links

$$w_{m_1,a_1} + w_{b_1,m_2} = c_{1,1}$$

…

$$w_{m_1,a_k} + w_{b_l,m_2} = c_{k,l}$$

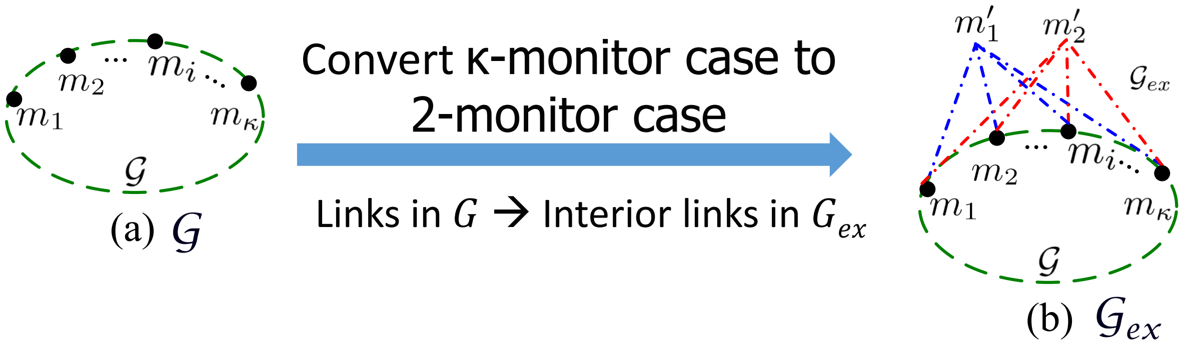
rank = $k + l - 1$

< #exterior links

Positive result:

Theorem. Using κ (κ ≥3) monitors, $\mathcal{G}$ is identifiable iff **the extended graph $\mathcal{G}_{ex}$ is 3-vertex-connected.**

Proof idea: (1) Establish conditions to identify interior links with 2 monitors; (2) Convert all-link identifiability in $\mathcal{G}$ to interior-link identifiability in $\mathcal{G}_{ex}$.



Convert κ-monitor case to 2-monitor case

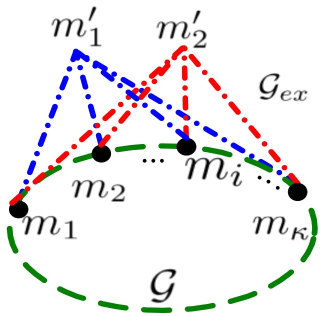
Links in G → Interior links in G_{ex}

Given a topology $\mathcal{G}$ and locations of κ (κ ≥3) monitors, **identifiability can be tested in O(|V|+|L|) time**

- $\mathcal{G}_{ex}$ can be decomposed into tri-connected components in O(|V|+|L|) [Hopcroft73]

[Hopcroft73] J. E. Hopcroft and R. E. Tarjan, “Dividing a graph into triconnected components,” SIAM Journal on Computing, vol. 2, pp. 135–158, 1973.

Problem 2: Monitor placement

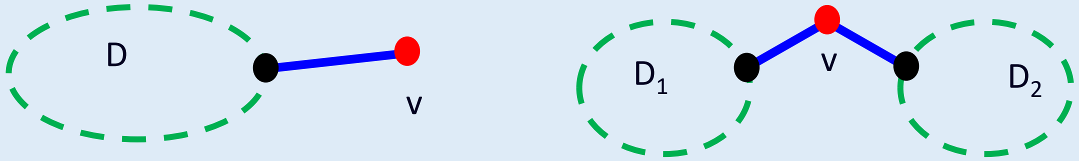


Objective

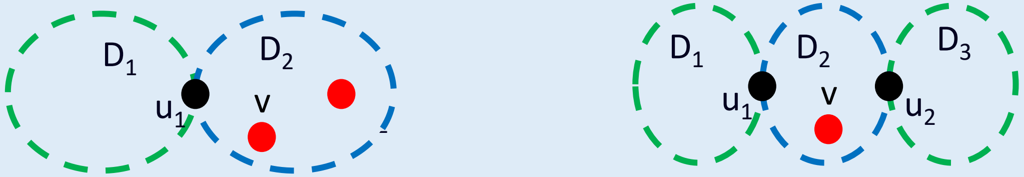
Place minimum #monitors to identify G, i.e., making G_{ex} 3-vertex-connected.

General rules:

(1) Nodes with degree 1 or 2 must be monitors



(2) Each subgraph with ≥ 3 nodes must have at least 3 monitors or connecting points with rest of G



(3) Total #monitors ≥ 3

Algorithm: *Minimum Monitor Placement (MMP)*

1) Select nodes with degree 1 or 2 as monitors
2) Decompose $\mathcal{G}$ into biconnected [Tarjan72] and then triconnected components [Hopcroft73], and ensure each component has ≥ 3 monitors or connecting points
3) Ensure total #monitors ≥ 3

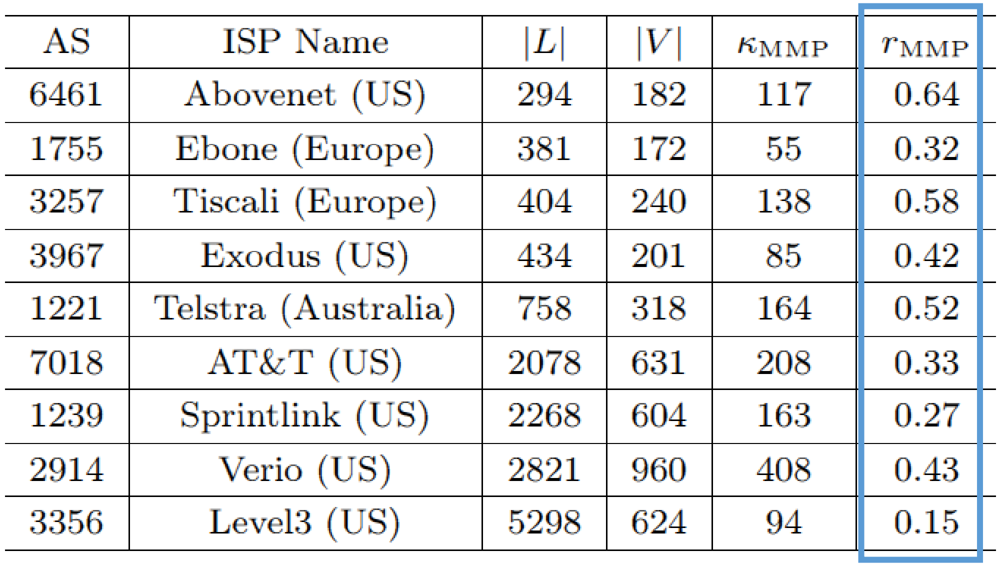
Theorem. MMP places the **minimum #monitors** to identify all links in $\mathcal{G}$

Time Complexity: O(|V|+|L|)

[Hopcroft73] J. E. Hopcroft and R. E. Tarjan, "Dividing a graph into triconnected components," SIAM Journal on Computing, vol. 2, pp. 135–158, 1973.
[Tarjan72] R. Tarjan, "Depth-first search and linear graph algorithms," SIAM Journal on Computing, vol. 1, pp. 146-160, 1972.

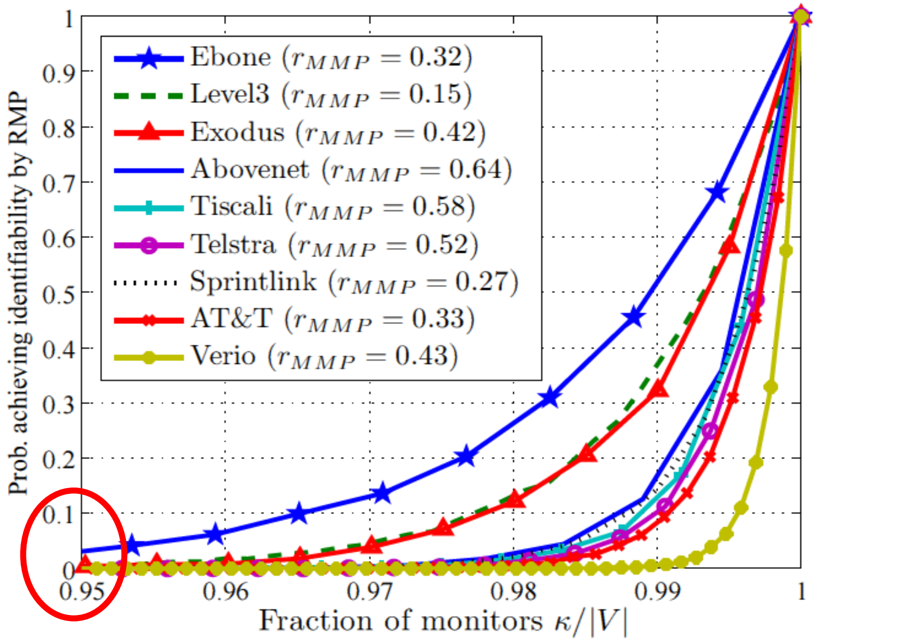
MMP vs. Random monitor placement

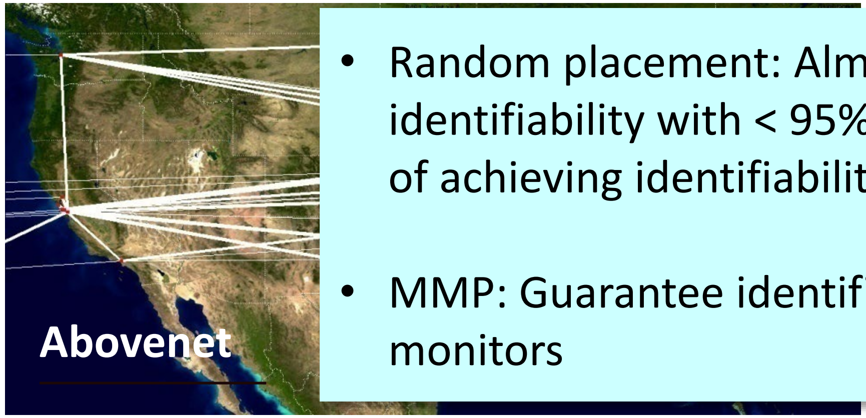
Fraction of monitors placed by MMP

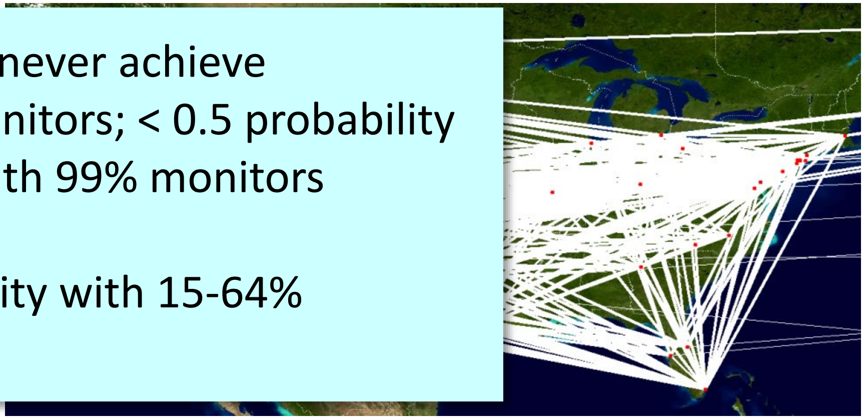


AS	ISP Name	$\|L\|$	$\|V\|$	κ_{MMP}	r_{MMP}
6461	Abovenet (US)	294	182	117	0.64
1755	Ebone (Europe)	381	172	55	0.32
3257	Tiscali (Europe)	404	240	138	0.58
3967	Exodus (US)	434	201	85	0.42
1221	Telstra (Australia)	758	318	164	0.52
7018	AT&T (US)	2078	631	208	0.33
1239	Sprintlink (US)	2268	604	163	0.27
2914	Verio (US)	2821	960	408	0.43
3356	Level3 (US)	5298	624	94	0.15

Probability of achieving identifiability by random placement







- Random placement: Almost never achieve identifiability with < 95% monitors; < 0.5 probability of achieving identifiability with 99% monitors

- MMP: Guarantee identifiability with 15-64% monitors

Problem 3: Path construction

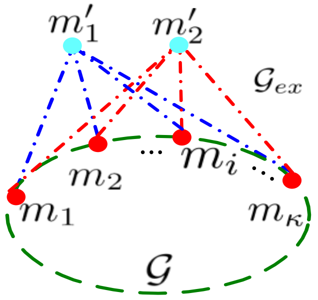
Objective

- Given an identifiable network G with n links, construct n linearly independent measurement paths that are cycle-free and start/end at monitors.
  - Cannot brute-force (exponentially many candidate paths)

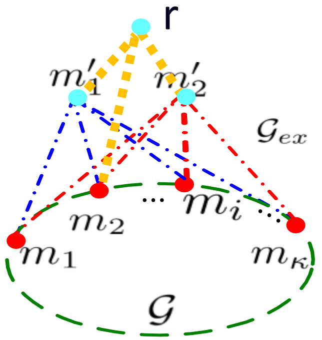
Efficient algorithm to find a basis of paths

Main idea

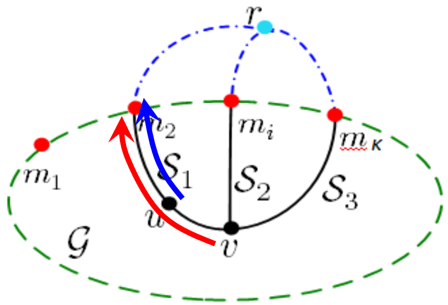
Given an identifiable $\mathcal{G}$, $\mathcal{G}^*_{\text{ex}}$ must be 3-vertex-connected → $\mathcal{G}^*_{\text{ex}}$ has 3 independent spanning trees rooted at r



Extended graph $\mathcal{G}_{\text{ex}}$



r-extended graph G^*_{ex}



3 vertex-independent paths to r

Measurement paths:

$$\mathcal{P}_1 = S_1 + S_2$$
$$\mathcal{P}_2 = S_2 + S_3$$
$$\mathcal{P}_3 = S_3 + S_1$$

Metrics of S_1, S_2, S_3

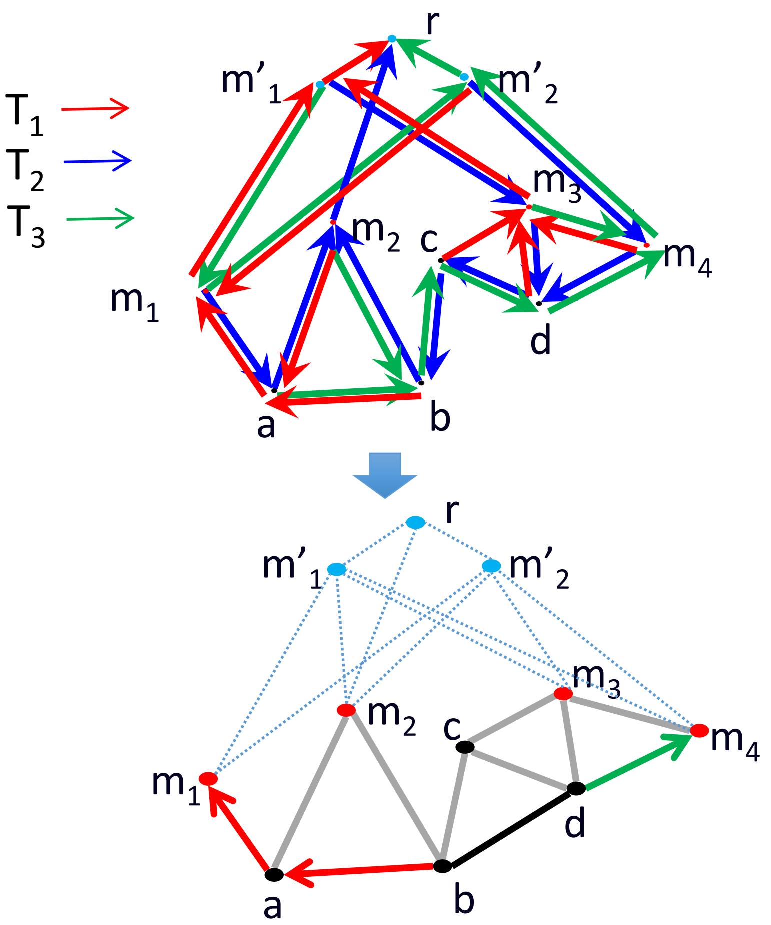
Link metrics:

$W_{uv} = W_{v \to m2} - W_{u \to m2}$

Spanning Tree-based Path Construction (STPC)

```
**Algorithm STPC**
Construct $\mathcal{G}^*_{ex}$ from $\mathcal{G}$
Find 3 independent spanning trees $T_1$, $T_2$, $T_3$ [Cheriyan88]
For each node v in $\mathcal{G}$ do
    If v is a monitor
        $\mathcal{P}_1 = S_1, \mathcal{P}_2 = S_2, \mathcal{P}_3 = S_3$
    else
        $\mathcal{P}_1 = S_1+S_2, \mathcal{P}_2 = S_2+S_3, \mathcal{P}_3 = S_3+S_1$
    end
end
For each link $l$ not in $T_1 \cup T_2 \cup T_3$
    Find a path traversing $l$ in graph $T_1 \cup T_2 \cup T_3 + l$
end
```

Time Complexity: O(|V|*|L|)



ISP	n	m	κ	r_{SUCC}	Υ	t_{STPC} (s)	t_{RWPC} (s)	t_{STLI} (ms)	t_{MILI} (ms)	h_{STPC}	h_{RWPC}
Abovenet	294	182	117	80.00%	99.61%	10.12	58.20	2.46	5.08	5.68	4.03

6-879x speedup

slightly longer paths

[Cheriyan88] J. Cheriyan and S. N. Maheshwari, “Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs,” Journal of Algorithms, vol. 9, pp. 507–537, 1988.

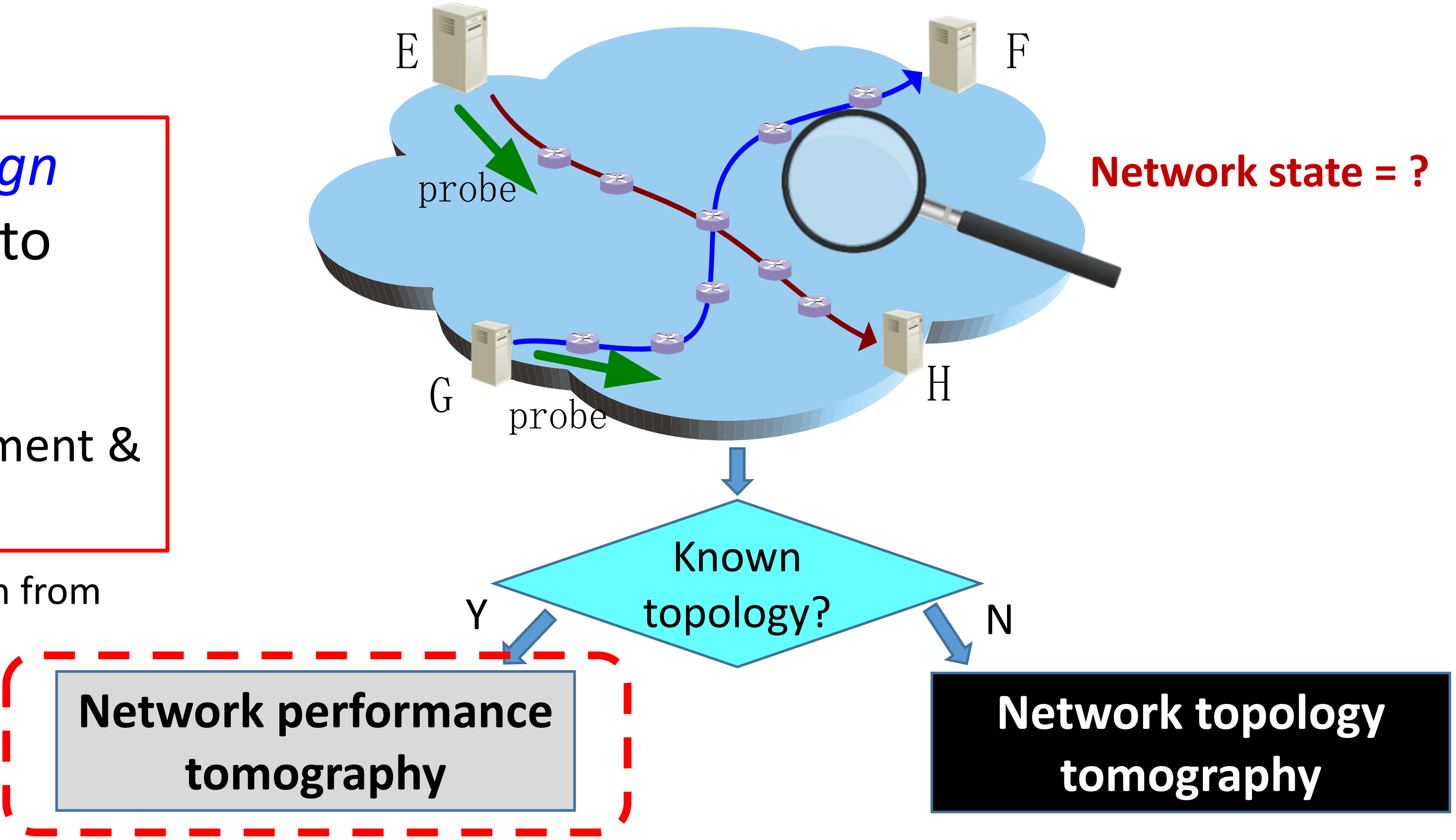
Summary: Network performance tomography

- Given topology, using *end-to-end measurements* to infer *link metrics*

Need to *carefully design measurement paths** to achieve *identifiability*

- Topological conditions
- Efficient monitor placement & path construction

*requires data-plane cooperation from the network



Known topology?

Network performance tomography

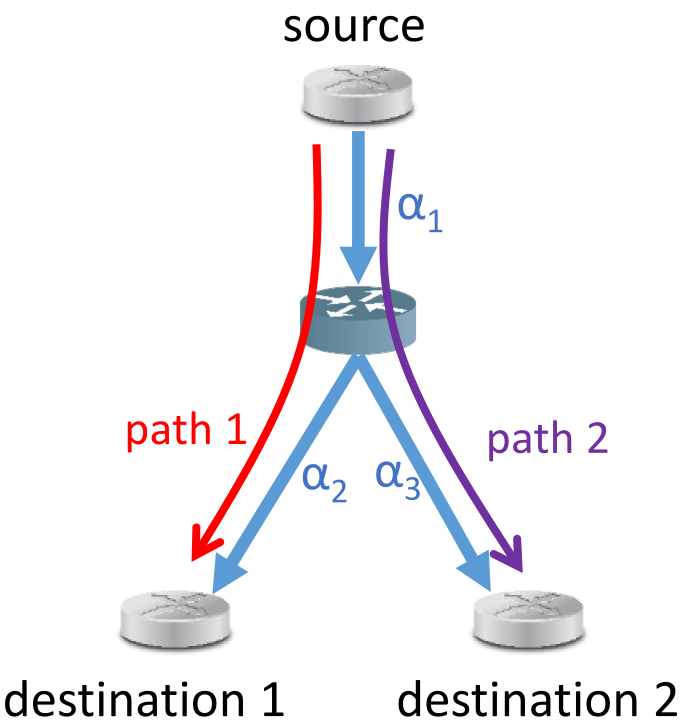
Network topology tomography

Network Topology Tomography

Limitation and Application

Toy example: Why is topology inference possible

- Multicast measurements reveal internal topology



α_i : link success probability

$-\log \alpha_i$: "link length"

$$-\log \alpha_1 - \log \alpha_2 = -\log \Pr\{X_{p_1} = 1\}$$
$$-\log \alpha_1 - \log \alpha_3 = -\log \Pr\{X_{p_2} = 1\}$$

"path length"

$$-\log \alpha_1 = -\log\left(\frac{\Pr\{X_{p_1} = 1\}\Pr\{X_{p_2} = 1\}}{\Pr\{X_{p_1} = X_{p_2} = 1\}}\right)$$

"shared path length"

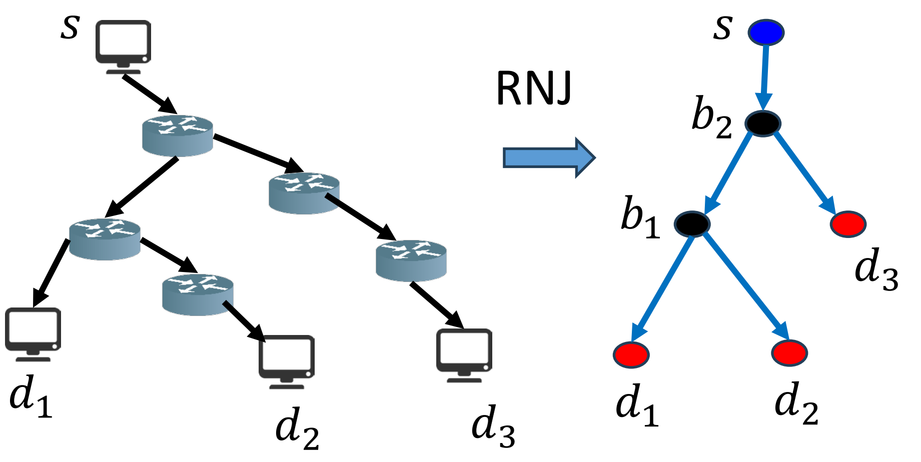
X_{p_i} : success indicator for path p_i

No shared link (or sharing a lossless link) ⟺ $-\log \alpha_1 = 0$

Sharing a lossy link ⟺ $-\log \alpha_1 > 0$

Classical case: Inference of multi-cast tree

- Infer the *logical* routing tree from multicasts sent by a single source



physical topology

logical topology

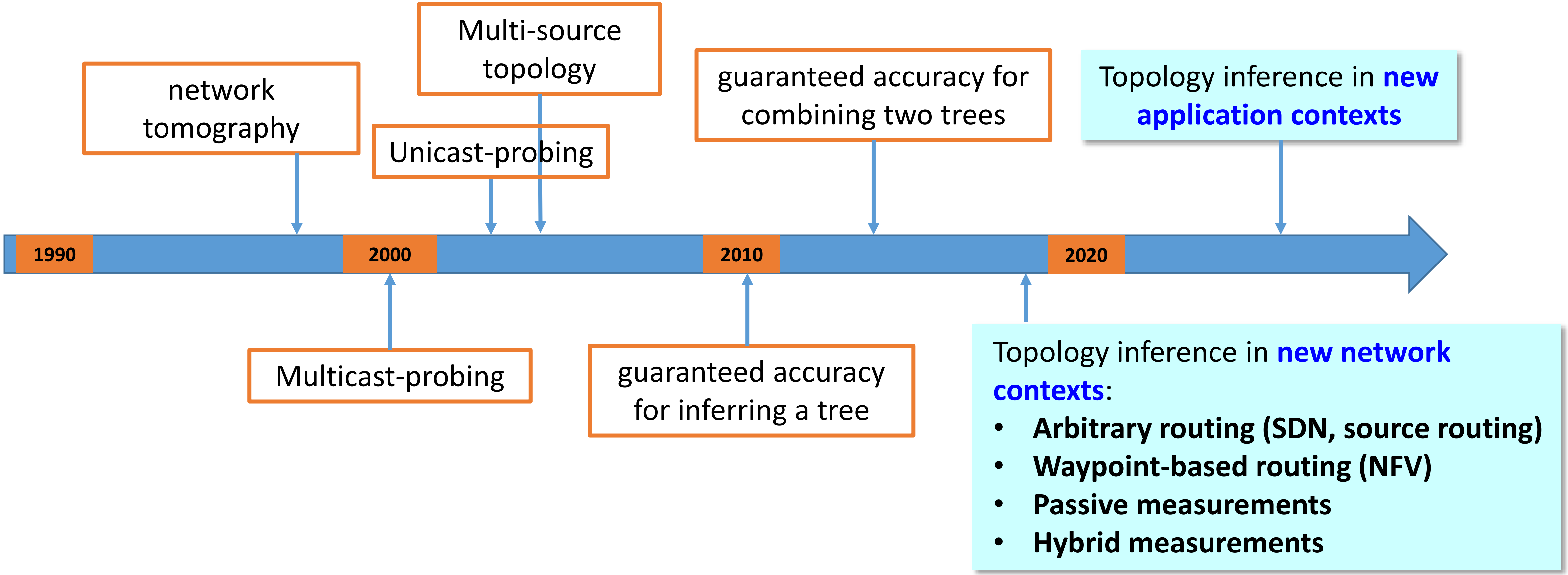
Rooted Neighbor Joining (RNJ): Given $s, D, (\rho_{ij})_{i,j \in D}$

```
While $|D| > 1$:
    find $(i, j)$ with the largest shared path length
    create a new node $b$ as the parent of $i, j$
    compute shared length between $b$ and every $k \in D$
    replace $i, j$ by $b$ in $D$
Connect the only node in $D$ to $s$
```

*additional steps to handle parent with > 2 children

Theorem: RNJ correctly reconstructs the logical routing tree if inferred shared path lengths are sufficiently accurate. (error < $\frac{1}{4}$ "minimum link length")

History: Where we are



network tomography
Multi-source topology
Unicast-probing
guaranteed accuracy for combining two trees
Topology inference in **new application contexts**
1990
2000
2010
2020
Multicast-probing
guaranteed accuracy for inferring a tree

Topology inference in **new network contexts**:

- **Arbitrary routing (SDN, source routing)**
- **Waypoint-based routing (NFV)**
- **Passive measurements**
- **Hybrid measurements**

State of the art on topology inference

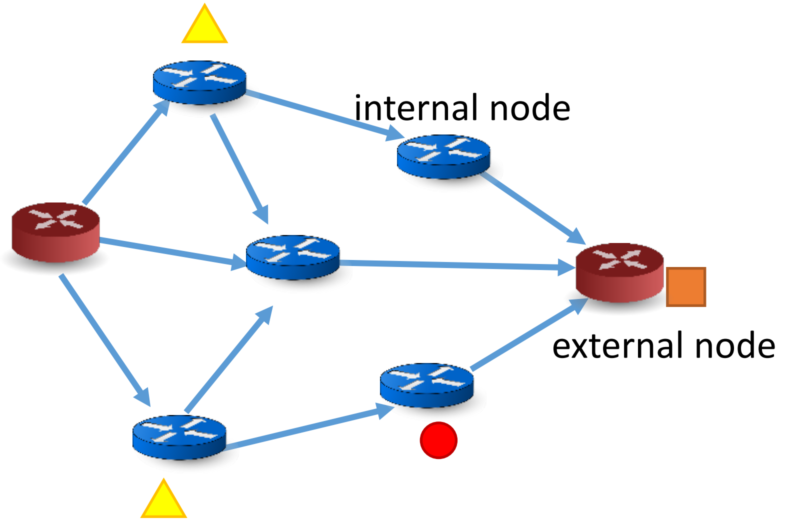
Restriction on measurement

	Tree-based	Waypoint-based	Arbitrary
Probe no path	Queue fingerprinting		
Probe some paths	Shared link detection		
Probe all paths	Most existing solutions, e.g., RNJ REA	1-1-N	**SAP**

Flexibility of routing

Scenario: Probe all paths, arbitrary routing

- **Motivation**: Inferring the structure and state of *SDN-NFV network*
  - general topology
  - waypoint traversal
  - known service chain
- **Observation:**
  - Measured: end-to-end performance measurements (e.g., losses)
  - Inferred: lengths of paths, shared paths, union of paths
    - "length" measured by additive metric
    - E.g., $\theta_e = -\log \alpha_e$ (α_e: success prob. of edge e)
  - Static: source, destination, service chain



NFV network

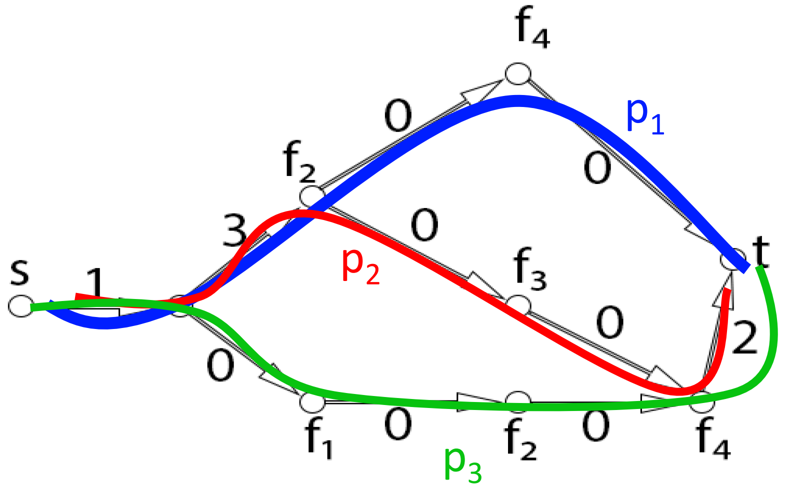
● network function 1 (e.g., IDS)

▲ network function 2 (e.g., firewall)

■ network function 3 (e.g., proxy)

Tree-based topology inference is insufficient

- Classic topology inference algorithms all assume tree-based routing
- But trees cannot always reconstruct the observations from a non-tree topology

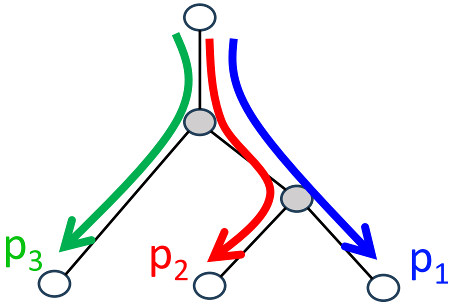


$\text{leng}(p_1)=4$
$\text{leng}(p_2)=6$
$\text{leng}(p_3)=3$

$\text{leng}(p_1 \cap p_2)=4$
$\text{leng}(p_1 \cap p_3)=1$
$\text{leng}(p_2 \cap p_3)=3$

No tree topology reconstructs all these lengths

→ **not even guarantee a feasible solution**



Identifiable information at finest granularity

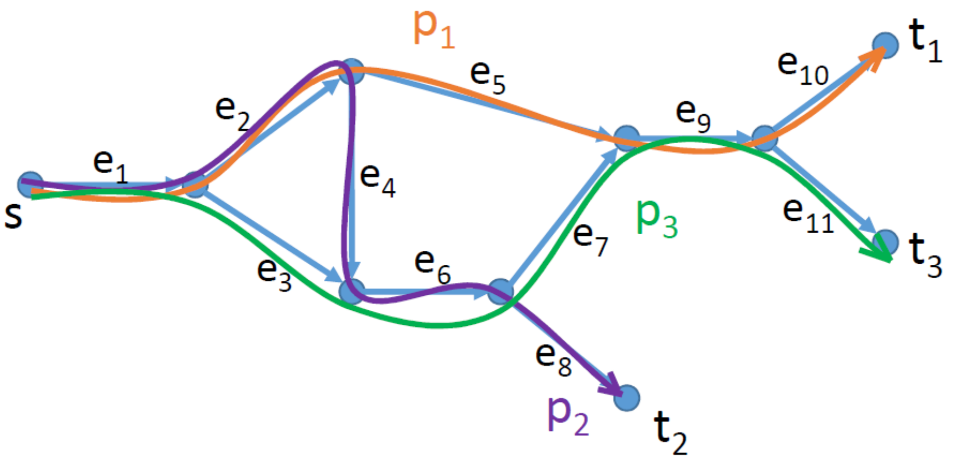
- **Weight Inference Problem**:
  - Partition edges into 2^n-1 *categories*
    - **Category $\boldsymbol{\Gamma_F}$**: set of edges *traversed by and only by* paths with indices in F
    - **Category weight $\boldsymbol{w_F}$**: sum metric of edges in category Γ_F
  - Estimate *cast weights* from e2e measurements
    - **Cast weight $\boldsymbol{\rho_F}$** for a multicast on paths in F:

$$\rho_F := -\log(\Pr\{X_F = 1\}) = -\log\left(\prod_{e\in\bigcup_{i\in F} p_i} \alpha_e\right) = \sum_{e\in\bigcup_{i\in F} p_i} \theta_e$$

  - Relationship between cast and category weights

Topology-agnostic

$$\boxed{\rho_F = \sum_{F'\subseteq E:F'\cap F\neq\emptyset} w_{F'}, \qquad \forall F \subseteq E}$$



$$\rho_1 = w_1 + w_{1,2} + w_{1,3} + w_{1,2,3}$$
$$\rho_2 = w_2 + w_{1,2} + w_{2,3} + w_{1,2,3}$$
$$\rho_3 = w_3 + w_{1,3} + w_{2,3} + w_{1,2,3}$$
$$\rho_{1,2} = w_1 + w_2 + w_{1,2} + w_{1,3} + w_{2,3} + w_{1,2,3}$$
$$\rho_{1,3} = w_1 + w_3 + w_{1,2} + w_{1,3} + w_{2,3} + w_{1,2,3}$$
$$\rho_{2,3} = w_2 + w_3 + w_{1,2} + w_{1,3} + w_{2,3} + w_{1,2,3}$$
$$\rho_{1,2,3} = w_1 + w_2 + w_3 + w_{1,2} + w_{1,3} + w_{2,3} + w_{1,2,3}$$

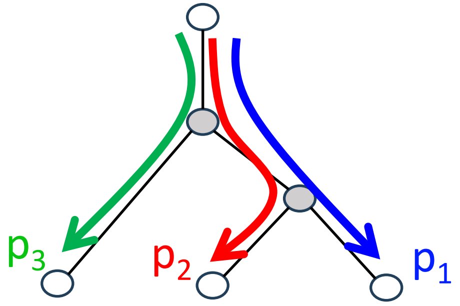
Theorem: Category weights $\overset{\textbf{uniquely determine}}{\Longleftrightarrow}$ Cast weights.

Category weights help, but are not enough

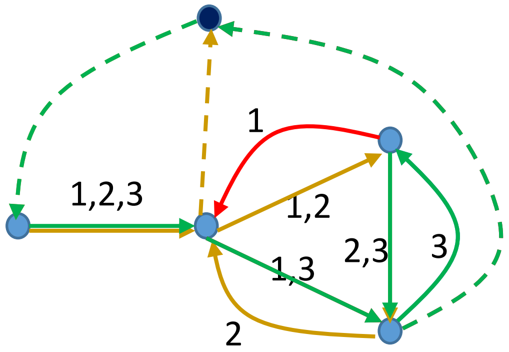
- Under mild assumption, category $\Gamma_F \neq \emptyset \leftrightarrow w_F \neq 0$
- For trees, knowing non-empty categories → knowing (logical) topology

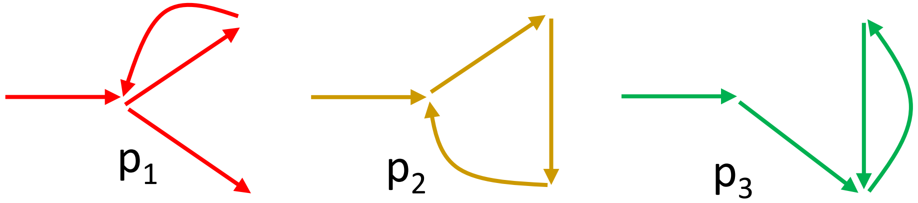
$$\Gamma_{1,2,3} \neq \emptyset$$
$$\Gamma_{1,2} \neq \emptyset$$
$$\Gamma_{1,}\Gamma_2, \Gamma_3 \neq \emptyset$$

⇨



- But not so for arbitrary topology
  - We can always embed the non-empty categories in a clique-like topology



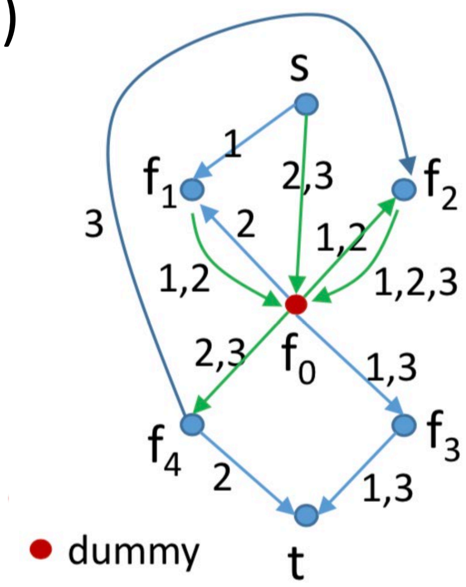


Idea: Combining categories with service chain

- **String Augmentation Problem (SAP):**
  - view each service chain as a string s_i, $f_{i,1}$, $f_{i,2}$,...,t_i
  - insert dummy letters f_0^1, f_0^2,... s.t. for every positive-weight category A, ∃a pair of letters appearing *only* in string i ($i \in \mathrm{A}$)

p'_1: s f_1 f_2 f_3 t
p'_2: s f_2 f_1 f_4 t
p'_3: s f_4 f_2 f_3 t
$\mathcal{A}_+$: {1},{2},{3},
{1,2},{1,3},{2,3},
{1,2,3}

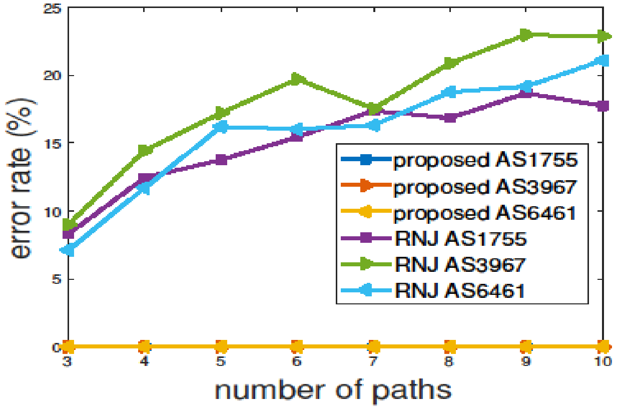
p_1: s f_1 f_0 f_2 f_0 f_3 t
p_2: s f_0 f_2 f_0 f_1 f_0 f_4 t
p_3: s f_0 f_4 f_2 f_0 f_3 t



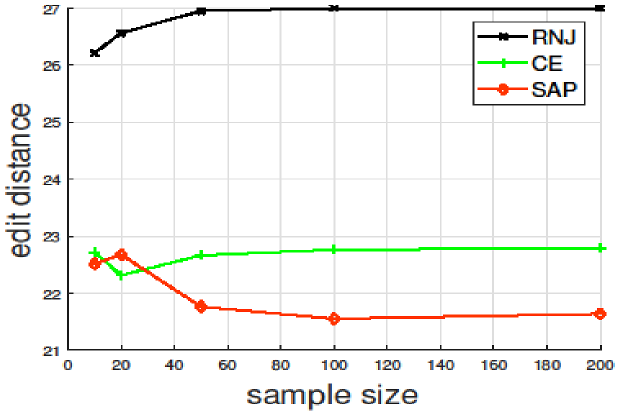
  - Minimize #nodes/#links (can be formulated as an ILP)

Evaluation: VNF topology inference

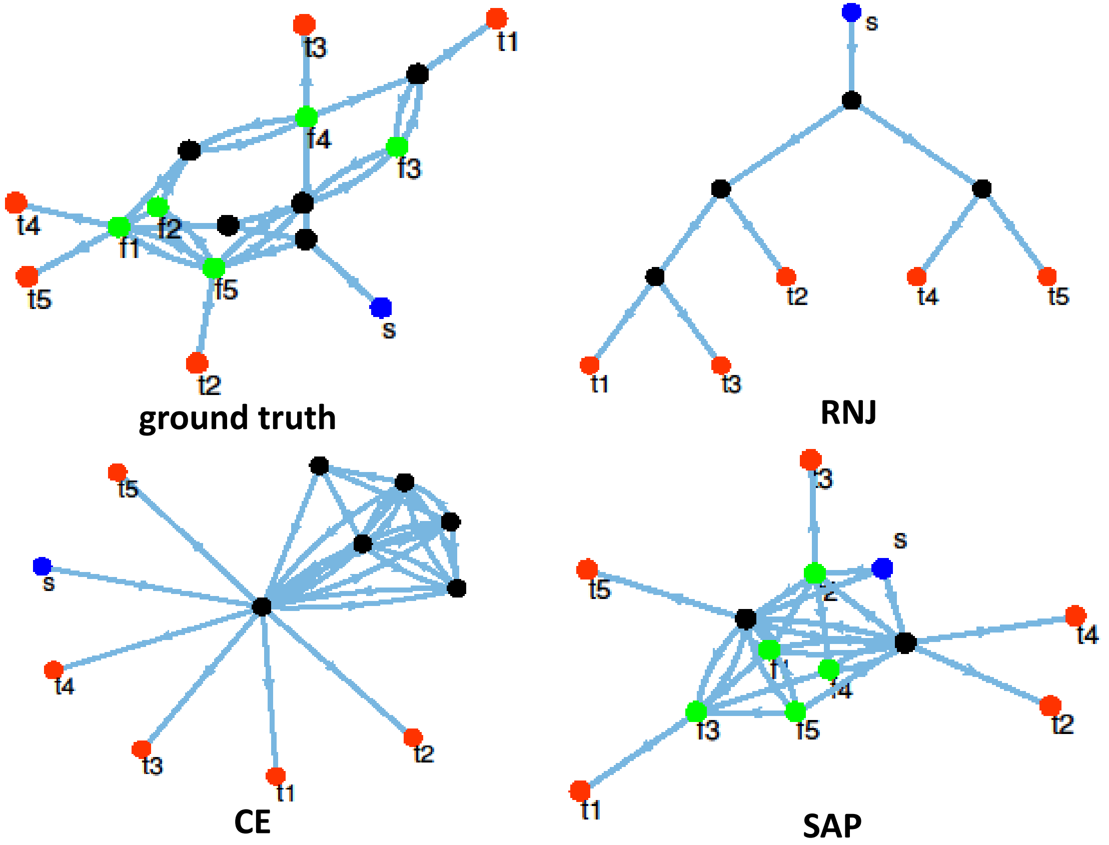
- Based on VNF overlays randomly generated on Rocketfuel AS topologies



(a) reconstruction error



(b) convergence



(c) inferred topologies

Tree-based inference → not feasible

Category-based inference → feasible but not accurate

Topology inference from the perspective of upper-layer application

Restriction on measurement

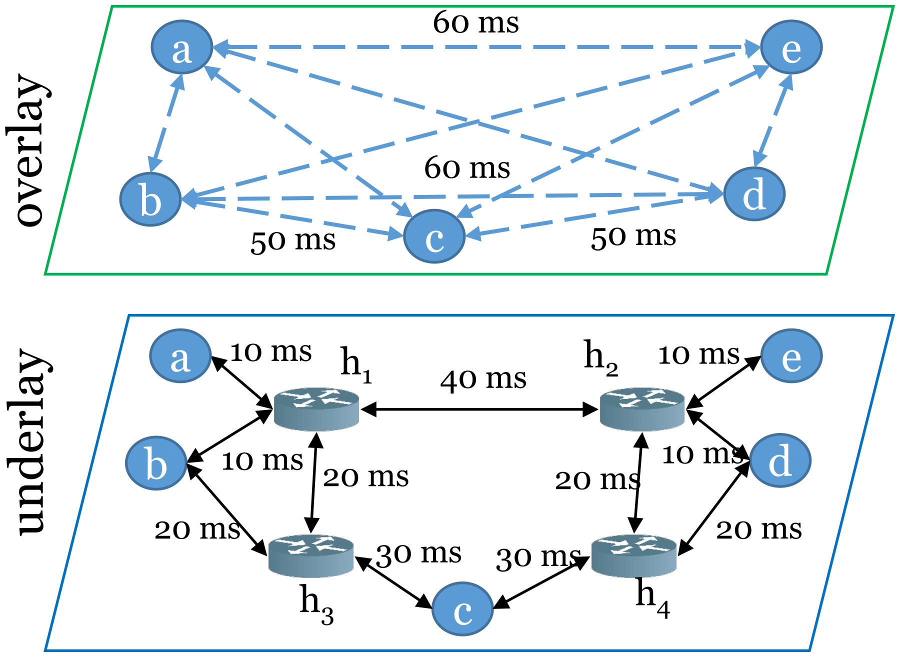
	Tree-based	Waypoint-based	Arbitrary
Probe no path	Queue fingerprinting		
Probe some paths	Shared link detection		
Probe all paths	Most existing solutions, e.g., RNJ REA	1-1-N	**SAP**

SAP ⇨ **Overlay routing**

Flexibility of routing

Motivation: Overlay-based data transfer

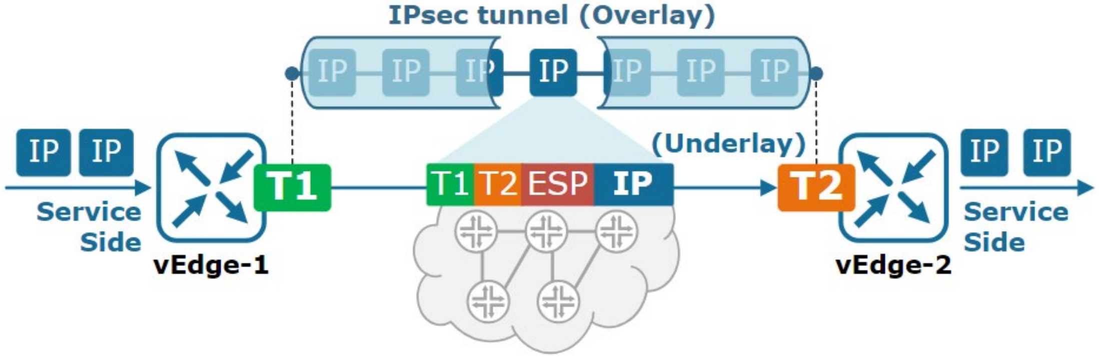
- **Overlay network:** A logical network running on top of a communication underlay
  - Enhance best-effort IP-based underlay
    - Caching, traffic engineering, service-chaining, multicast, fast failover, network slicing, …
  - Focus: **overlay-based routing**



- Example: SD-WAN
  - Software-Defined Wide-Area Networks

Managed SD-WAN Solutions

AT&T SD-WAN solutions can improve your network's agility and provide centralized control and improve total cost of ownership.

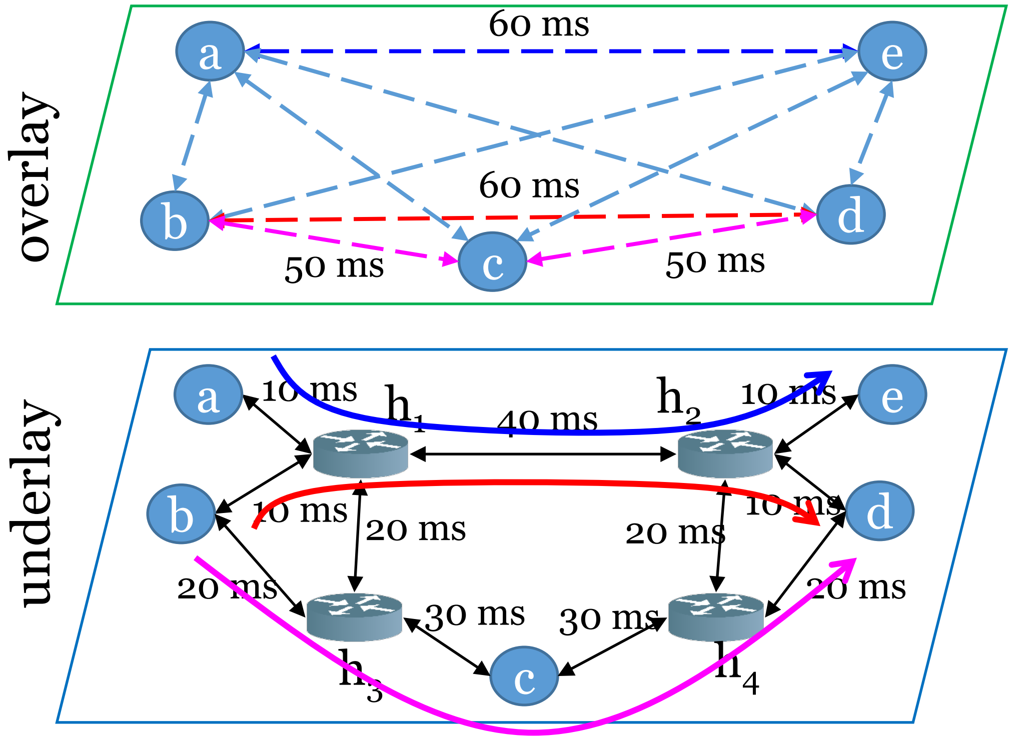


Cisco SD-WAN overlay fabric

Challenges for routing in overlay network

- Overlay routing differs from classical routing problems
  - Seemingly independent *tunnels (overlay links)* share underlay links
    - Congestion-prone
  - Uncooperative underlay
    - No direct underlay topology information

Q: Do we need the full topology for overlay routing?
A: No!



- Flow: a->e and b->d
- Direct tunnel: both traverse $h_1 \rightarrow h_2$
- Congestion-free overlay routing:
  - a->e
  - b->c->d

Overlay Routing Problem

$$\min_{\boldsymbol{x}} \sum_{all\_tunnels} tunnel\_cost \sum_{all\_demands} demand \cdot x_{tunnel}^{demand}$$

$s.t. \quad x_{tunnel}^{demand} \in \{0,1\}$

flow conservation constraints

Depend on underlay routing & link capacities

$$\boxed{\textstyle\sum_{tunnels\_traverse\_link} \sum_{demands} f_{tunnel}^{demand} \leq link\_capacity, \forall links}$$

Q: What is the **minimum information** for **imposing equivalent capacity constraints**?

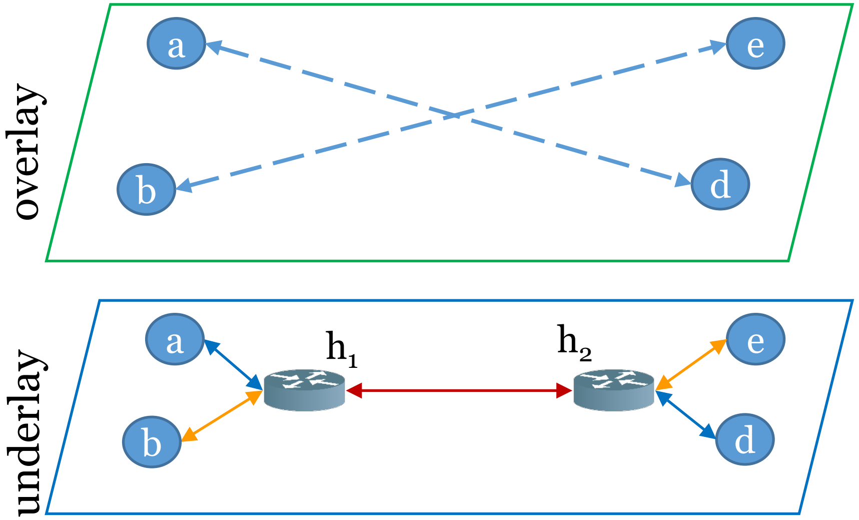
Recall: Categorization of underlay links

- (Underlay) link category
  - $\Gamma_F(E)$: A **category of links traversed by *F* out of *E*** ($F \subseteq E$) is the set of underlay links traversed **by and only by** the tunnels in F out of all the tunnels in E
    - i.e., $\Gamma_F(E) \coloneqq \boxed{\left(\bigcap_{(i,j)\in F} \underline{p}_{i,j}\right)} / \boxed{\left(\bigcup_{(i,j)\in E\setminus \mathrm{F}} \underline{p}_{i,j}\right)}$

      Links shared by F

      All links traversed by $E\setminus F$
  - Category weight: $w_F(E) \coloneqq \sum_{\underline{e}\in\Gamma_F(E)} \theta_{\underline{e}}$

Claim: Knowledge of link categories suffices for **congestion-free overlay routing**

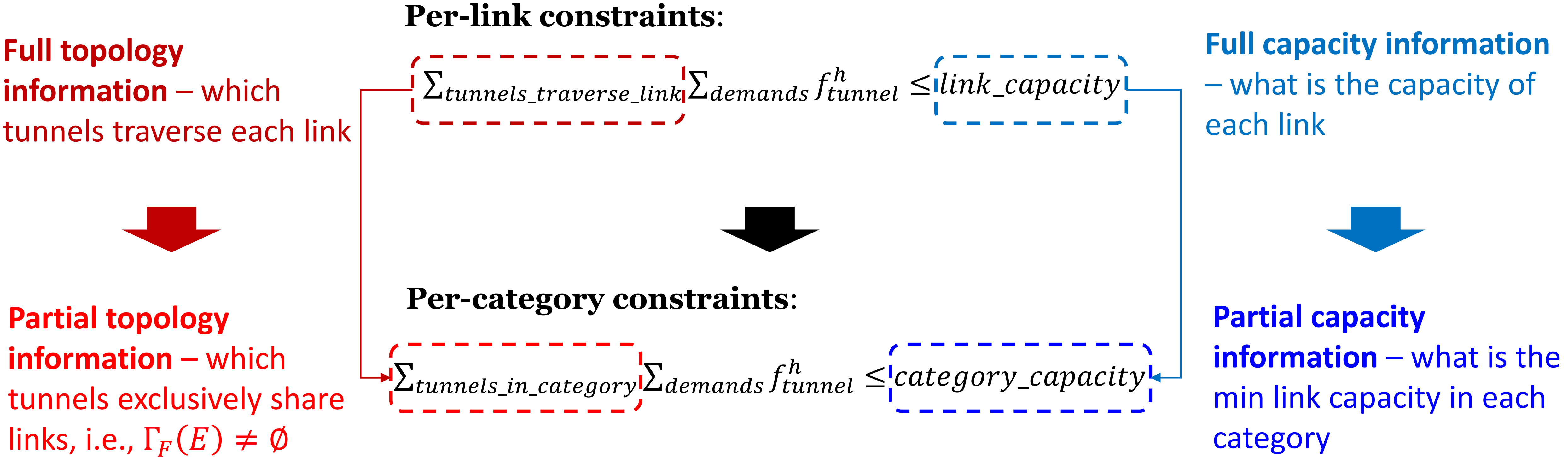


Example: $E = \{(a,d),(b,e)\}$

- $F_1 = \{(a,d),(b,e)\}$
  - $\Gamma_{F_1}(E) = \{(h_1,h_2)\}$
- $F_2 = \{(a,d)\}$
  - $\Gamma_{F_2}(E) = \{(a,h_1),(h_2,d)\}$
- $F_3 = \{(b,e)\}$
  - $\Gamma_{F_3}(E) = \{(b,h_1),(h_2,e)\}$

Category-based capacity constraints

Links in the same category receive the same traffic load from the overlay



Per-link constraints:

$$\sum_{tunnels\_traverse\_link}\sum_{demands} f_{tunnel}^{h} \le link\_capacity$$

Full topology information – which tunnels traverse each link

Full capacity information – what is the capacity of each link

Per-category constraints:

$$\sum_{tunnels\_in\_category}\sum_{demands} f_{tunnel}^{h} \le category\_capacity$$

Partial topology information – which tunnels exclusively share links, i.e., $\Gamma_F(E) \neq \emptyset$

Partial capacity information – what is the min link capacity in each category

Computational challenge in inferring category weights

Measurements in overlay → ρ_F

$$\rho_F := \sum_{\underline{e} \in \bigcup_{(i,j)\in F} \underline{p}_{i,j}} \theta_{\underline{e}}$$

Candidate category weight w_F

$$w_F(E) := \sum_{\underline{e} \in \Gamma_F(E)} \theta_{\underline{e}}$$

$$\boxed{\rho_F = \sum_{F' \subseteq E : F' \cap F \neq \emptyset} \mathrm{w}_{F'}(E)\,, \forall F \subseteq E}$$

- **Full rank** linear system
- Under mild assumption, $w_F(E) > 0 \Longrightarrow \Gamma_F(E) \neq \emptyset$

Q: Is problem solved?
A: Unfortunately, no.

- **Exponential complexity:** #variables = $2^{|E|} = 2^{O(|V|^2)}$

Example: $|V| = 10$, number of candidate categories: 2^{90}

Meanwhile, **most candidate categories are empty**

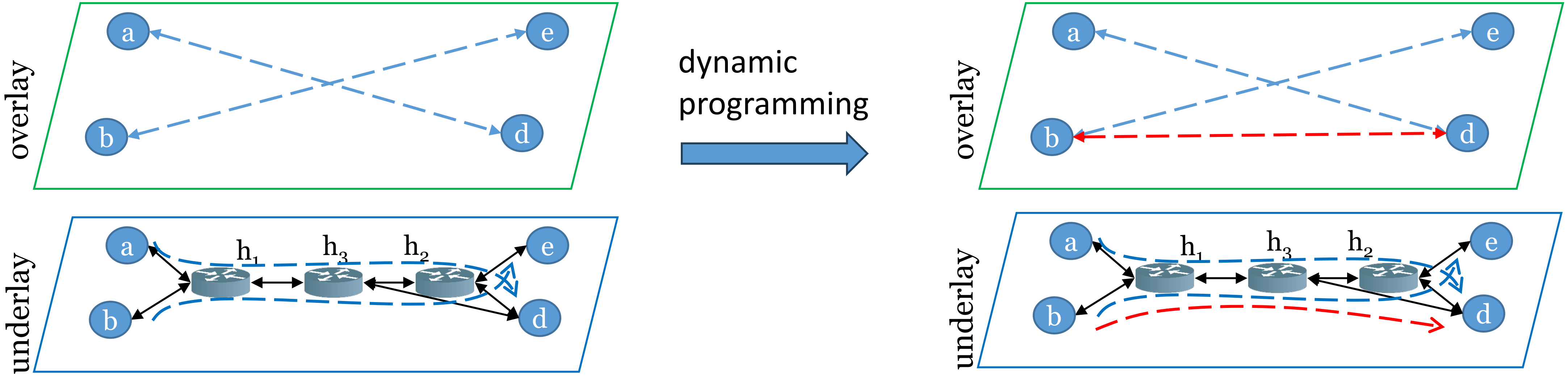
- #non-empty categories ≤ #underlay links

Idea: Dynamic programming

As we consider one more tunnel, empty categories remain empty, but each nonempty category may be decomposed into two new nonempty categories

$E_{t-1} = \{(a,d),(b,e)\}$

$E_t = \{(a,d),(b,e),(b,d)\}$



$\{w_F(E_{t-1})\}$

$\{(a,d),(b,e)\}$ → $\{(a,d),(b,e),(b,d)\},\{(a,d),(b,e)\}$

$\{(a,d)\}$ → $\{(a,d),(b,d)\},\{(a,d)\}$

$\{(b,e)\}$ → $\{(b,e),(b,d)\},\{(b,e)\}$

$\{(b,d)\}$

$\{w_F(E_t)\}$ candidates

$|supp(\mathbf{w}(E_{t-1}))| \le |\underline{E}|$

(#non-empty categories ≤ #underlay links)

#variables $= 2|supp(\mathbf{w}(E_{t-1}))| + 1$

Algorithm for Category Weight Inference

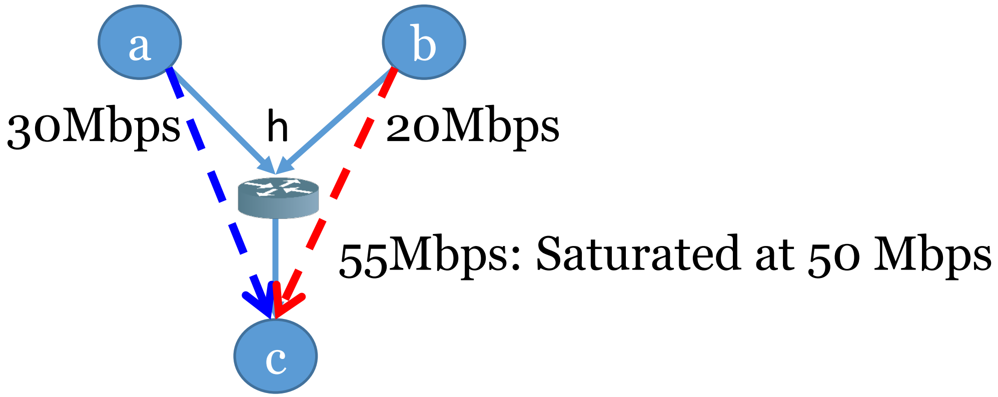
- Dynamic programming: In each iteration t,
  - $E_t \leftarrow E_{t-1} \cup \{e\}$
  - $w_{\{e\}}(E_t) \leftarrow \rho_{E_t} - \rho_{E_{t-1}}$
  - For $F \in supp\big(\boldsymbol{w}(E_{t-1})\big)$ in an increasing order of $|F|$:
    - $w_{F\cup\{e\}}(E_t) \leftarrow \rho_{(E_{t-1}\setminus F)\cup\{e\}} - \rho_{E_{t-1}\setminus F} - w_{\{e\}}(E_t) - \sum_{F'\subset F: F\in supp(\boldsymbol{w}(E_{t-1}))} w_{F'\cup\{e\}}(E_t)$
    - $w_F(E_t) \leftarrow w_F(E_t) - w_{F\cup\{e\}}(E_t)$
  - #variables = $2\left|supp\big(\boldsymbol{w}(E_{t-1})\big)\right| + 1 = O\big(|\underline{E}|\big)$

- In each iteration, solve a linear system whose size is **linear in the underlay network size**.
- In total $|E|$ iterations, **linear in the overlay network size**

→ **Polynomial-time algorithm** for category weight inference

Category Capacity Estimation

- The minimum capacity of the links in a category may not be measurable



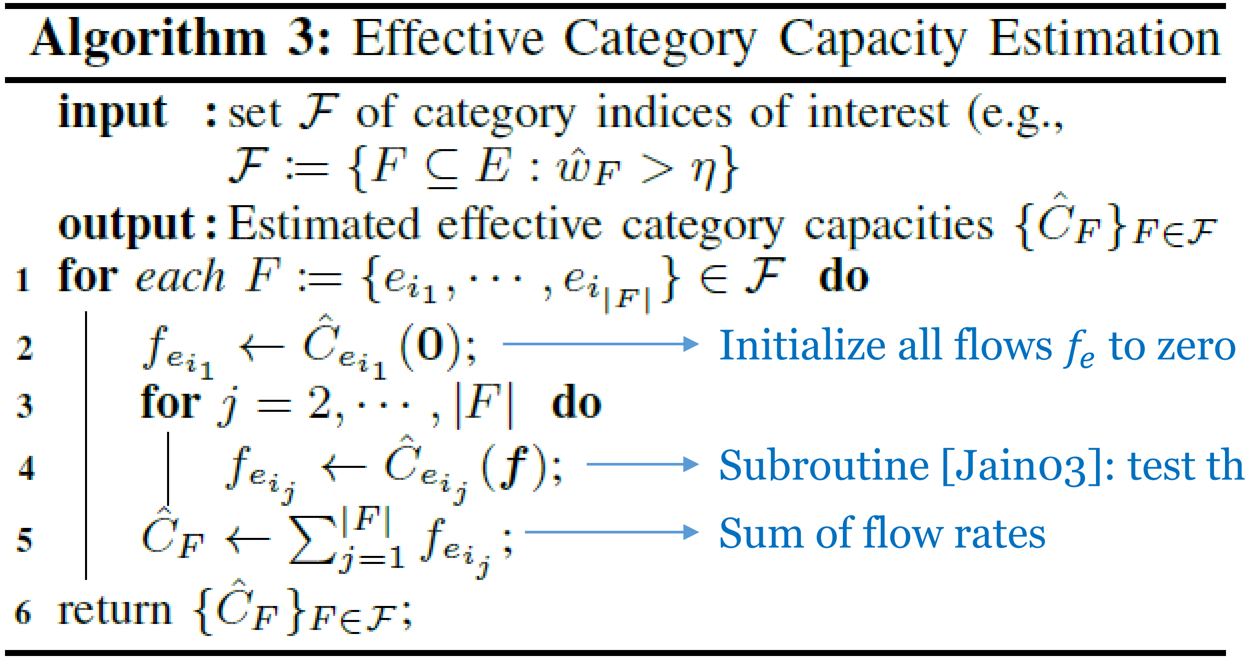
- **Effective category capacity:** Maximum total flow through the tunnels associated with the category
  - $\tilde{C}_F \coloneqq \max\limits_{(f_e)_{e\in E}} \sum_{e\in F} f_e$ (f_e: flow assigned to tunnel e)

    s.t. $\sum_{e'\in F'} f_{e'} \le \boxed{C_{F'}}, \forall F' \subseteq E, \Gamma_{F'} \neq \emptyset$

    $f_e \ge 0, \forall e \in E$ unknown

Algorithm for Effective Category Capacity Estimation

- Algorithm:



```
Algorithm 3: Effective Category Capacity Estimation
input  : set $\mathcal{F}$ of category indices of interest (e.g.,
         $\mathcal{F} := \{F \subseteq E : \hat{w}_F > \eta\}$
output : Estimated effective category capacities $\{\hat{C}_F\}_{F \in \mathcal{F}}$
1 for each $F := \{e_{i_1}, \cdots, e_{i_{|F|}}\} \in \mathcal{F}$ do
2     $f_{e_{i_1}} \leftarrow \hat{C}_{e_{i_1}}(\mathbf{0})$;
3     for $j = 2, \cdots, |F|$ do
4         $f_{e_{i_j}} \leftarrow \hat{C}_{e_{i_j}}(\boldsymbol{f})$;
5     $\hat{C}_F \leftarrow \sum_{j=1}^{|F|} f_{e_{i_j}}$;
6 return $\{\hat{C}_F\}_{F \in \mathcal{F}}$;
```

Initialize all flows f_e to zero

Subroutine [Jain03]: test the residual capacity of a tunnel given flow assignment

Sum of flow rates

- Performance guarantee:
  - If Line 4 is accurate, then Algorithm 3 achieves **$^1/_{q_F}$ approximation**
    - $q_F := \max_{e \in F} \left|\{F' \subseteq E : e \in F', \Gamma_{F'} \neq \emptyset, |F' \cap \mathrm{F}| > 1\}\right|$
    - i.e., maximum number of categories a tunnel in F traverses that are shared by at least another tunnel in F

[Jain03] Jain M, Dovrolis C. “End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput,” IEEE/ACM TNET, 2003.

Resulting Overlay Routing Problem

$$\min_{\boldsymbol{x}} \sum_{all\_tunnels} tunnel\_cost \sum_{all\_demands} demand \cdot x_{tunnel}^{demand}$$

$$s.t. \quad x_{tunnel}^{demand} \in \{0,1\}$$

flow conservation constraints

$$\sum\nolimits_{\boxed{tunnels\_in\_category}} \sum\nolimits_{demands} f_{tunnel}^{h} \leq \boxed{category\_capacity}$$

Partial topology information – which categories are nonempty

Partial capacity information – what is the effective category capacity

An NP-hard ILP (integer linear program)

- Can now be tackled as usual (e.g., branch-and-bound)

Performance evaluation in NS3

- Topologies from Internet Topology Zoo

	AttMpls	AboveNet	GTS-CE	BellCanada
$\lvert\underline{V}\rvert$	25	23	149	48
$\lvert\underline{E}\rvert$	114	62	386	130
$C_{\underline{e}}$ (Gbps)	1	1	1	1
Link delays (us)	[206,4973]	[100, 13800]	[5,1081]	[78, 6160]

- Background traffic
  - ON-OFF process for each link independently
    - Duration follows Pareto distribution
    - Utilization: [10%,40%]
- Probing
  - Number of overlay nodes: 10
  - 50-byte packets for probing; 1000-byte packets for routings
  - Measurements: end-to-end delays
- Routing cost: link (propagation) delays

Performance of overlay-based inference

Nonempty Category Detection

	AttMpls	AboveNet	GTS-CE	BellCanada
#empty cat.	$2^{90}-69$	$2^{90}-52$	$2^{90}-59$	$2^{90}-51$
#nonempty cat.	69	52	59	51
#false alarms	603	542	2159	1695
#misses	20	27	40	30

- **Low false alarm rate** although the absolute number is not small
- **High miss rate:** Inaccurate estimation of ρ_F if (1) $|F|$ is large or (2) tunnels in F have different sources

Effective Category Capacity Estimation

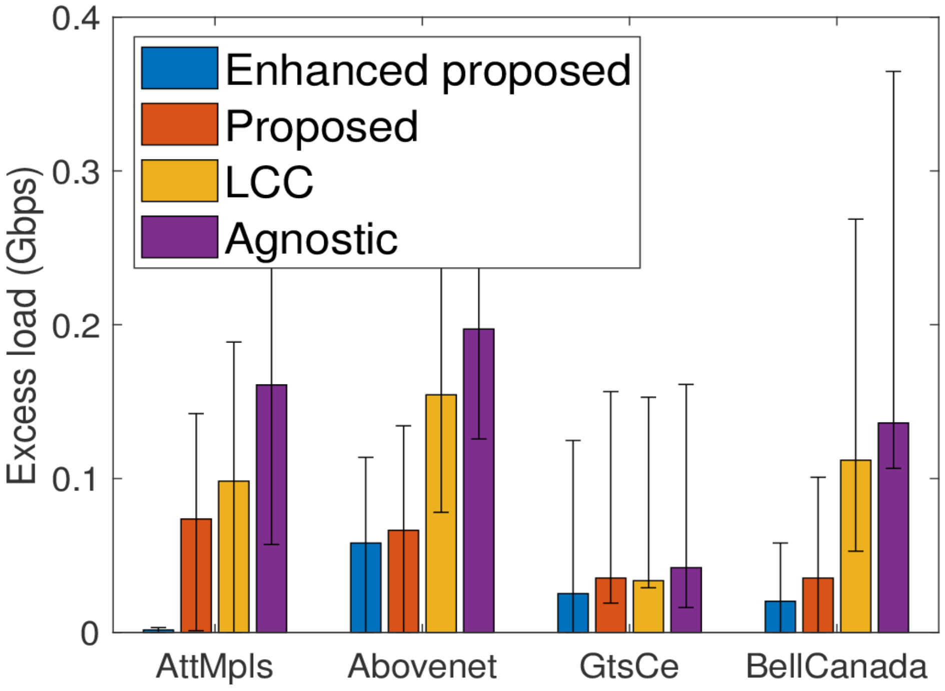
	AttMpls	AboveNet	GTS-CE	BellCanada
ideal subroutine	0.10%	0.13%	0.13%	0.4%
Pathload [Jain03]	1.07%	1.18%	1.15%	1.49%

- **Highly accurate capacity estimation:** ***False alarms will not hurt*** in most case, but ***misses may lead to congestions***.

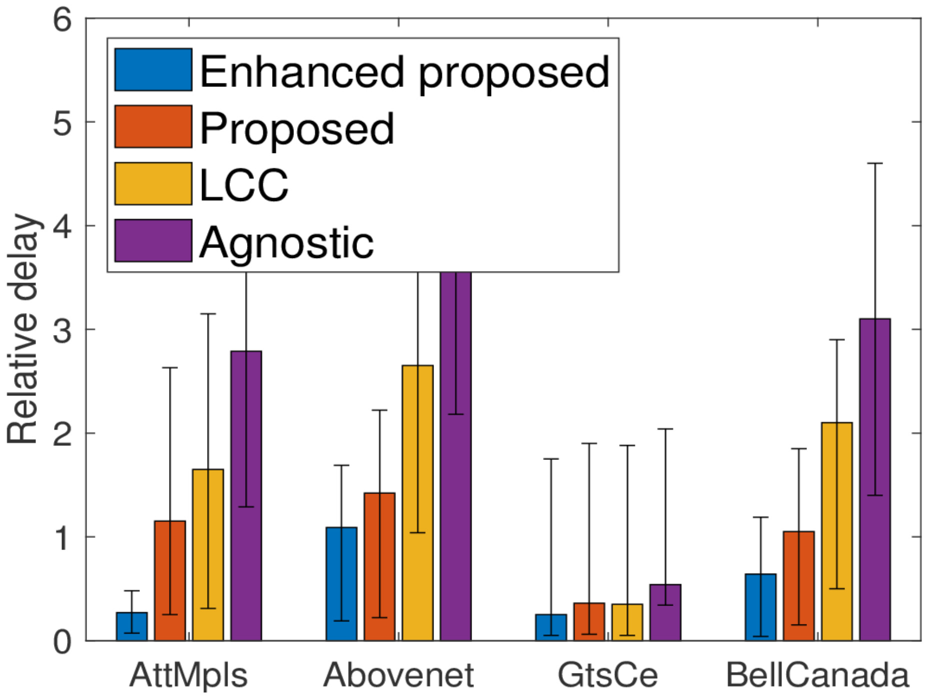
Performance of overlay routing

- Benchmarks
  - "**Agnostic**": an underlay-agnostic routing
  - "**LCC**": the state-of-the-art solution from [Zhu08]
  - "**Proposed**"
  - "**Enhanced proposed**": "Proposed" + "LCC"

Network topology tomography significantly improves overlay routing, despite notable inference errors



max excess load

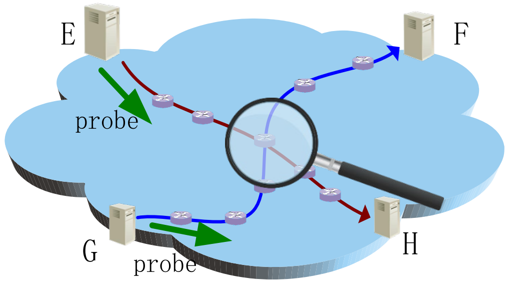


avg relative delay

[Zhu08] Y. Zhu and B. Li, "Overlay networks with linear capacity constraints," IEEE TPDS, 2008

Concluding remark

- **Network tomography** is:
  - a ***tool*** for ***applications*** to monitor the ***internal network state***
  - while requiring ***little/no cooperation*** from the network



- **Performance tomography**: Identifiability is possible, but
  - Need ***careful design of measurement paths*** to achieve it
- **Topology tomography**: No unique solution (unidentifiable), but
  - ***Some internal information is identifiable*** (e.g., category weights), and
  - can be ***quite useful*** (e.g., infer logical routing trees, or capacity region for overlay routing)

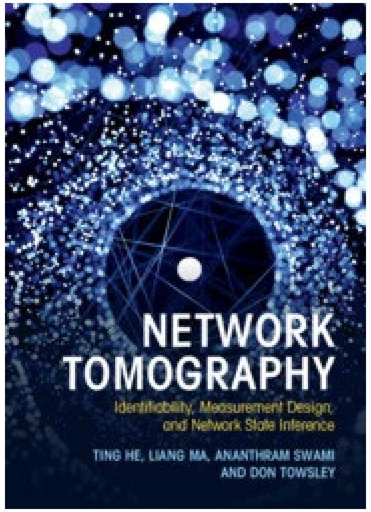
Outlook: Tomography for *new types of networks* & *new applications*

- Can tomography be applied to non-communication networks?
- What other overlay functions can benefit from tomography?

Network Tomography:

Inverse Methods for Network State Monitoring from End-to-End Measurements

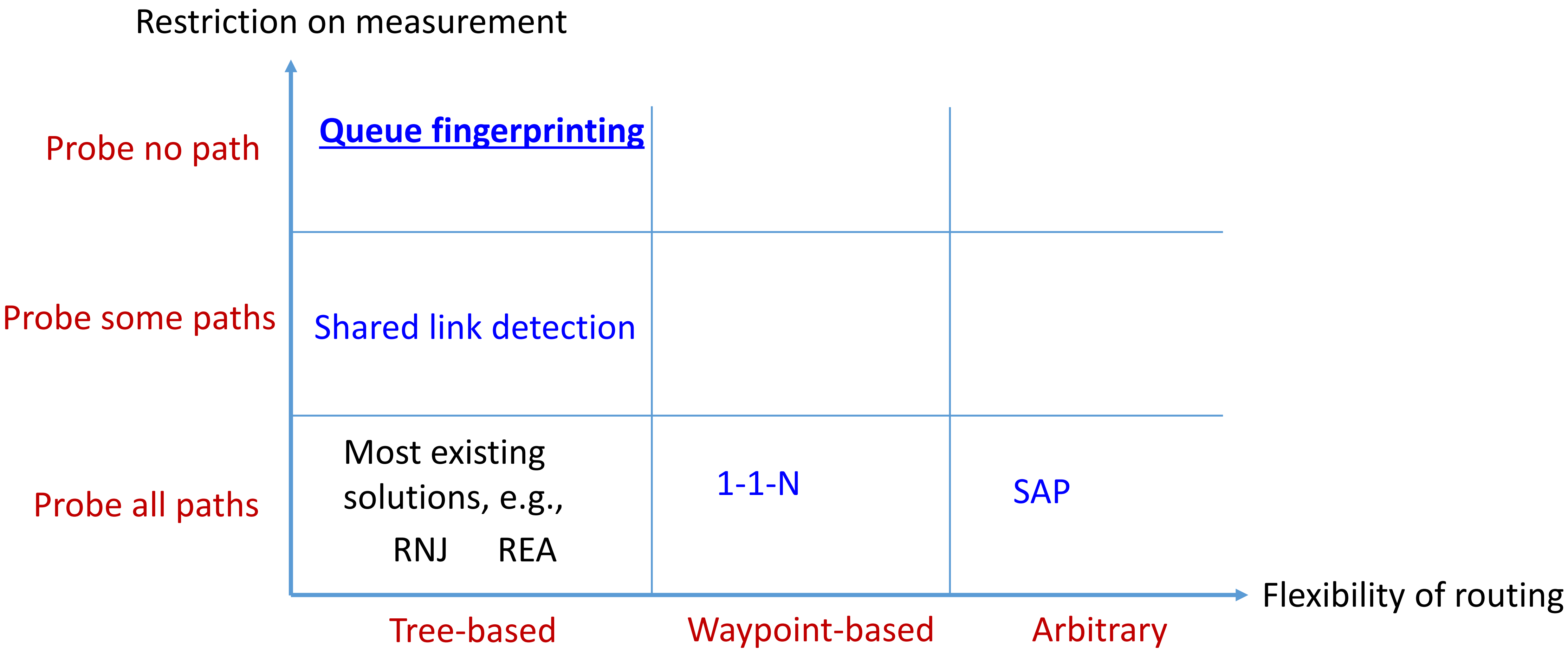
Ting He
tinghe@psu.edu


NETWORK TOMOGRAPHY
Identifiability, Measurement Design, and Network State Inference
TING HE, LIANG MA, ANANTHRAM SWAMI AND DON TOWSLEY

THANK YOU

Backup slides

Outline



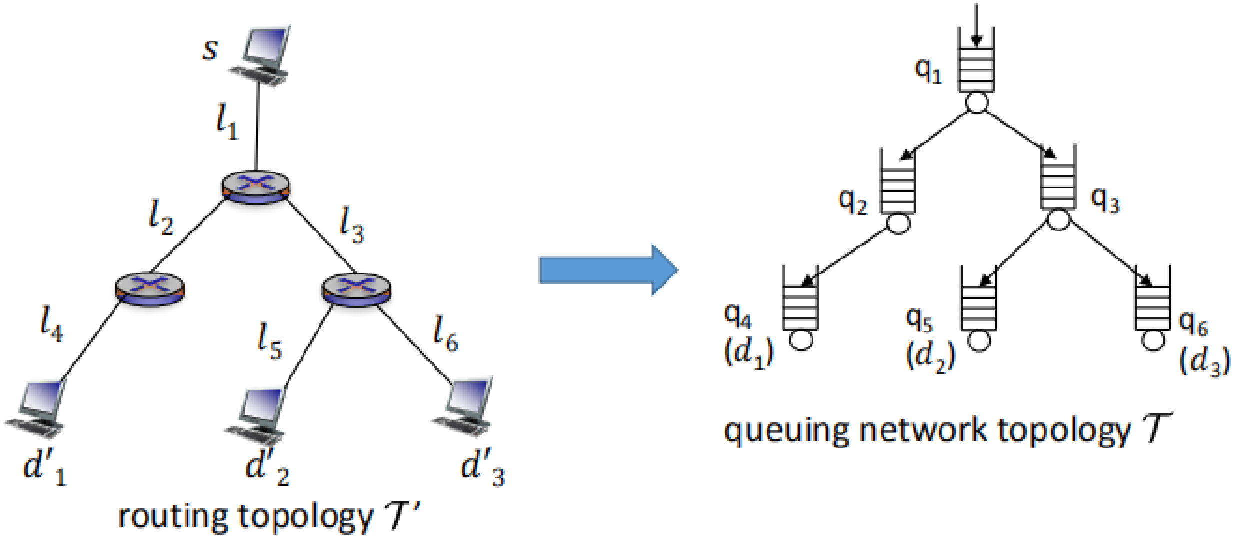
Restriction on measurement

Flexibility of routing

	Tree-based	Waypoint-based	Arbitrary
Probe no path	**Queue fingerprinting**		
Probe some paths	Shared link detection		
Probe all paths	Most existing solutions, e.g., RNJ REA	1-1-N	SAP

Scenario: Passive monitoring only

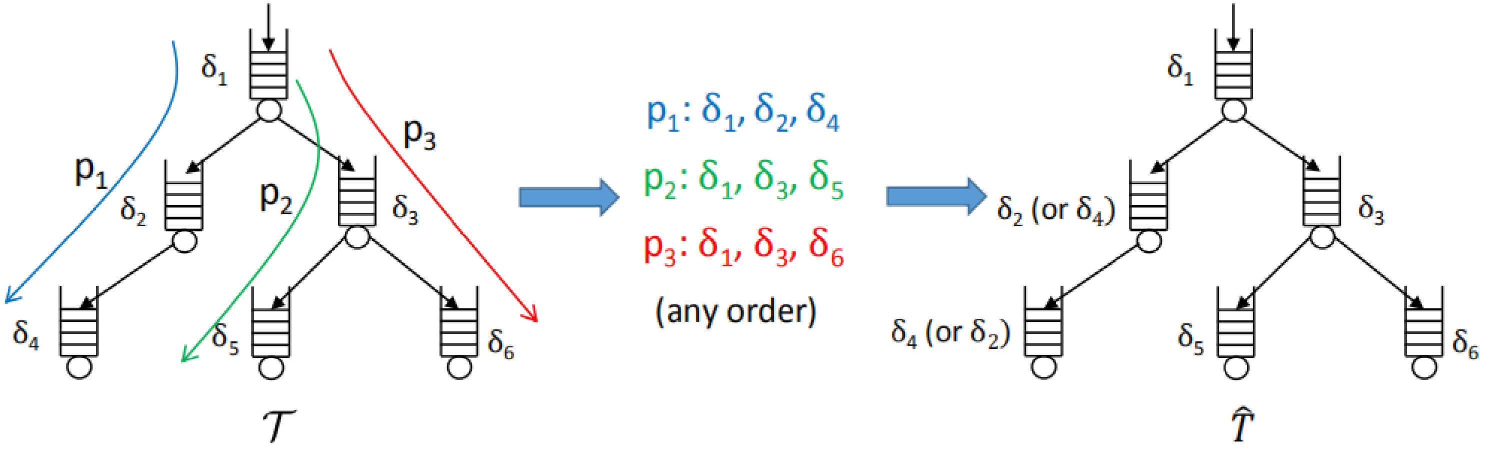
- A network of independent M/M/1 queues



- **Goal**: Address two key limitations of existing solutions
  - Active probing → **passive monitoring**
  - Logical topology → **physical topology**

Why it is feasible

- Queue parameter: $\delta_i = \mu_i - \lambda_i$ (residual capacity)
- Sojourn time: exponential r.v. with PDF $\delta_i e^{-\delta_i t_i}$
- End-to-end delay: hypoexponential r.v. with parameters $\boldsymbol{\delta} := (\delta_i)_{i=1}^{K}$
- Idea: **Queue fingerprinting**



p_1: δ_1, δ_2, δ_4

p_2: δ_1, δ_3, δ_5

p_3: δ_1, δ_3, δ_6

(any order)

(a)

(b)
Estimated parameters

(c)
Inferred topology

Parameter estimation for tandem of M/M/1 queues: Estimator

- Idea 1: **MLE**

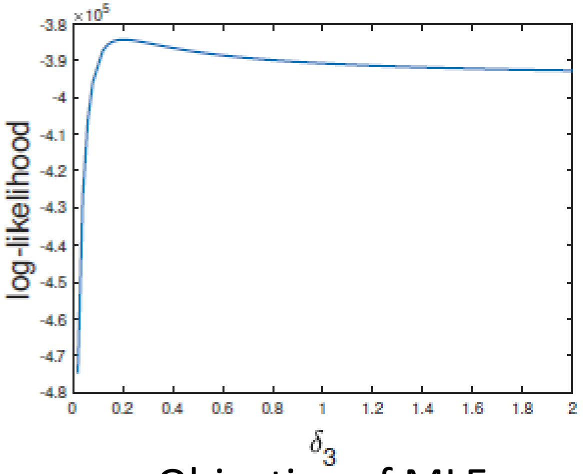
$$\boxed{\hat{\boldsymbol{\delta}} = argmax_{\boldsymbol{\delta}} \sum_{h=1}^{n} \log g(x_h; \boldsymbol{\delta})}$$

  - PDF: $g(x;\, \boldsymbol{\delta}) = \sum_{i=1}^{K} \delta_i e^{-x\delta_i} \left(\prod_{j=1, j\neq i}^{K} \frac{\delta_j}{\delta_j - \delta_i} \right)$

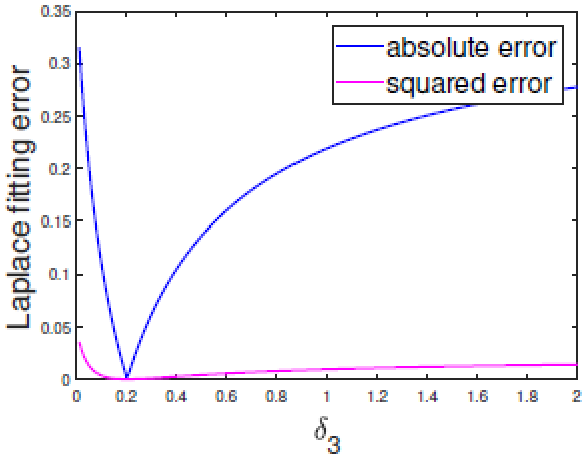
- Idea 2: **Fitting Laplace transform**
  - Laplace transform: $L(s;\, \boldsymbol{\delta}) := \prod_{i=1}^{K} \frac{\delta_i}{\delta_i + s}, \quad s > -\min_{i=1,\dots,K} \delta_i.$
  - Empirical Laplace transform: $\hat{L}(s;\, \boldsymbol{x}) := \frac{1}{n} \sum_{h=1}^{n} e^{-s x_h}$

→

$$\boxed{\begin{array}{c} \min \quad \sum_{s \in S} |L(s;\, \boldsymbol{\delta}) - \hat{L}(s;\, \boldsymbol{x})| \\ \text{s.t. } 0 < \delta_1 \leq \cdots \leq \delta_K, \end{array}}$$



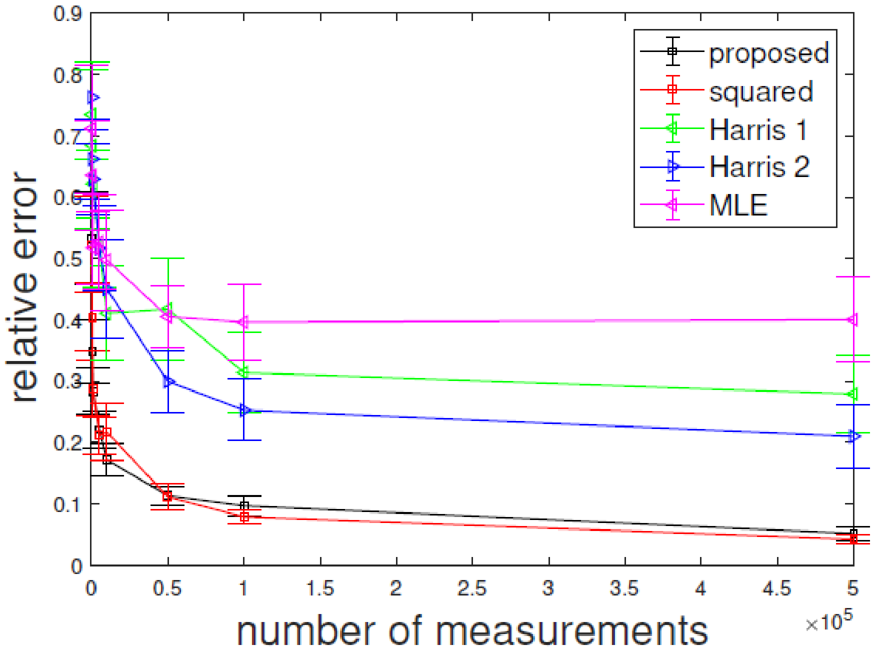
Objective of MLE



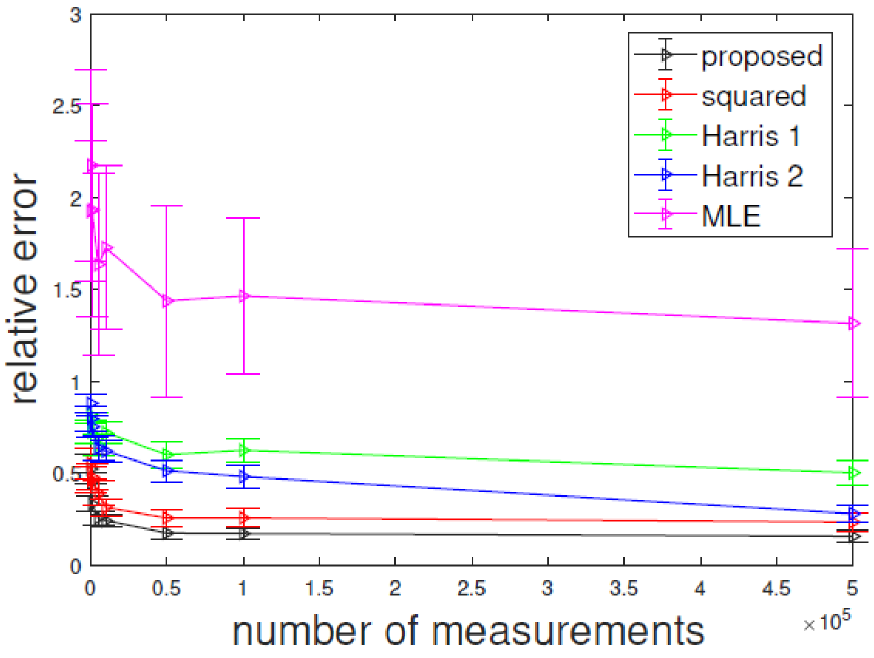
Objective of Laplace fitting

Parameter estimation for tandem of M/M/1 queues: Performance

- **Theorem.** As $n \to \infty$, Laplace fitting has a unique optimal solution that equals the ground truth $\boldsymbol{\delta}$ if $|S| > K$.



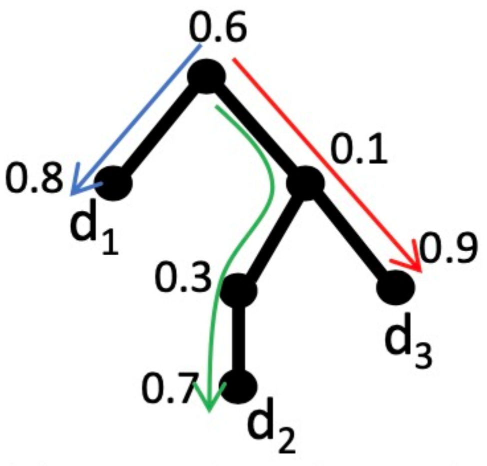
K = 3



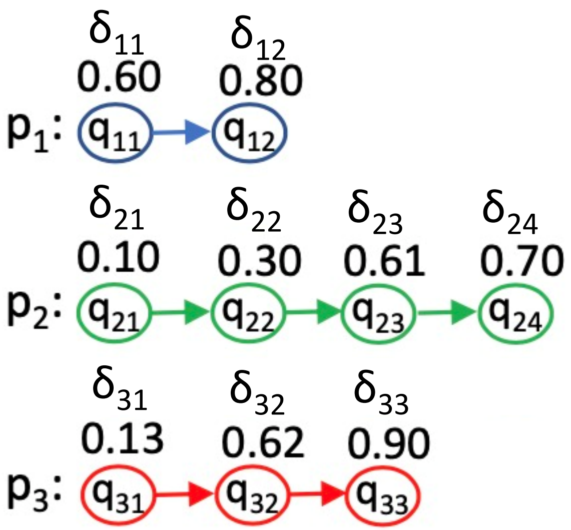
K = 4

Queueing topology inference: idea

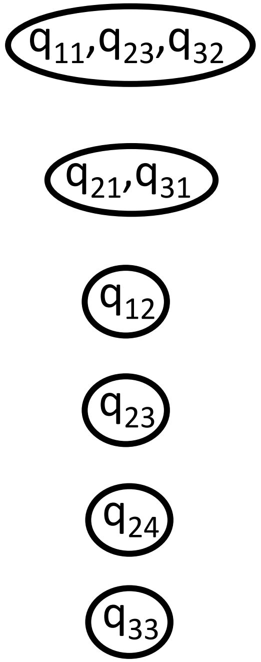
- Ideal case:



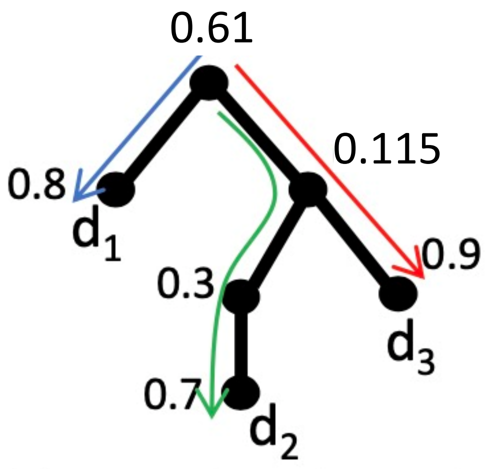
Ground truth topology



Estimated parameters



Parameters associated with the same queue



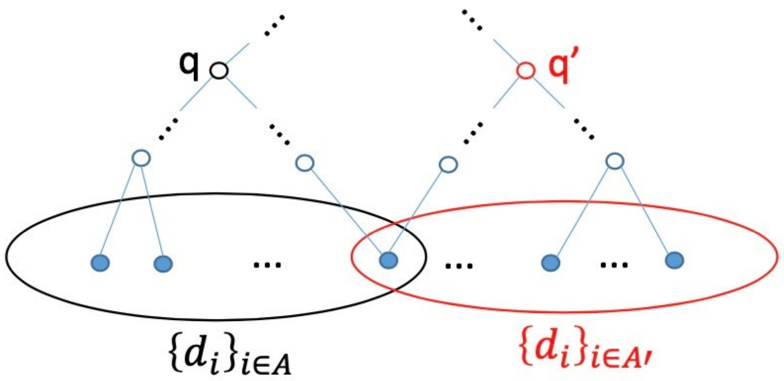
Inferred topology

Queueing topology inference: challenges

- Parameter estimation is not perfect
  - An upper bound Δ, such that

$$D_{\{q_{i_1 j_1},\ldots,q_{i_k j_k}\}} := \max\{\delta_{i_1 j_1},\ldots,\delta_{i_k j_k}\} - \min\{\delta_{i_1 j_1},\ldots,\delta_{i_k j_k}\} \leq \Delta$$

- Topology is not arbitrary
  - Partially overlapping categories cannot coexist
- Exponential complexity if brute-forcing
  - $O(K^N)$ ways to merge queues



Queueing topology inference: solution

- A *greedy* algorithm with *progressively constructed search space* to infer estimated parameters associated with the same queue
  - $O(K^4N^5)$ time complexity, $O(K^2N^3)$ space complexity
  - Correct if estimated parameters are sufficiently accurate

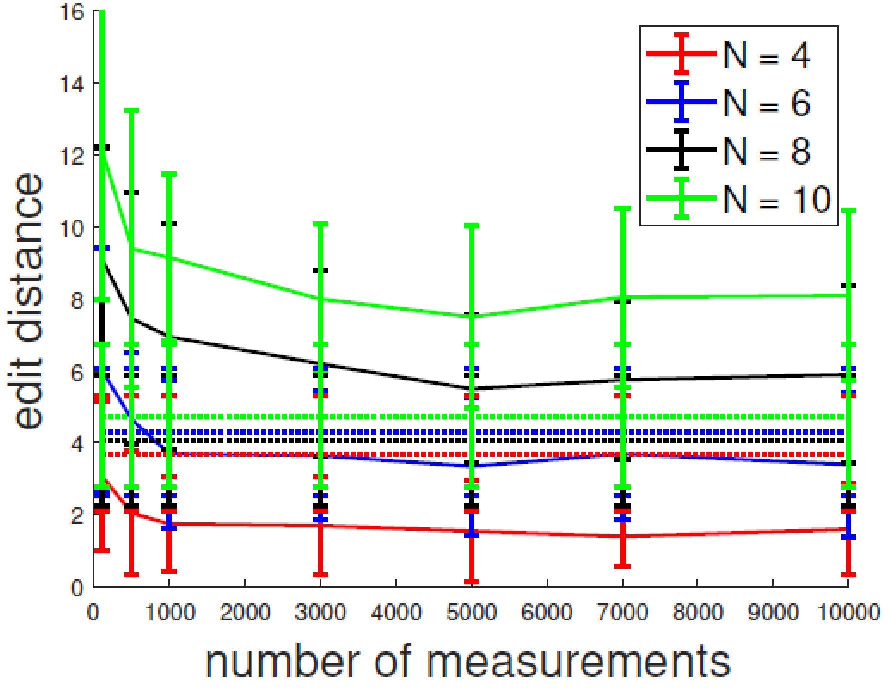
- **Theorem.** All parameters for the same queue are correctly identified if

$$\left|\delta_{ij} - \delta_{ij}^*\right| \leq \frac{\Delta}{2} < \frac{\Delta^*}{4} \quad (\text{where } \Delta^* \coloneqq \min_{e \neq e\prime} |\delta_e^* - \delta_{e\prime}^*|)$$

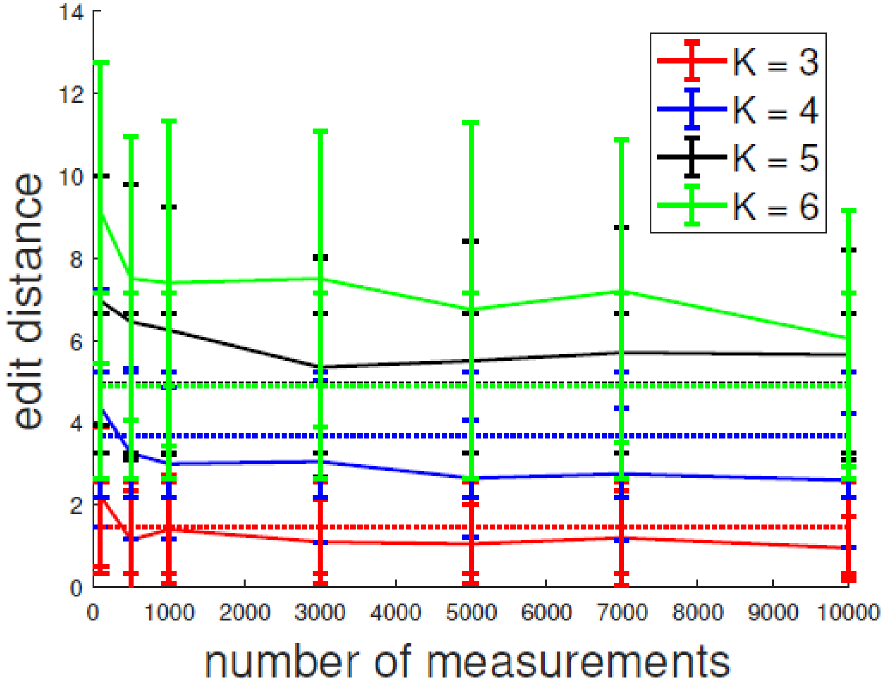
→ Under this condition**, the inferred topology will be identical to the ground truth**, up to a permutation of queues on the same branch.

Performance evaluation

- Routing trees generated from AS6461 of Abovenet



(a) vary N $(K = 4)$

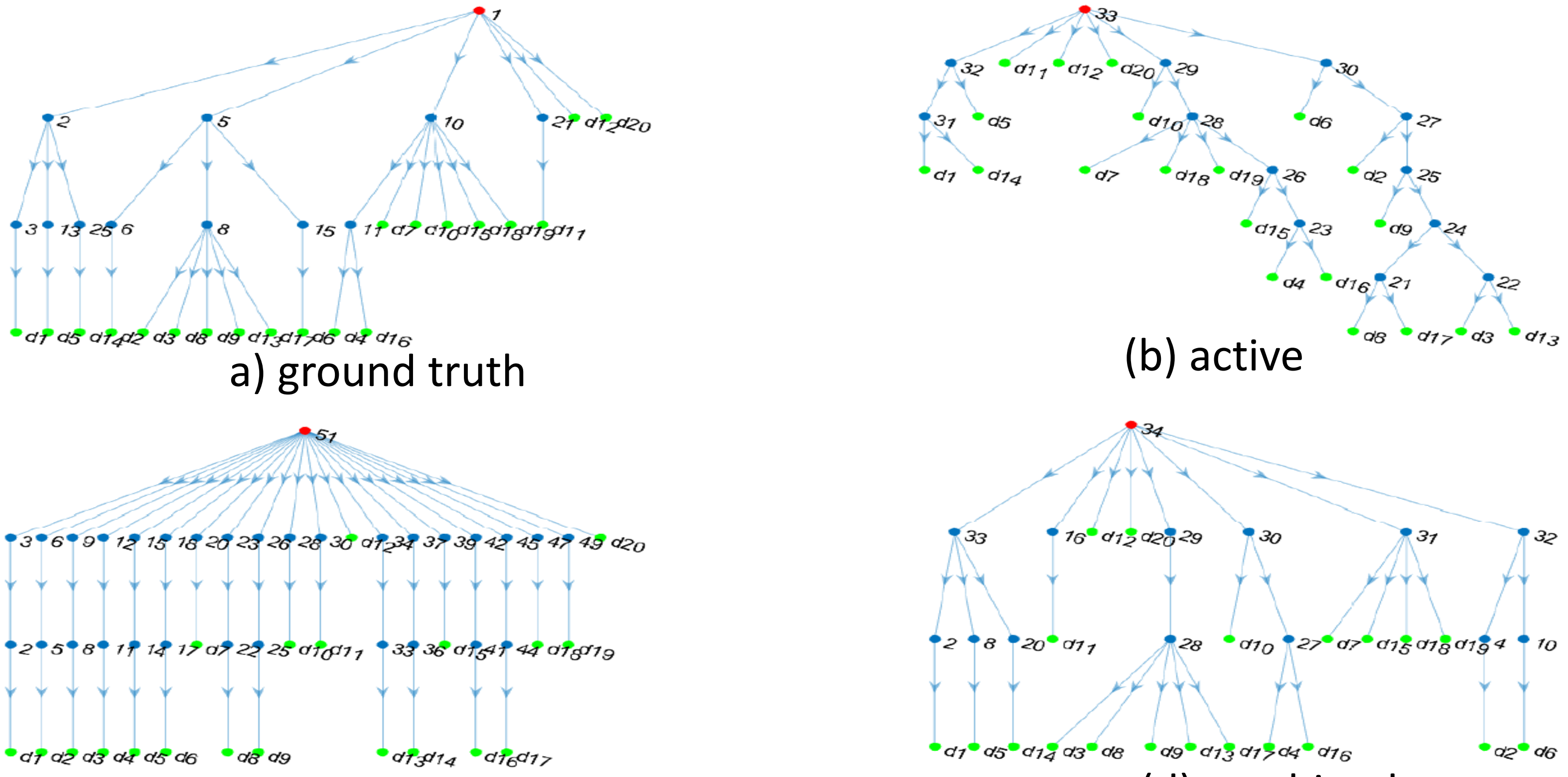


(b) vary K $(N = 5)$

solid line: edit distance for inferred topology; dotted line: edit distance for multicast tree

How to improve the scalability

- Idea: Combining passive & active measurements
  - Passive measurements → queue fingerprints
  - Active measurements → shared path length



a) ground truth

(b) active

(c) passive

(d) combined

Outline

Restriction on measurement

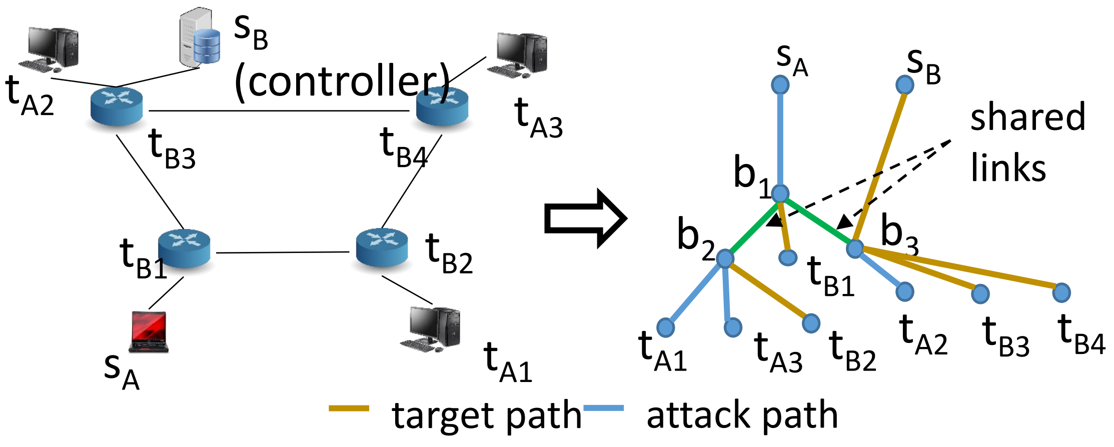
	Tree-based	Waypoint-based	Arbitrary
Probe no path	Queue fingerprinting		
Probe some paths	**Shared link detection**		
Probe all paths	Most existing solutions, e.g., RNJ REA	1-1-N	SAP

Flexibility of routing

Scenario: Cross-path attack

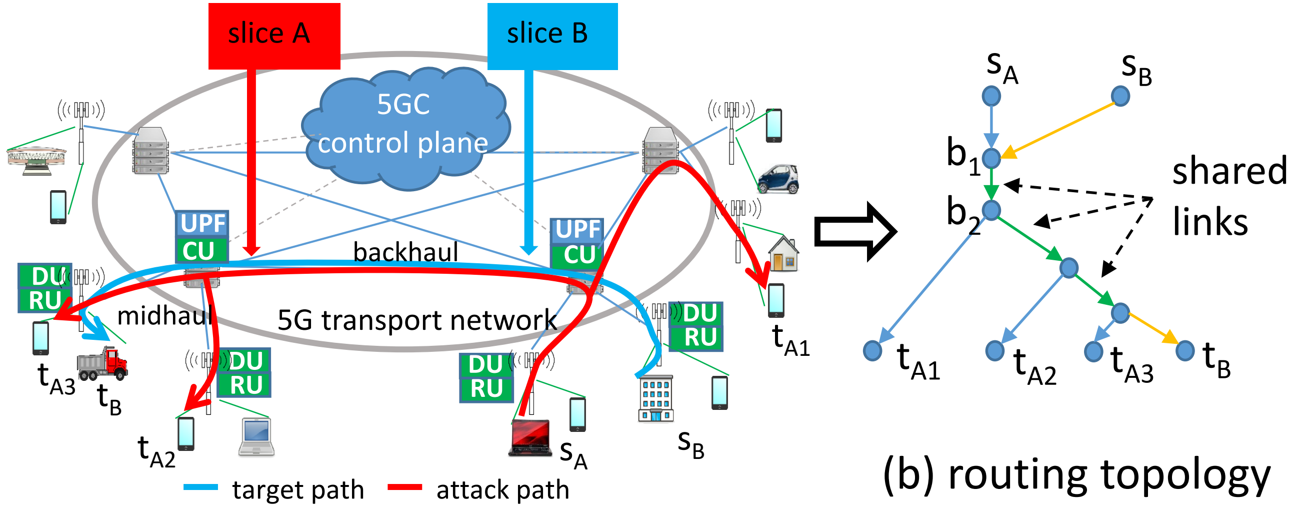
- An attacker in control of a set of *attack paths* wants to launch indirect DoS attack on a set of *target paths* by consuming shared resources

Example 1: Data→Control Plane Attack in SDN



(a) physical network (b) routing topology

Example 2: Cross-slice Attack in 5G



(a) physical network

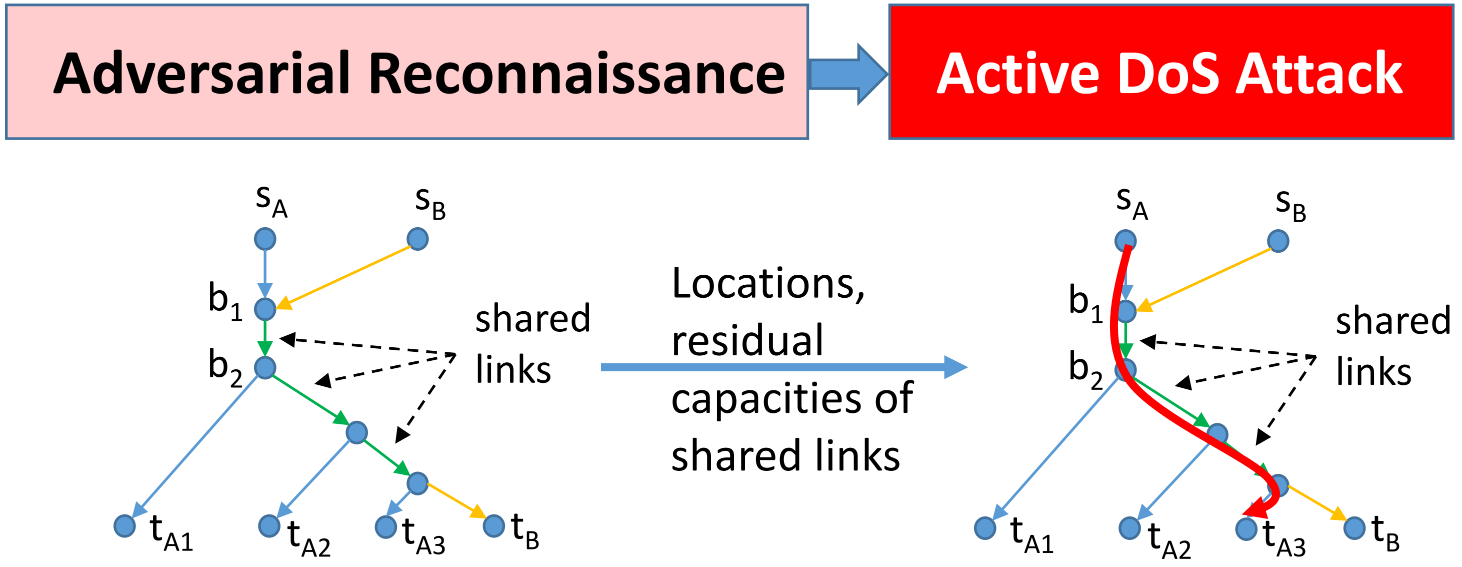
(b) routing topology

Cross-path attack: A high-level description

- Cross-path attack contains a *reconnaissance phase* and an *active attack phase*

Which attack paths share resource with target paths?
What is the capacity of the shared resource?

Which attack paths to use?
How much traffic to send?



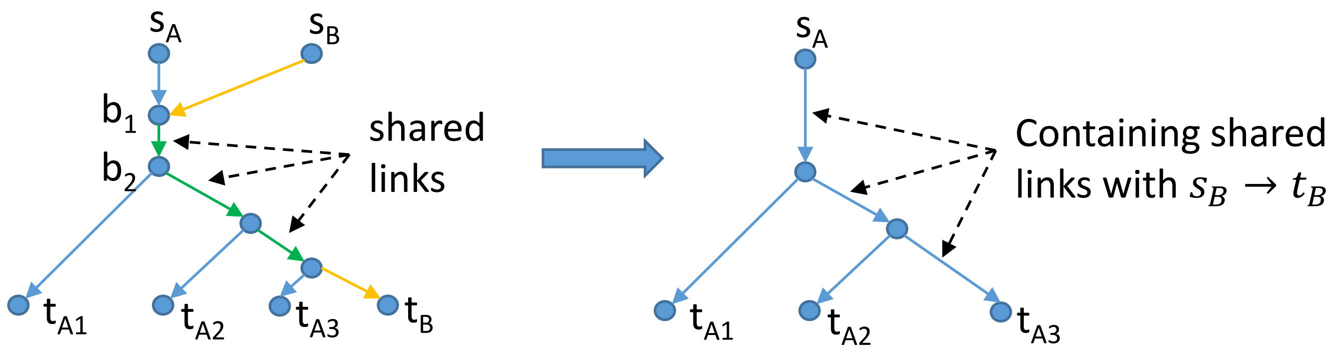
Adversarial Reconnaissance

Active DoS Attack

Locations, residual capacities of shared links

Adversarial reconnaissance: A topology inference problem

- **Observation model**: *Active probing* on attack paths, *passive monitoring* on target paths
- **Goal:** Support optimal attack design
  - Knowing the true routing topology formed by all attack/target paths is sufficient, but not necessary
- **Idea:** Use mimicked multicast to infer "attack paths + 1 target path" topologies



Adversarial reconnaissance: Results

- Recursive algorithm to detect shared links
  - **Theorem.** If all shared links have non-zero metrics and **category weights are estimated accurately**, then all **shared links will be correctly detected**.

- Recursive algorithm to estimate parameters of detected shared links
  - Modeled as M/M/1, M/D/1, or G/G/1 queue
  - Estimated by fitting average delay under K different probing rates
  - **Theorem.** If all shared links are correctly detected, and the **average delays on target paths are accurately estimated**, then the **parameters of shared links will be accurately estimated** if (i) $K > 2$ under M/M/1 or M/D/1, and (ii) $K > 4$ under G/G/1

Attack design: Objectives and results

- Objective 1: Delay maximization

$$\begin{aligned}
\max f(\bar{\boldsymbol{\lambda}}) &:= \sum_{i=1}^{N_B} \beta_i \sum_{e \in \mathcal{T}: W_{ie}>0} d(\xi_{ie}; \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k) \\
\text{s.t. } & \sum_{k=1}^{N_A} \bar{\lambda}_k \le \lambda, \\
& \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k \le \tilde{r}_e, \ \forall e \in \mathcal{T}, \\
& \bar{\lambda}_k \ge 0, \ k = 1, \ldots, N_A,
\end{aligned}$$

- Objective 2: Overload maximization

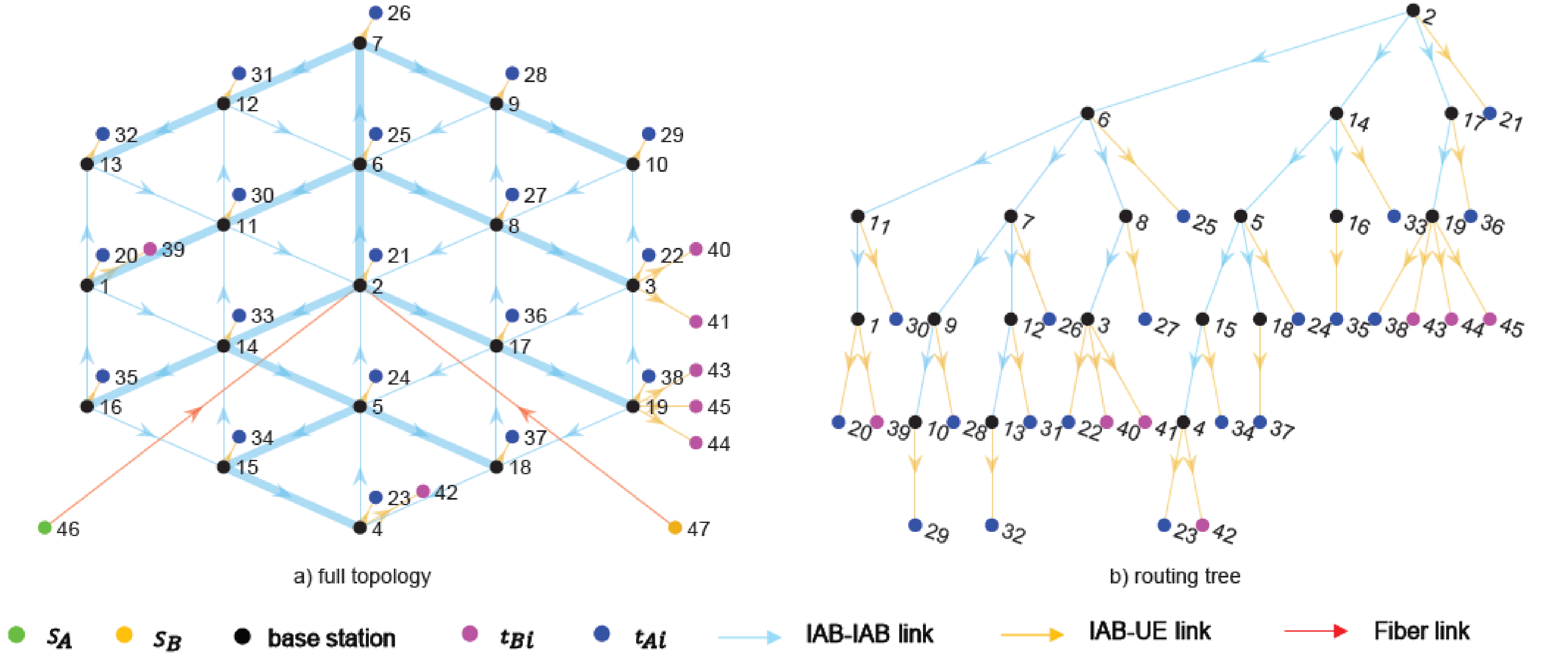
$$\max_{\bar{\boldsymbol{\lambda}}} \max_{e \in \mathcal{T}: \exists W_{ie}>0} \left(\sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k - \min_{i \in \{1,\ldots,N_B\}: W_{ie}>0} r_{ie} \right)$$

$$\text{s.t. } \sum_{k=1}^{N_A} \bar{\lambda}_k \le \lambda, \quad \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k \le \tilde{r}_e, \ \forall e \in \mathcal{T}, \quad \bar{\lambda}_k \ge 0, \ k = 1, \ldots, N_A,$$

Both **maximizing convex function under linear constraints**
→ Optimum at a vertex
→ If attack rate $\lambda \le \min_{e \in T} \tilde{r}_e$, optimal to send all attack traffic on one attack path

Performance evaluation: NS3 + 5G Lena

- Scenario: 5G IAB (Integrated Access and Backhaul) network

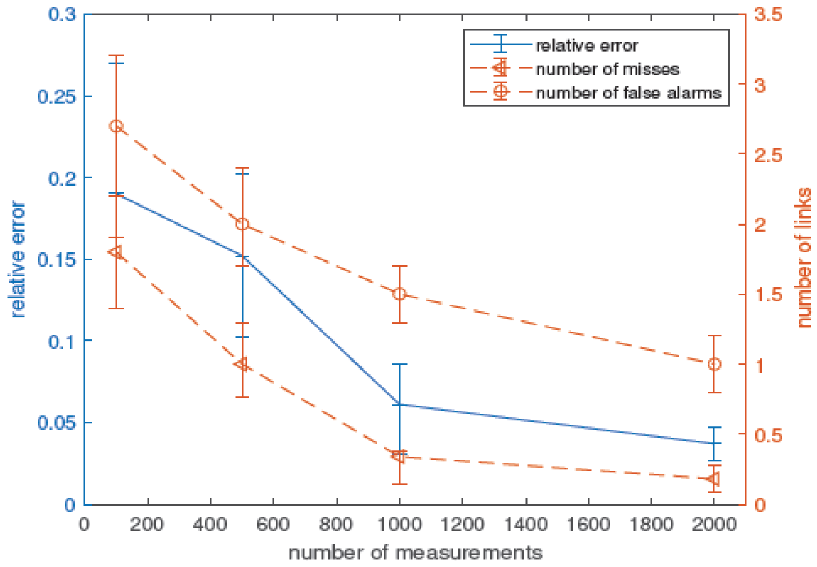


a) full topology

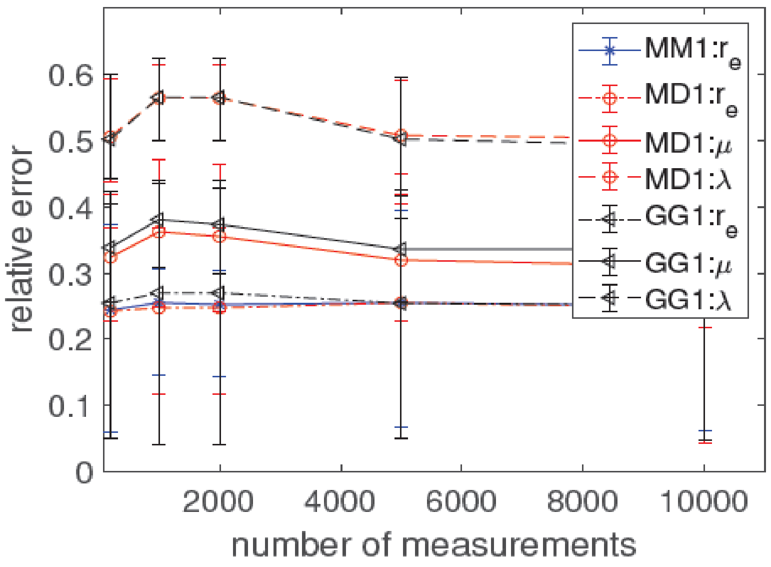
b) routing tree

- ON-OFF traffic, discrete packet sizes

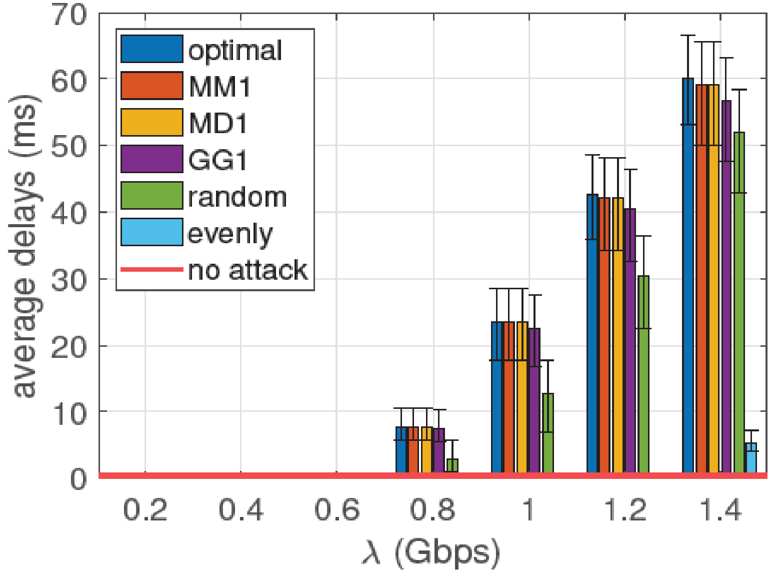
Performance evaluation: Results



(a)



(b)



(c)

a) Can detect most of the shared links

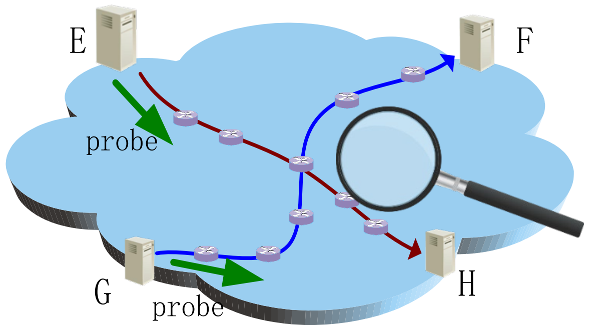
b) Notable error in estimated parameters

c) Near-optimal performance in attack design

Concluding Remark

- Topology inference: Jointly infer network *internal structure* from *external observations*
  - what "internal structure" to infer, what structures are possible, what measurements are allowed

  → A double-sided sword (overlay management vs. adversarial reconnaissance)



Network structure & state = ?

Restriction on measurement

	Tree-based	Waypoint-based	Arbitrary
Probe no path	Queue fingerprinting		
Probe some paths	Shared link detection		
Probe all paths	Most existing solutions, e.g., RNJ REA	1-1-N	SAP

Flexibility of routing

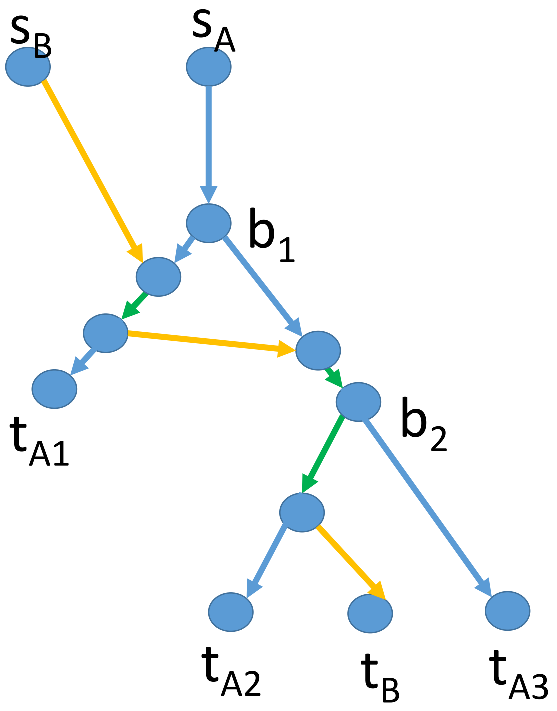
CT Scan for Network: Topology Inference from End-to-End Measurements

Ting He, tinghe@psu.edu

THANK YOU

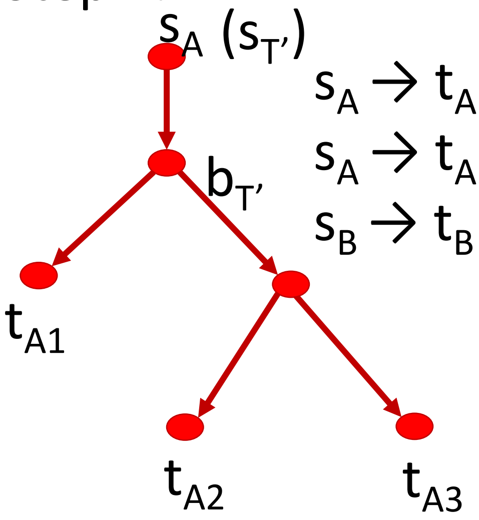
Backup slides

Example: Shared link detection



Ground truth

Step 1:

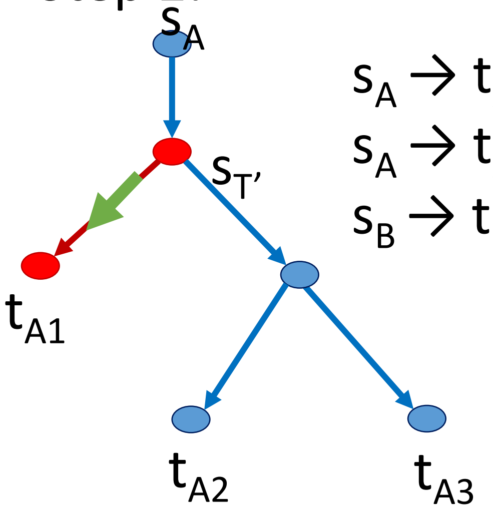


$s_A \rightarrow t_{A1}$

$s_A \rightarrow t_{A2}$

$s_B \rightarrow t_B$

Step 2:

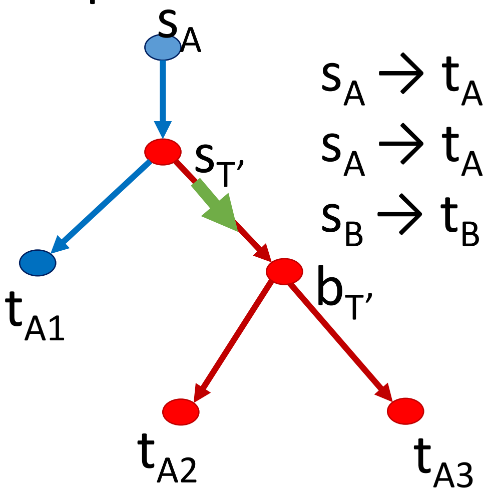


$s_A \rightarrow t_{A1}$

$s_A \rightarrow t_{A1}$

$s_B \rightarrow t_B$

Step 3:

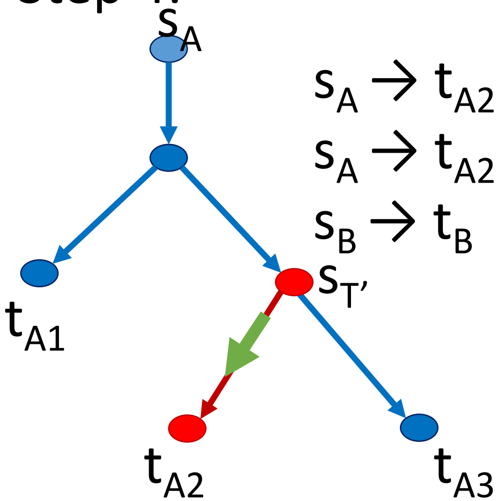


$s_A \rightarrow t_{A2}$

$s_A \rightarrow t_{A3}$

$s_B \rightarrow t_B$

Step 4:

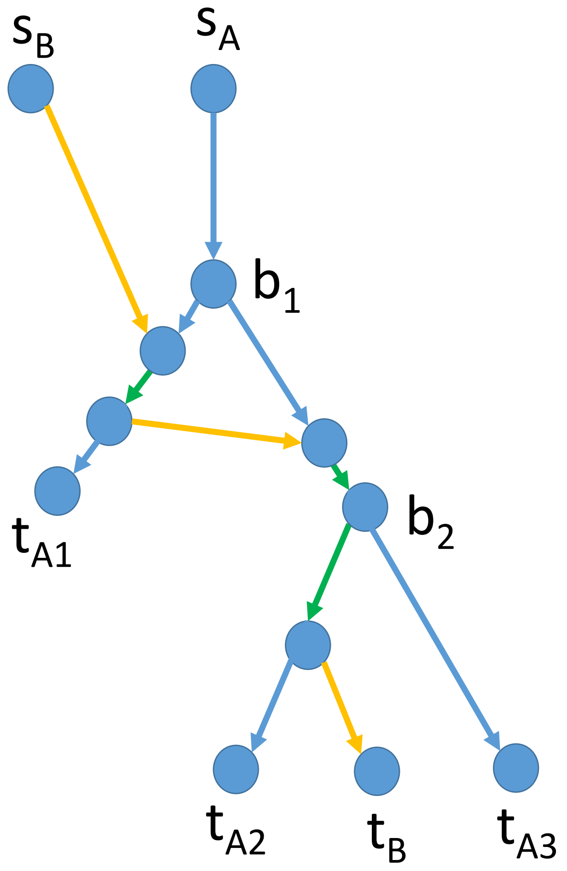


$s_A \rightarrow t_{A2}$

$s_A \rightarrow t_{A2}$

$s_B \rightarrow t_B$

Parameter Estimation



Ground truth

Top-down: One queue at a time

Attack rate

parameter

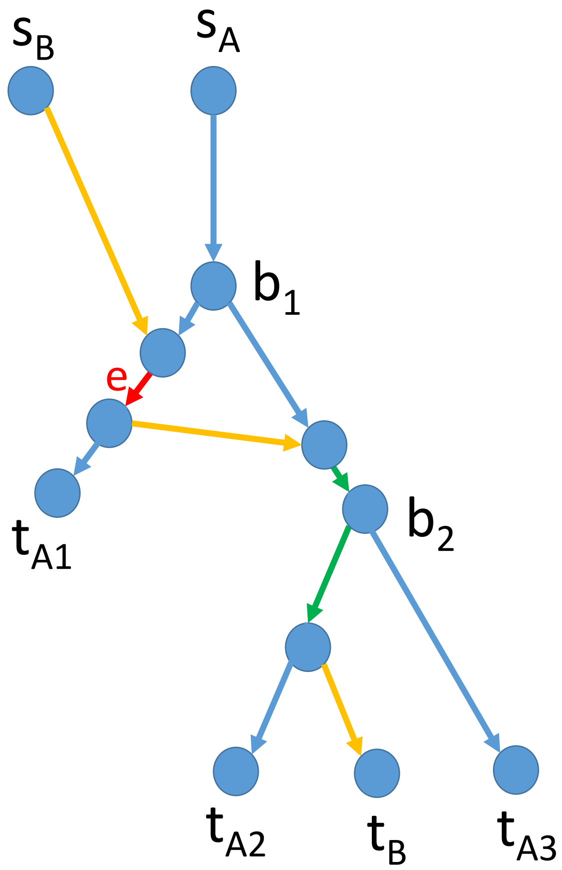
M/M/1: $d(r_e;\bar{\lambda}) = \dfrac{1}{r_e - \bar{\lambda}}$

M/D/1: $d(\lambda_e,\mu_e;\bar{\lambda}) = \dfrac{2\mu_e - \lambda_e - \bar{\lambda}}{2\mu_e(\mu_e - \lambda_e - \bar{\lambda})}$

G/G/1: $d(\lambda_e,\mu_e,\sigma_{ae},\sigma_{se};\bar{\lambda}) \approx$

$$\frac{1}{2\mu_e}\frac{\lambda_e+\bar{\lambda}}{\mu_e-\lambda_e-\bar{\lambda}}\left(\sigma_{ae}^2(\lambda_e+\bar{\lambda})^2+\sigma_{se}^2\mu_e^2\right)+\frac{1}{\mu_e}$$

Parameter Estimation



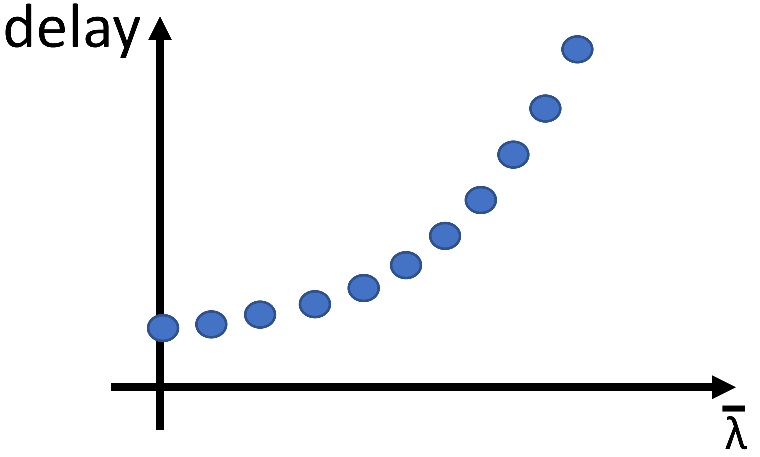
Ground truth

M/M/1: $d(r_e;\bar{\lambda}) = \dfrac{1}{r_e - \bar{\lambda}}$

Send probes s_A->t_{A1} with rate $\bar{\lambda}$ = 0, …, r/2

Measure delay of path s_B->t_B

$$\underbrace{c_{\mathcal{T}^*}}_{\text{Constant}} + \frac{1}{r_e - \bar{\lambda}}$$



Theorem:

Accurate delay

& enough dimension

⇓

Accurate parameter