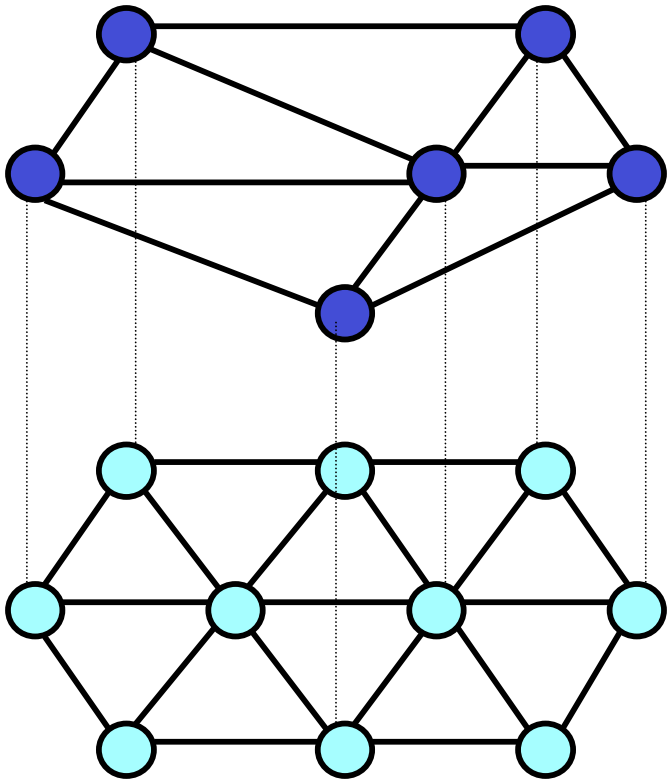
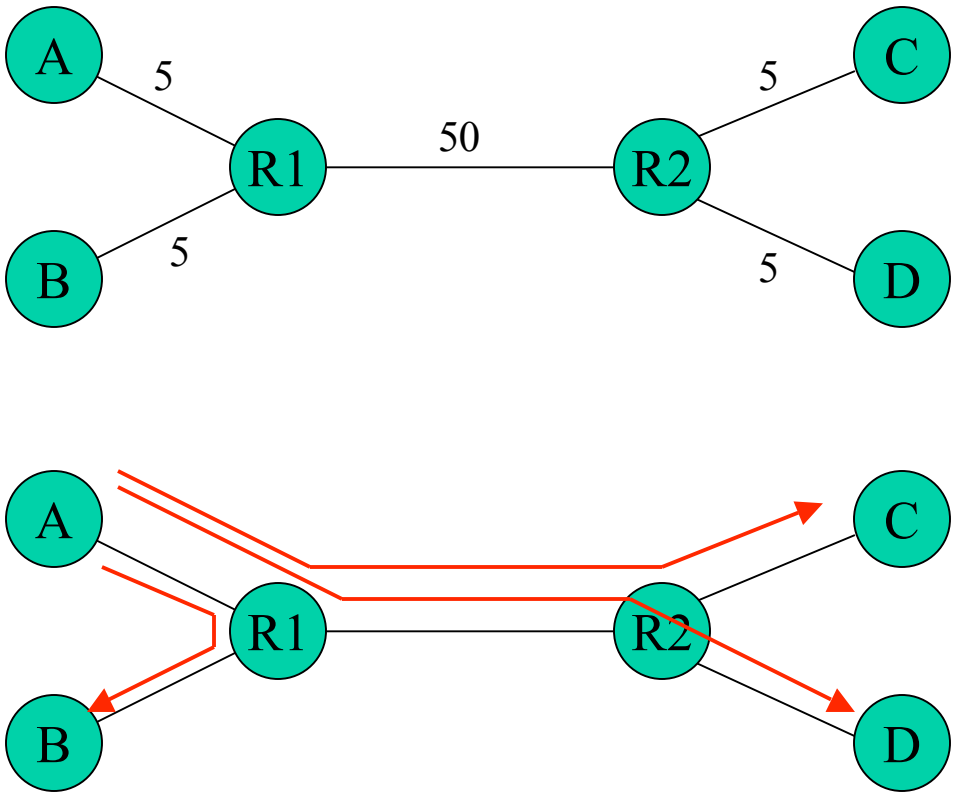
Overlay Routing
(Routing Underlay)

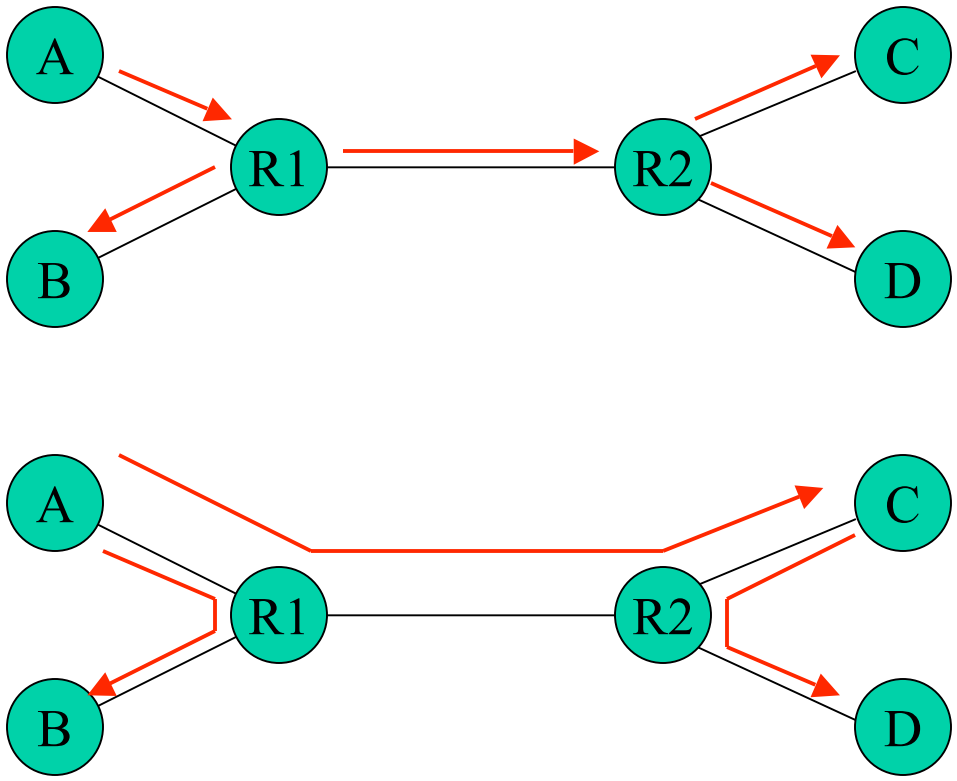
Overlay Networks



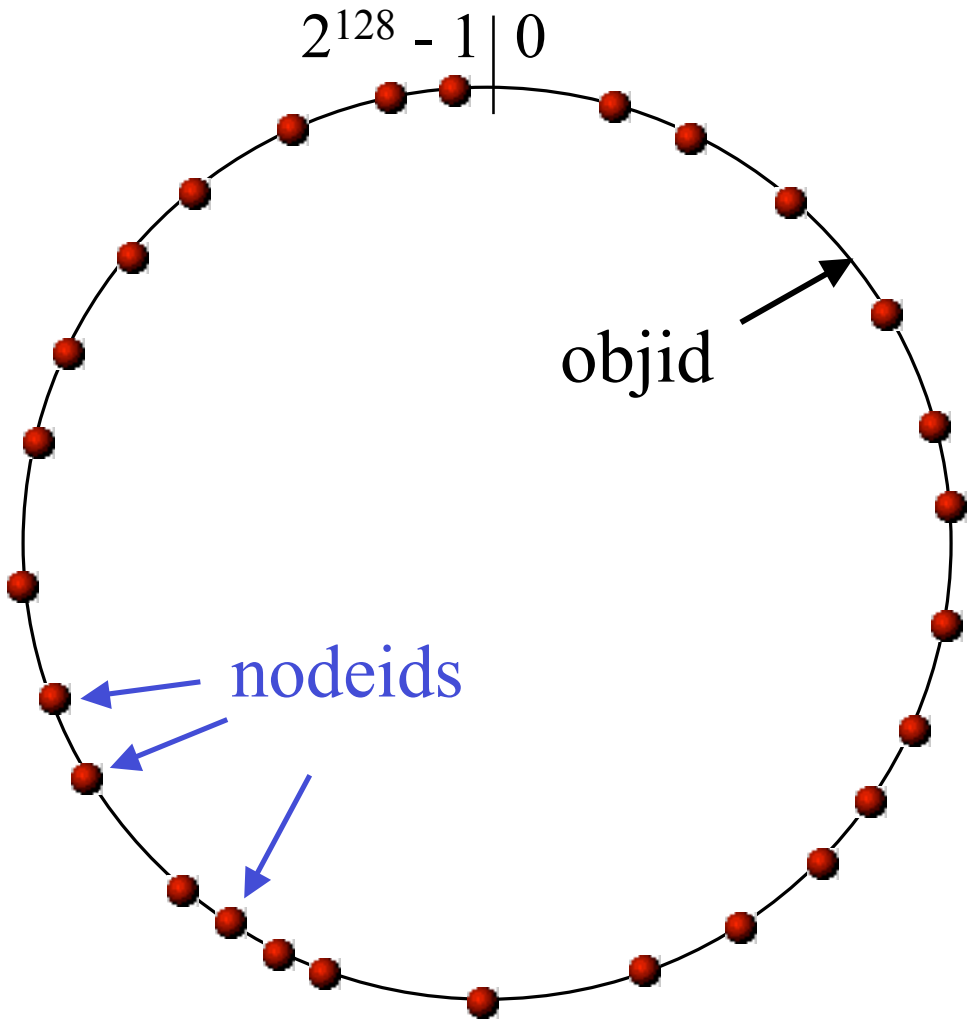
End System Multicast



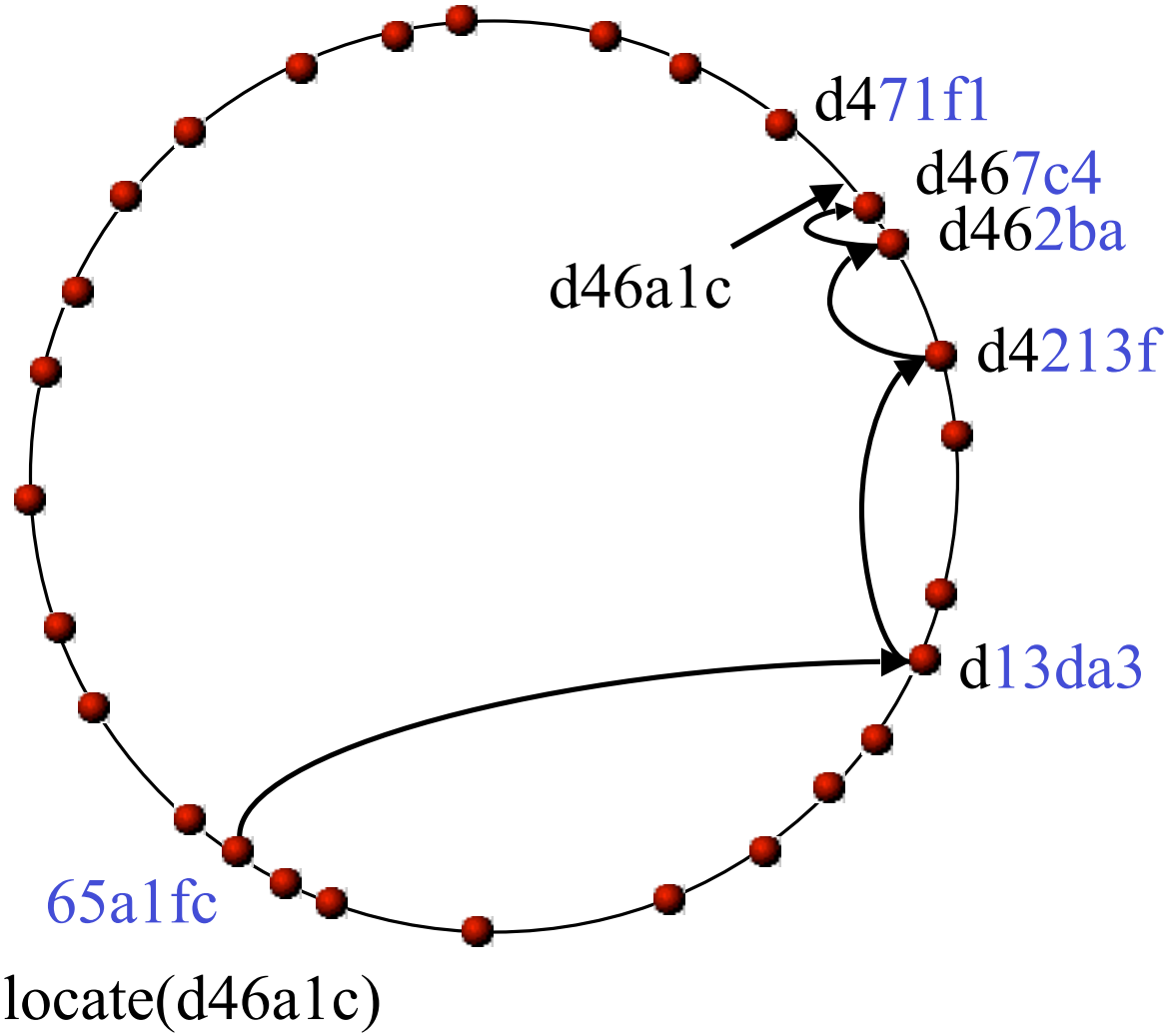
ESM (cont)



Distributed Hash Tables

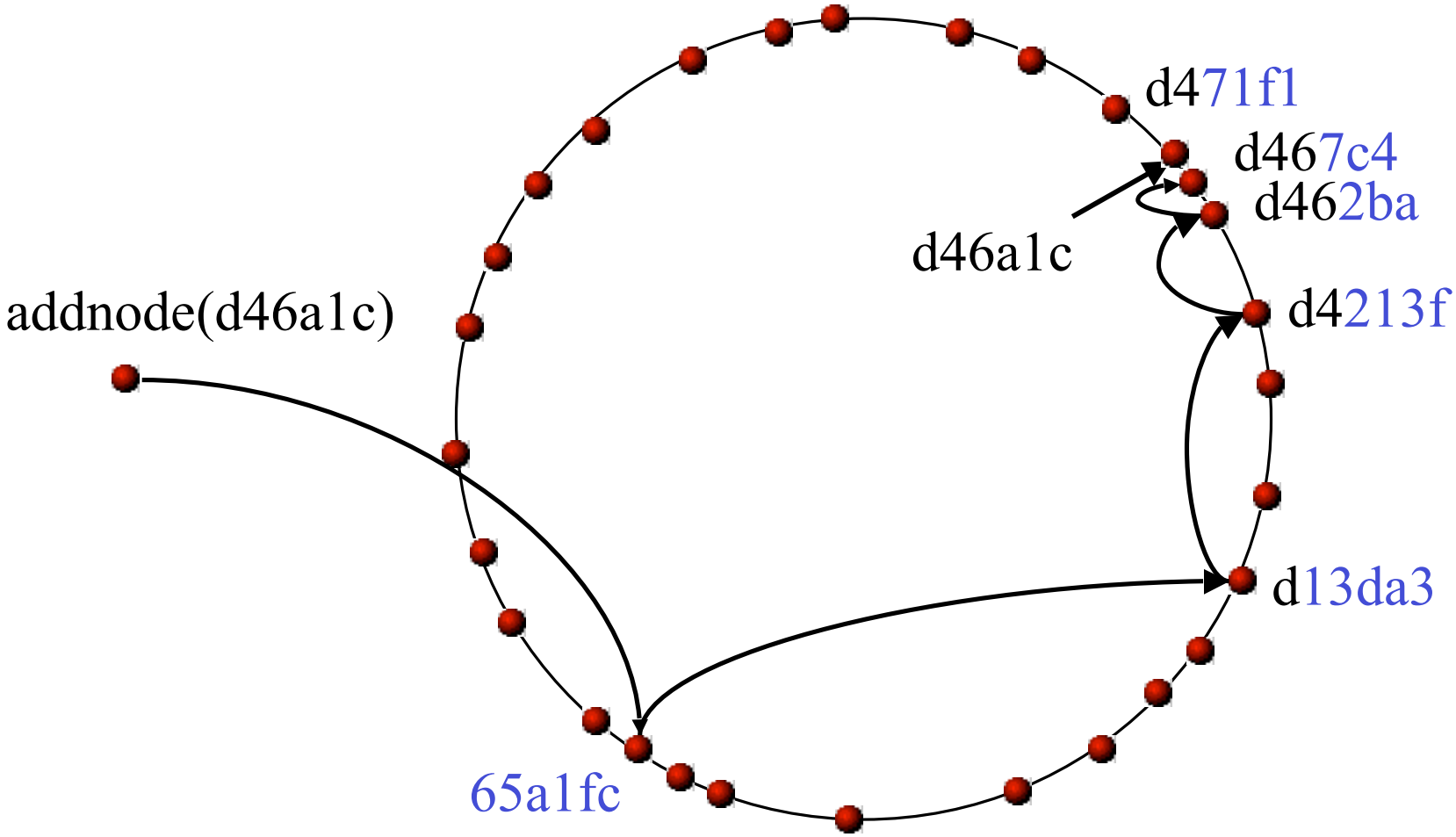


DHT (cont)

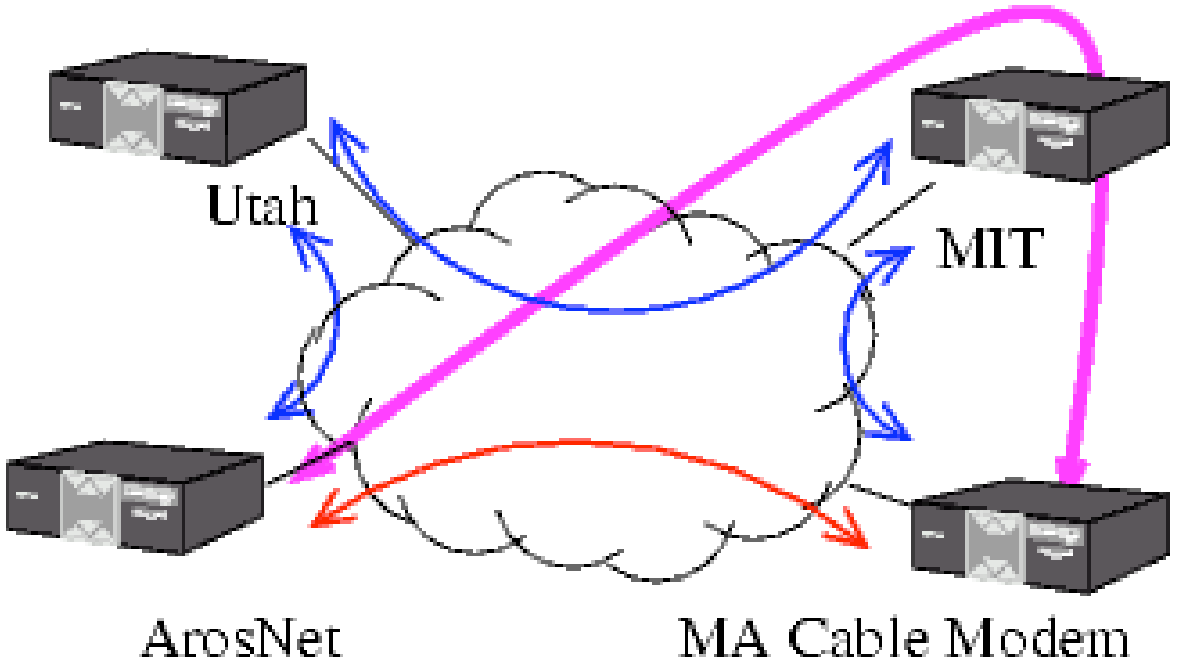


locate(d46a1c)

DHT (cont)


d471f1
d467c4
d462ba
d46a1c
addnode(d46a1c)
d4213f
d13da3
65a1fc

Resilient Overlay Networks

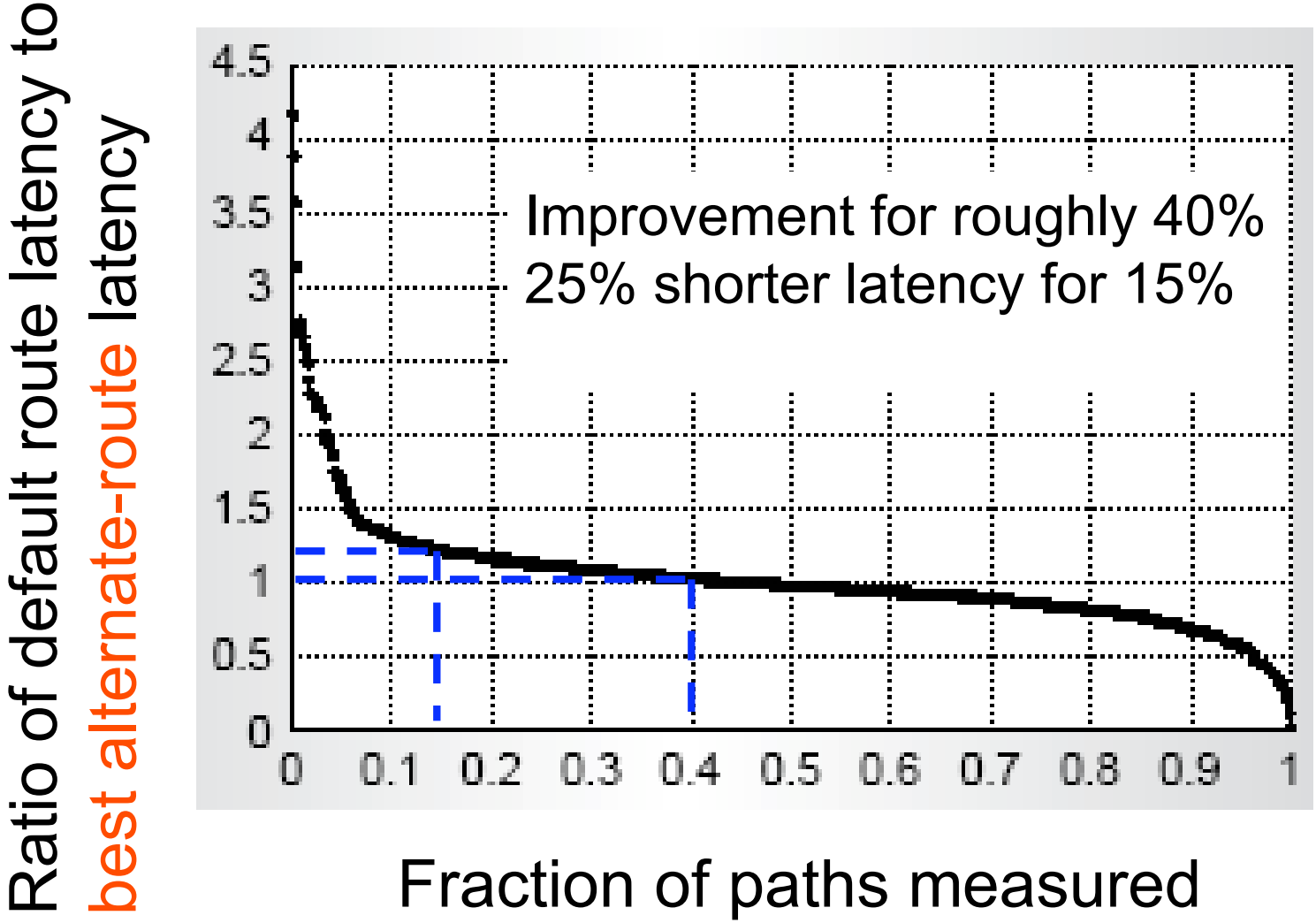


RON (cont)

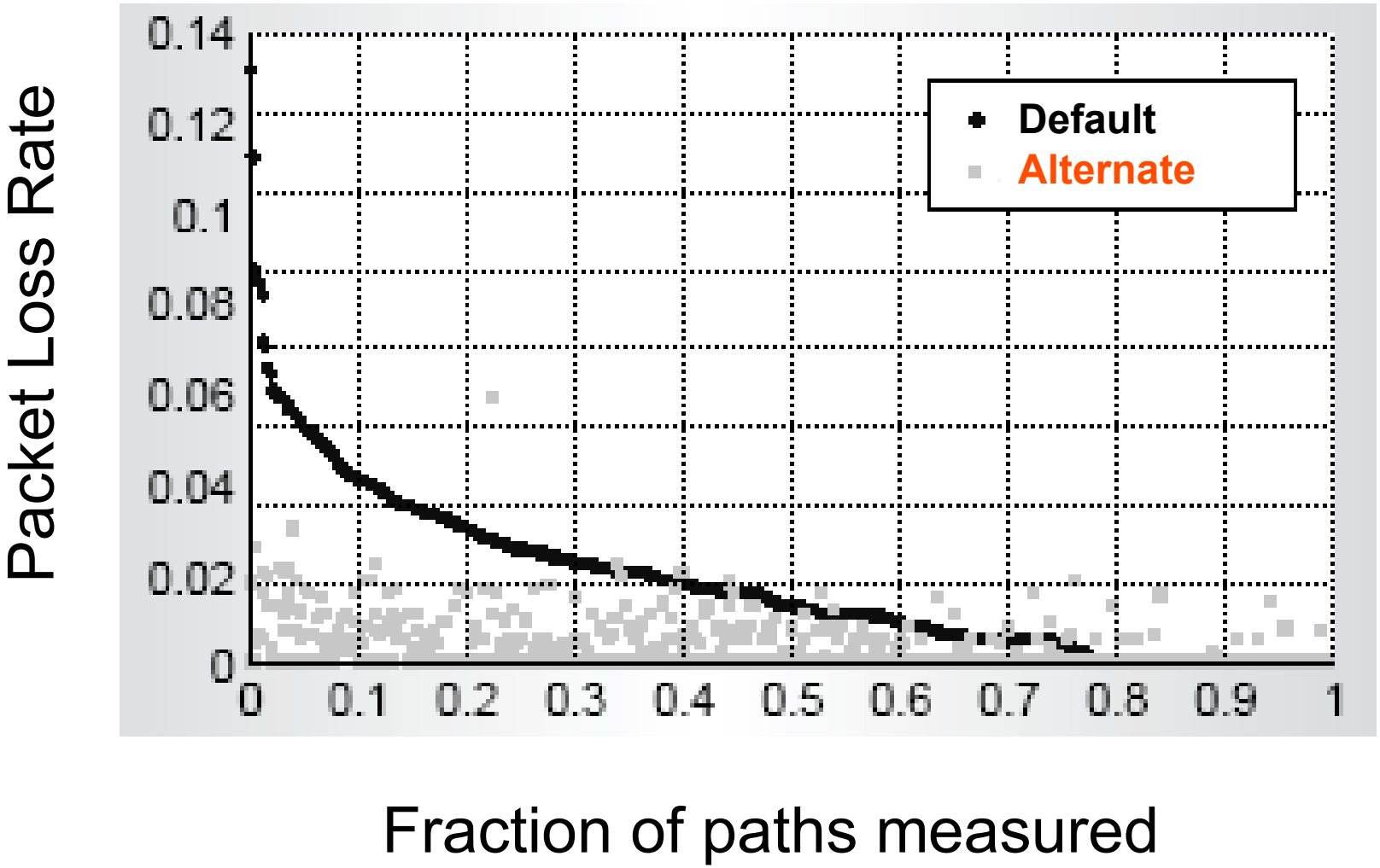
BGP inefficiencies

- Poor Metrics
  - minimize AS hops
- Long failover times
  - measured in minutes
- Manual Load Balancing
  - configuration errors
- Single Path
  - under-utilize alternate paths

RON (cont)


Ratio of default route latency to best alternate-route latency
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Improvement for roughly 40%
25% shorter latency for 15%
Fraction of paths measured

RON (cont)


Packet Loss Rate
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Default
Alternate
Fraction of paths measured

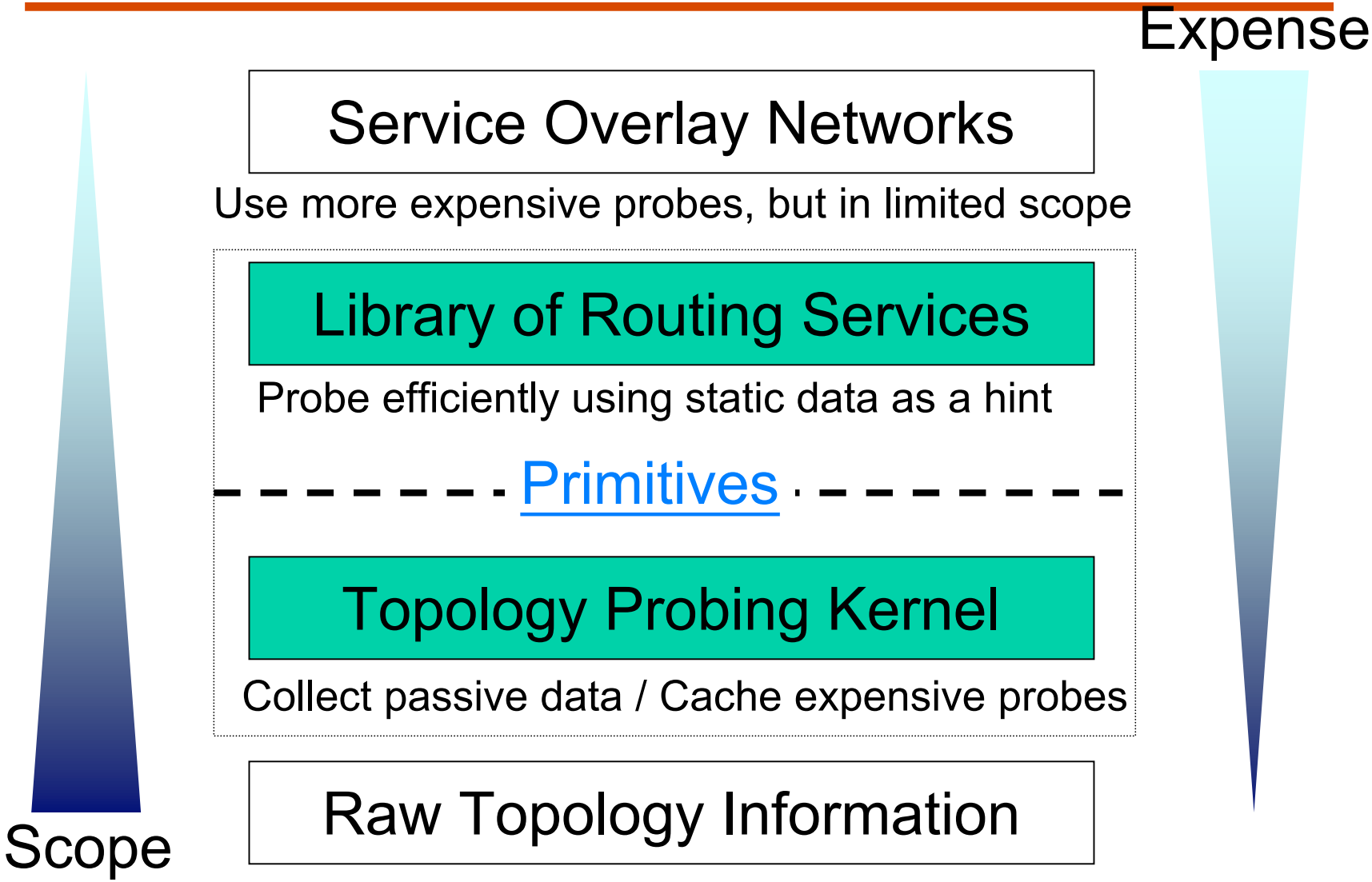
Problem

- Discovering efficient topology requires expensive/disturbing network probes
- Single overlay network
  - aggressive probing does not scale (RON)
- Multiple overlay networks
  - Redundant probing to discover the same topological information
  - 1GB-per-day of ping traffic on PlanetLab
    - one ping-per-sec-per-node across 125 nodes

Routing Underlay

- Sits between overlays and the Internet
- Exposes topological information
  - already collected by the Internet (BGP tables)
  - caches active measurements
- Enables cost-effective network probes
  - primitives: interface to shared probes
  - layered architecture: hierarchical probes

Hierarchical Probes



Primitives

- GetGraph (resolution, scope) _ connectivity
- GetPath (resolution, from, to) _ route
- GetDistance (metric, from, to) _ distance

resolution = AS level, router level, ….

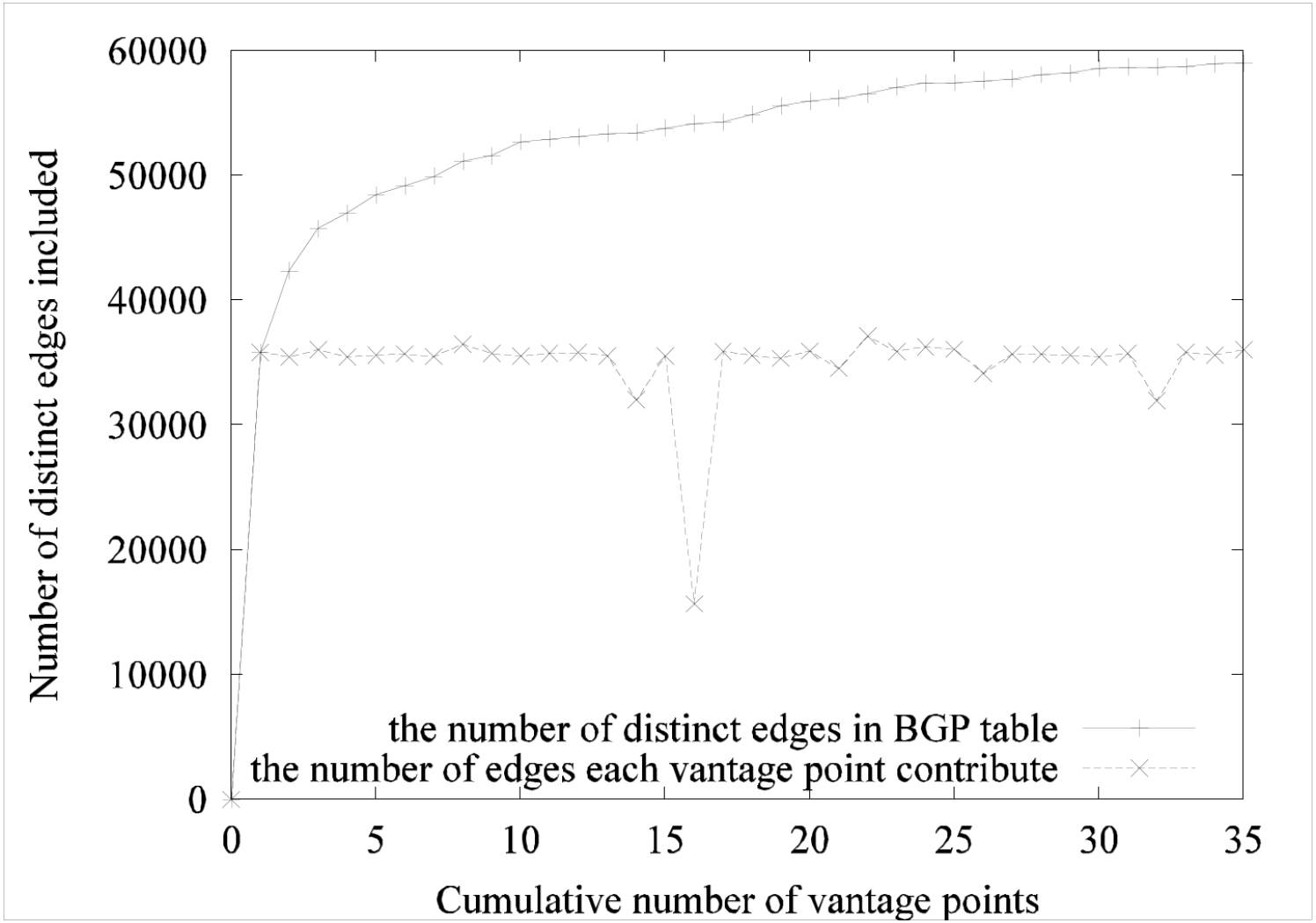
scope = entire network, within an ISP, ….

metric = AS hop, router hop, RTT ….

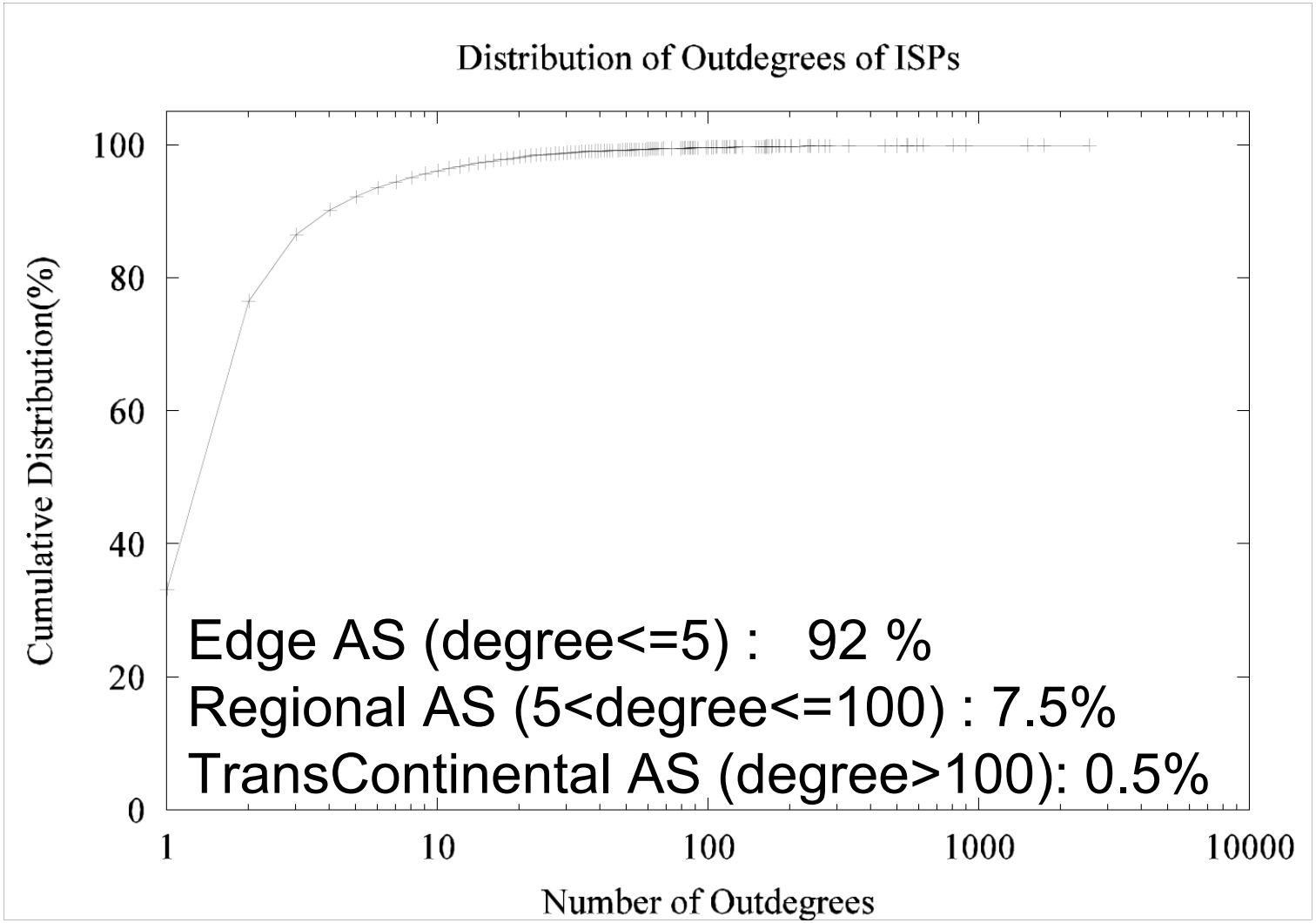
Primitives at AS Level

- Resolution = AS Level
  - GetGraph → AS Peering Graph
  - GetPath → AS Path
  - GetDistance → AS hop count
- Helpers
  - GetPrefixMap → IP to AS translation
  - …

Peering Graph Completeness



Degree Distribution



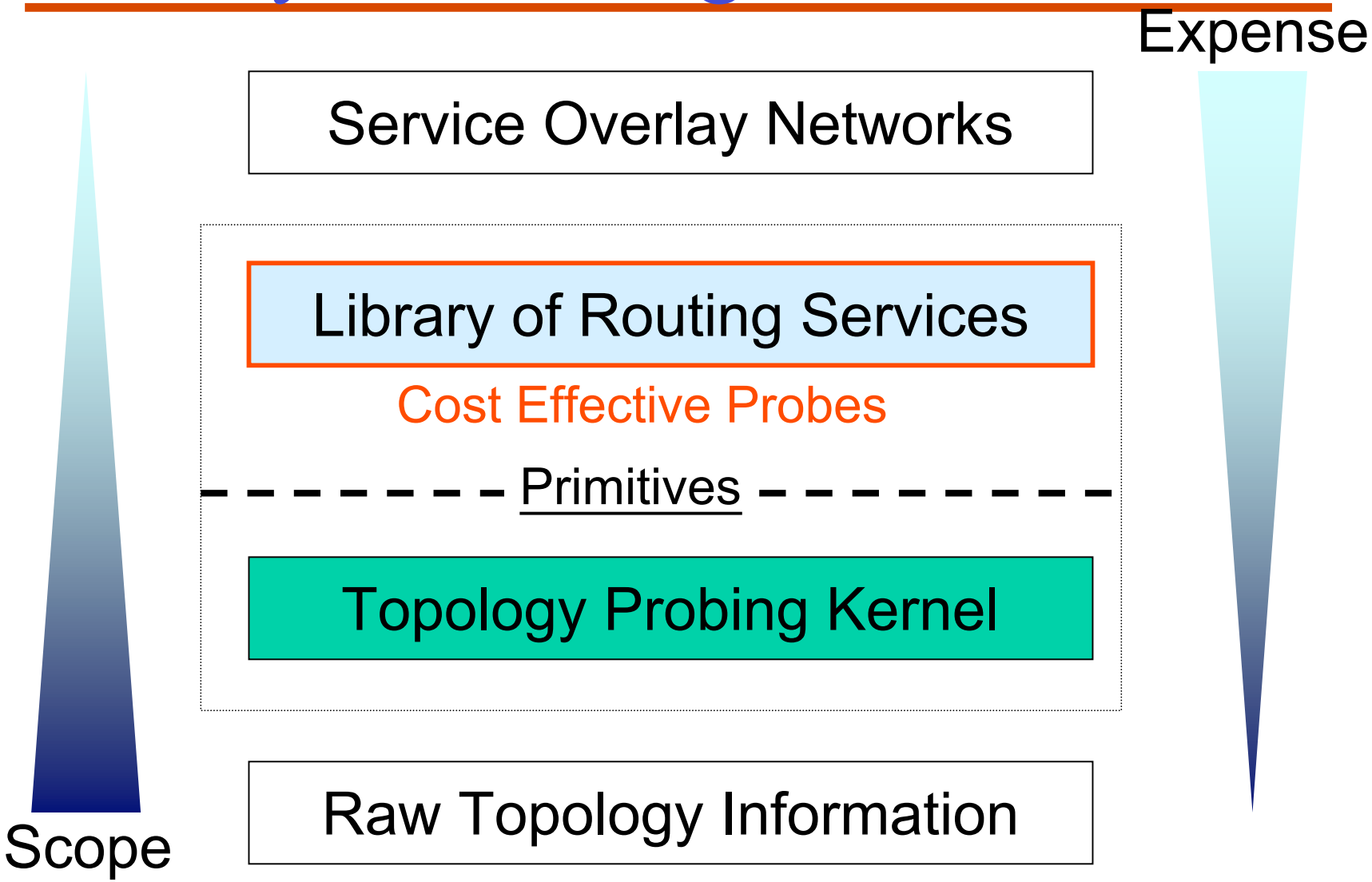
Degree Distribution

Degree (# of peers)	Cumulative Distribution (%)
1	33.26
2	76.55
3	86.56
5	92.26
10	96.13
20	98.20
50	99.30
99	99.65

Top 10 players

Degree	ASN	Organization
2554	701	UUNET Technologies, Inc.
1733	1239	Sprint
1502	7018	AT&T WorldNet Services
890	209	Qwest
798	3561	Cable & Wireless USA
621	1	Genuity
589	702	UUNET Technologies, Inc
589	3549	Global Crossing
545	3356	Level 3 Communications, LLC
541	2914	Verio, Inc.

Library of Routing Services



Library of Routing Services

- Strategy for efficient probes: Guess & Verify
  - Guess candidate solutions using inexpensive probes over a large scope
  - Verify them with more expensive probes in a limited scope
- Example Library Services

  (1) Nearest neighbors (DHT-based routing)

  (2) Edge-disjoint paths (multi-path routing)

  (3) Physically representative mesh (RON, ESM)

Nearest Neighbors

NodeSet = NearestNodes(*N*,*k*)

N : a given set of nodes
k : the number of neighbors
NodeSet: a set of *k* nodes in *N* that are closest to the local overlay node, in terms of latency.

Nearest Neighbors (cont)

Basic idea:

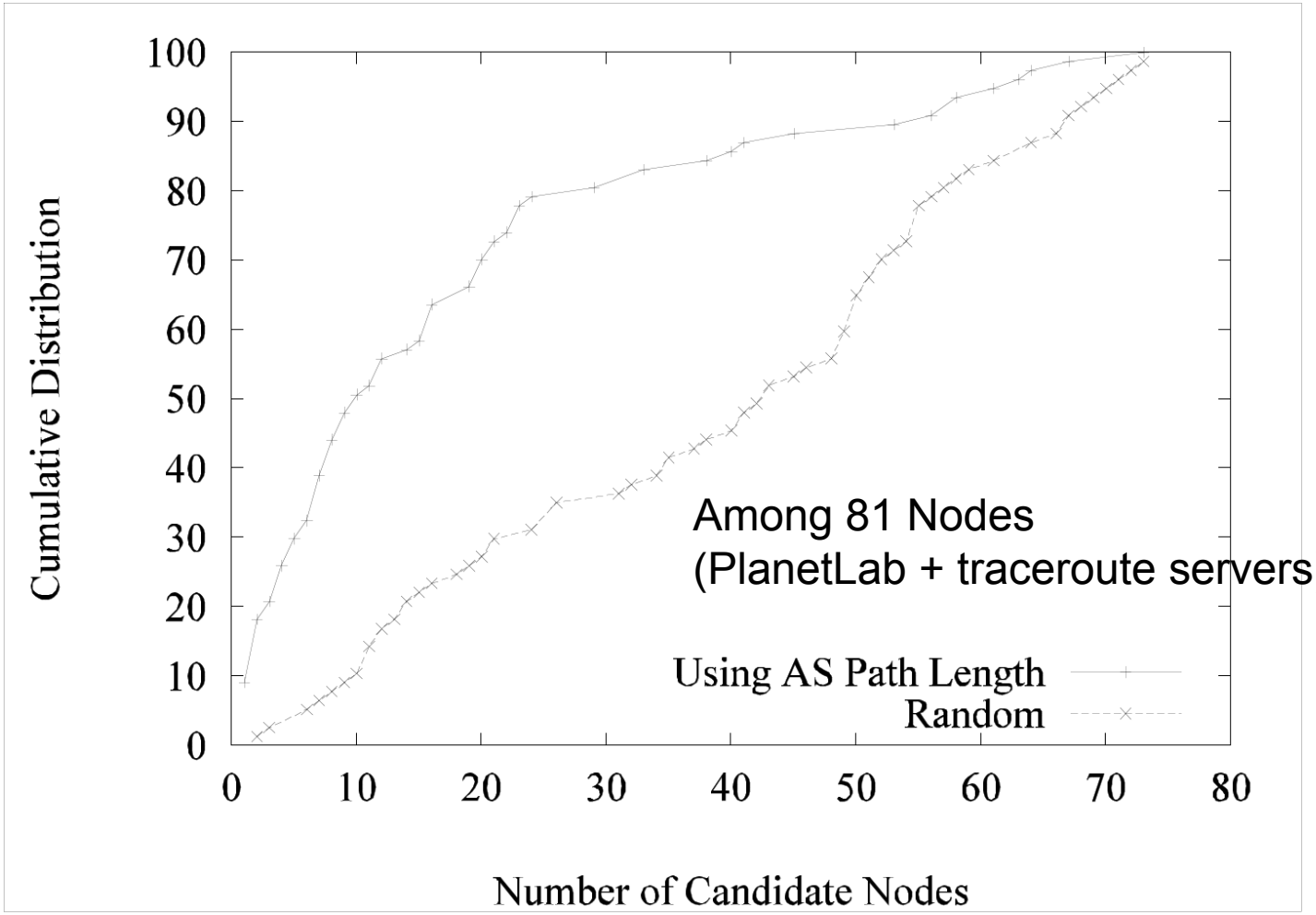
Use weak correlation between latency and AS hops

For a given local node *u*, …

(1) Sort *w* 's in *N* according to AS hop count (GetPath(*u,w*)) into a candidate sequence {*w's*}

(2) Invoke GetDistance(*u,w,ping*) on the first *j* nodes in {*w's*}, and select the best *k* nodes as nearest neighbors (*j>k*)

Nearest Neighbors (cont)



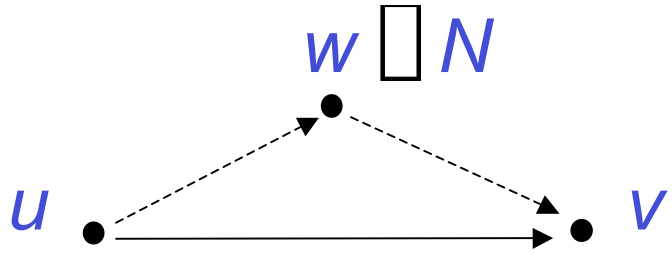
Disjoint Paths

PathSet = DisjointPaths (*u, v, N, k*)

u, v : a given pair of overlay nodes

N : a set of intermediate nodes

k : the number of disjoint paths



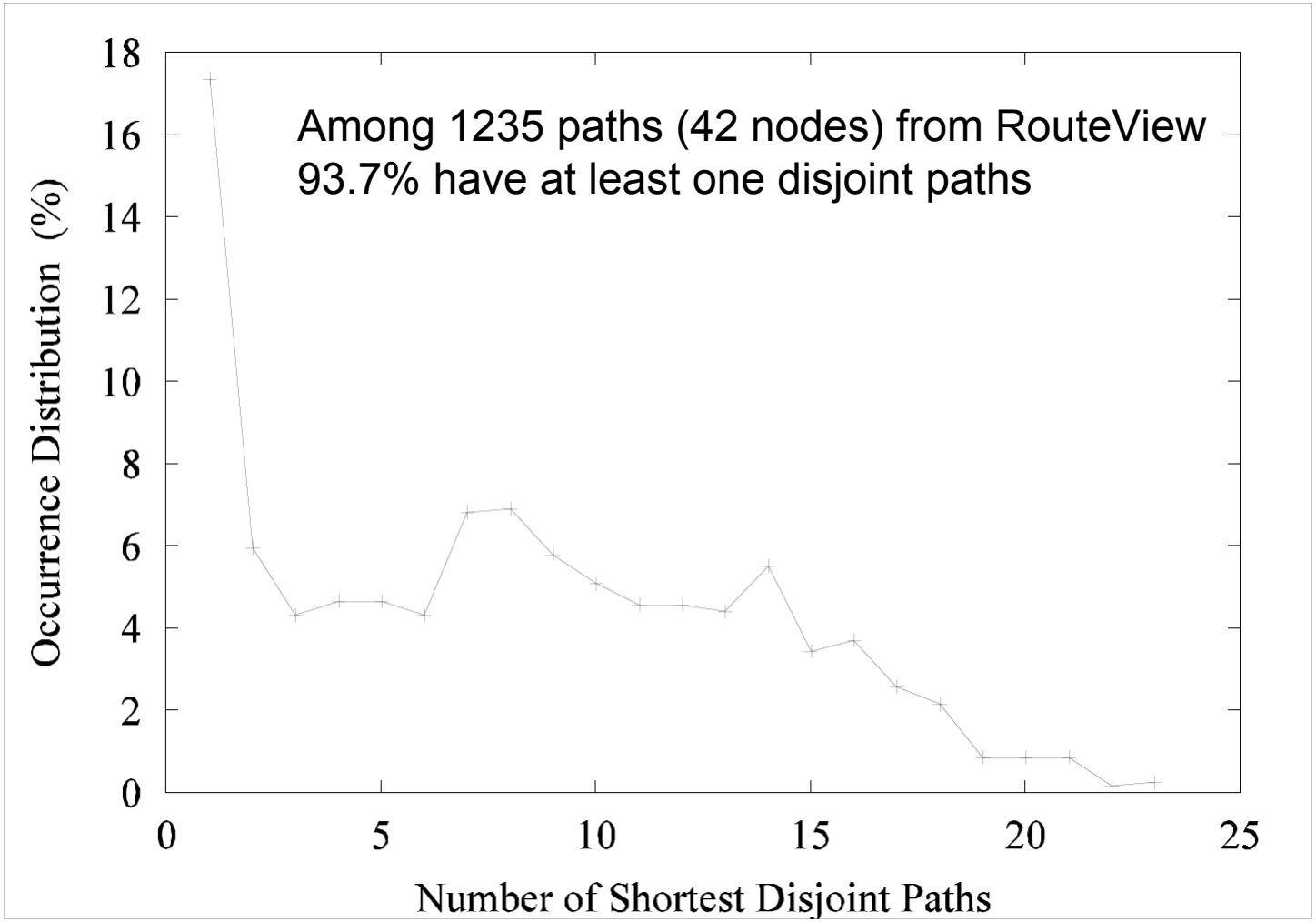
A single-hop indirection (*u, w, v*) that is edge-disjoint to (*u, v*) at the AS level

Disjoint Paths (cont)

How often we find disjoint paths using a single hop indirection?

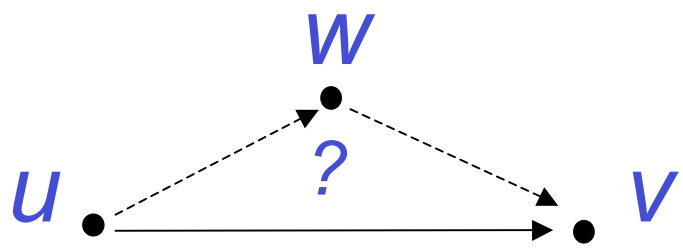
- Examined 1235 paths between multi-homed ASes (from RouteViews ; 42 vantage points)
- 93.7% have at least one disjoint path

Disjoint Paths (cont)



Disjoint Paths (cont)

Local node *u* *is* looking for alternate paths to *v* …



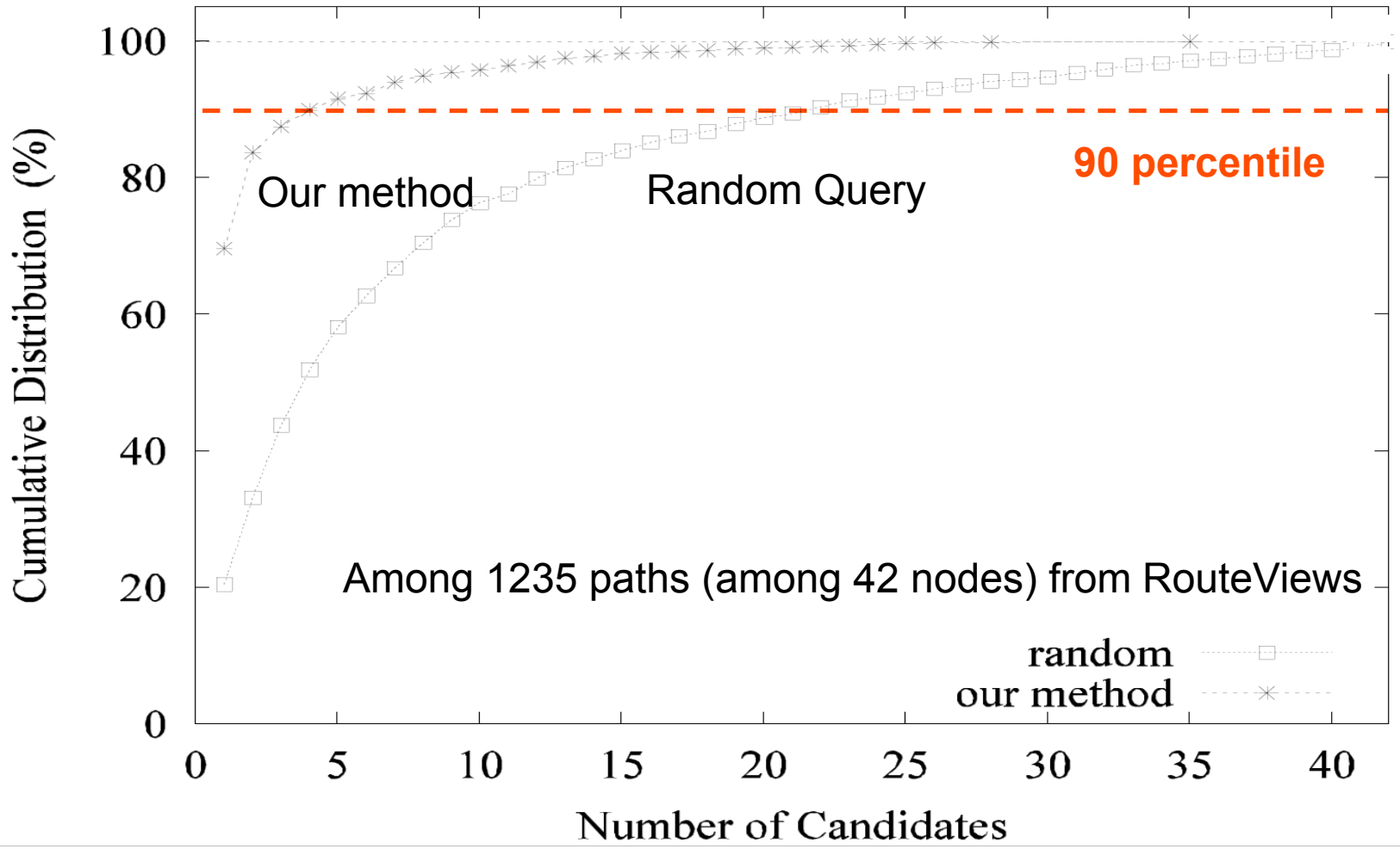
(1) Guess *w* 's likely to produce disjoint paths

- GetPath(*u,v*) and GetGraph

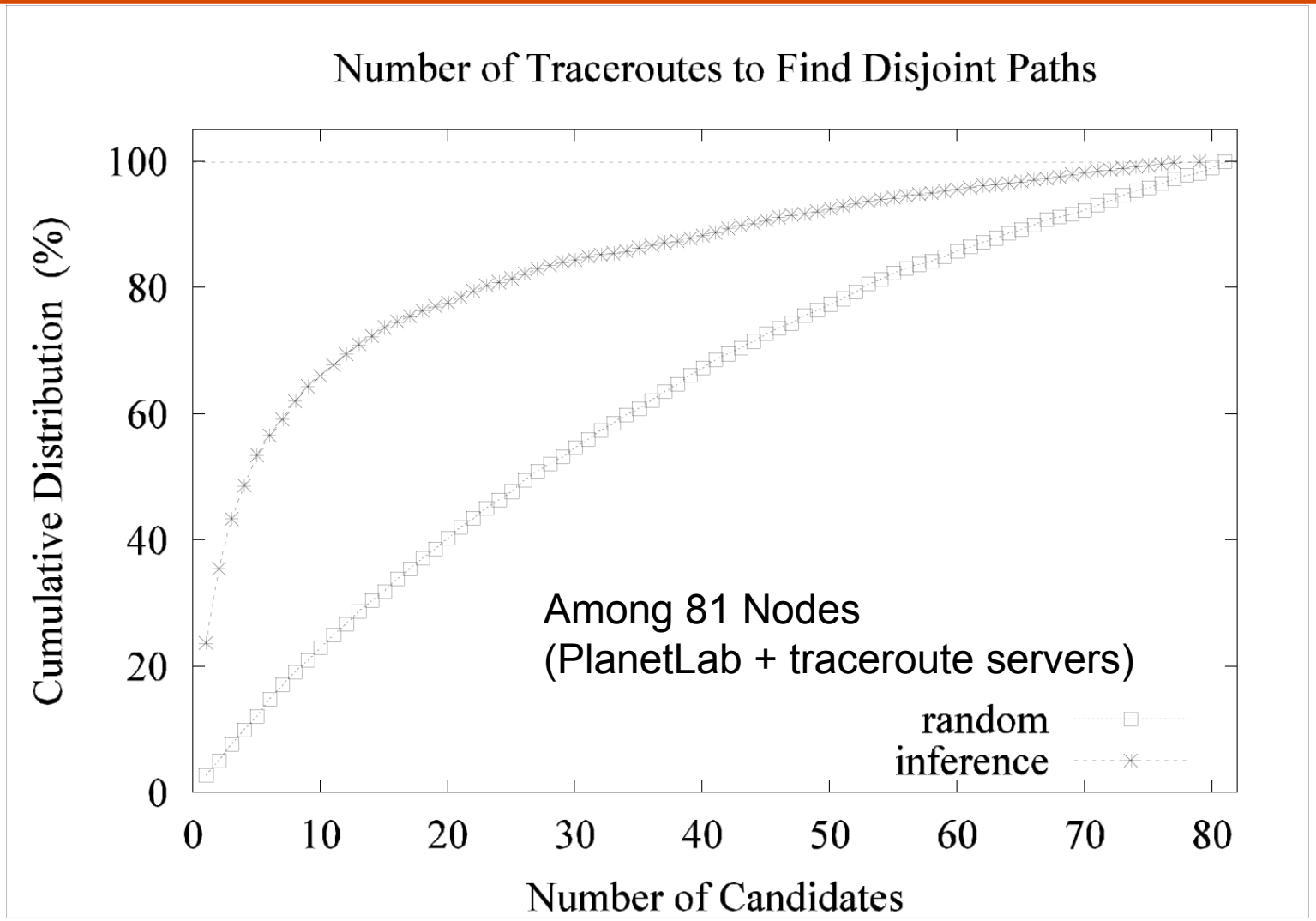
(2) Verify path (*u,w,v*) is disjoint to path (*u,v*)

- GetPath(*u,w*) and GetPath(*w,v*)

Disjoint Paths (cont)



Disjoint Paths (cont)

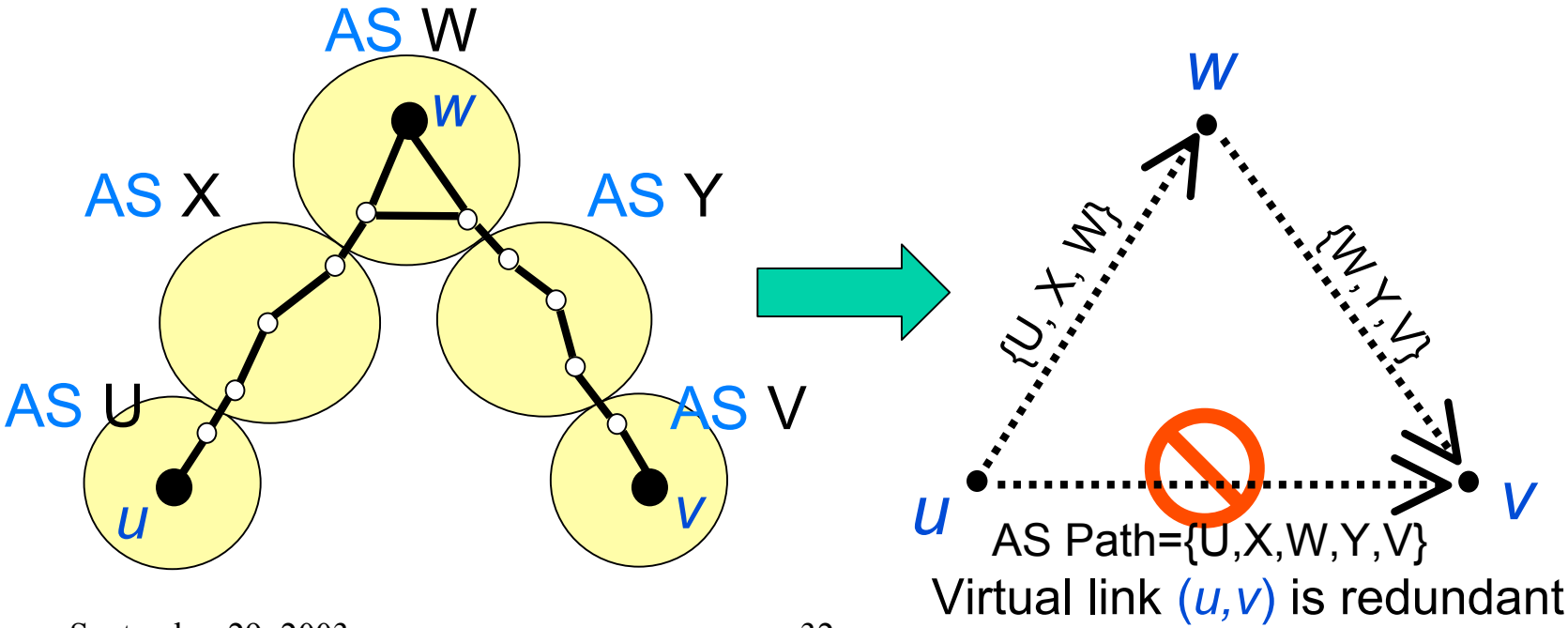


Representative Mesh

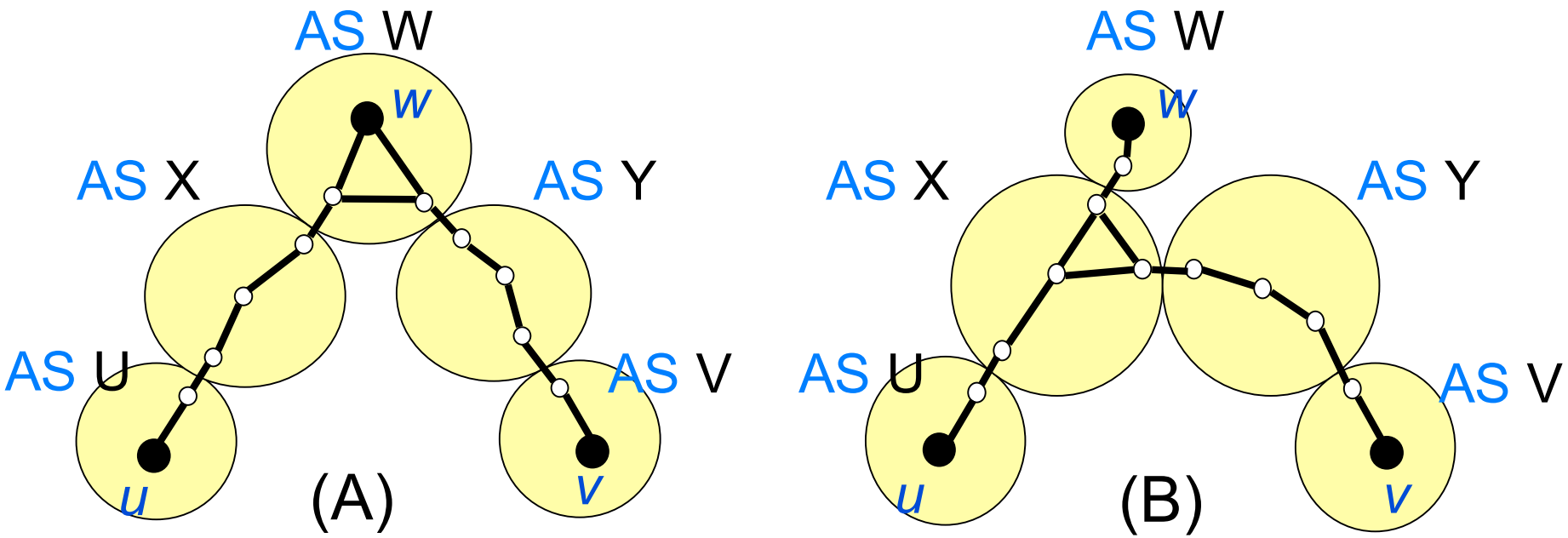
Mesh = BuildMesh(*N*)

N : a set of overlay nodes

Mesh : a graph that retains only independent edges in the underlying network



Mesh (cont)

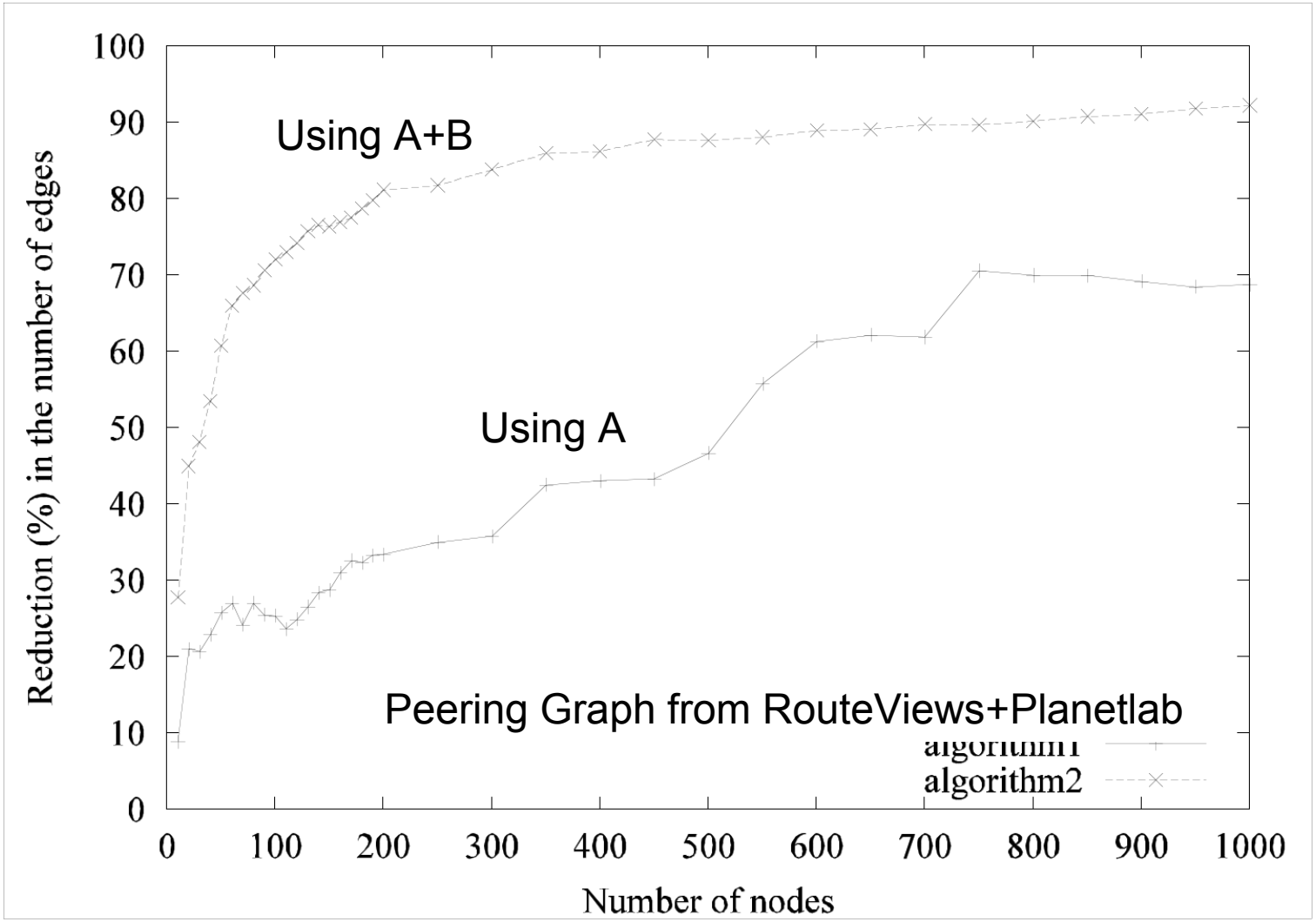


Local node *u* checks virtual links to the other *v's*

(1) Guess *w* 's likely to conform to either topology
- GetPath(*u,v*) and GetGraph

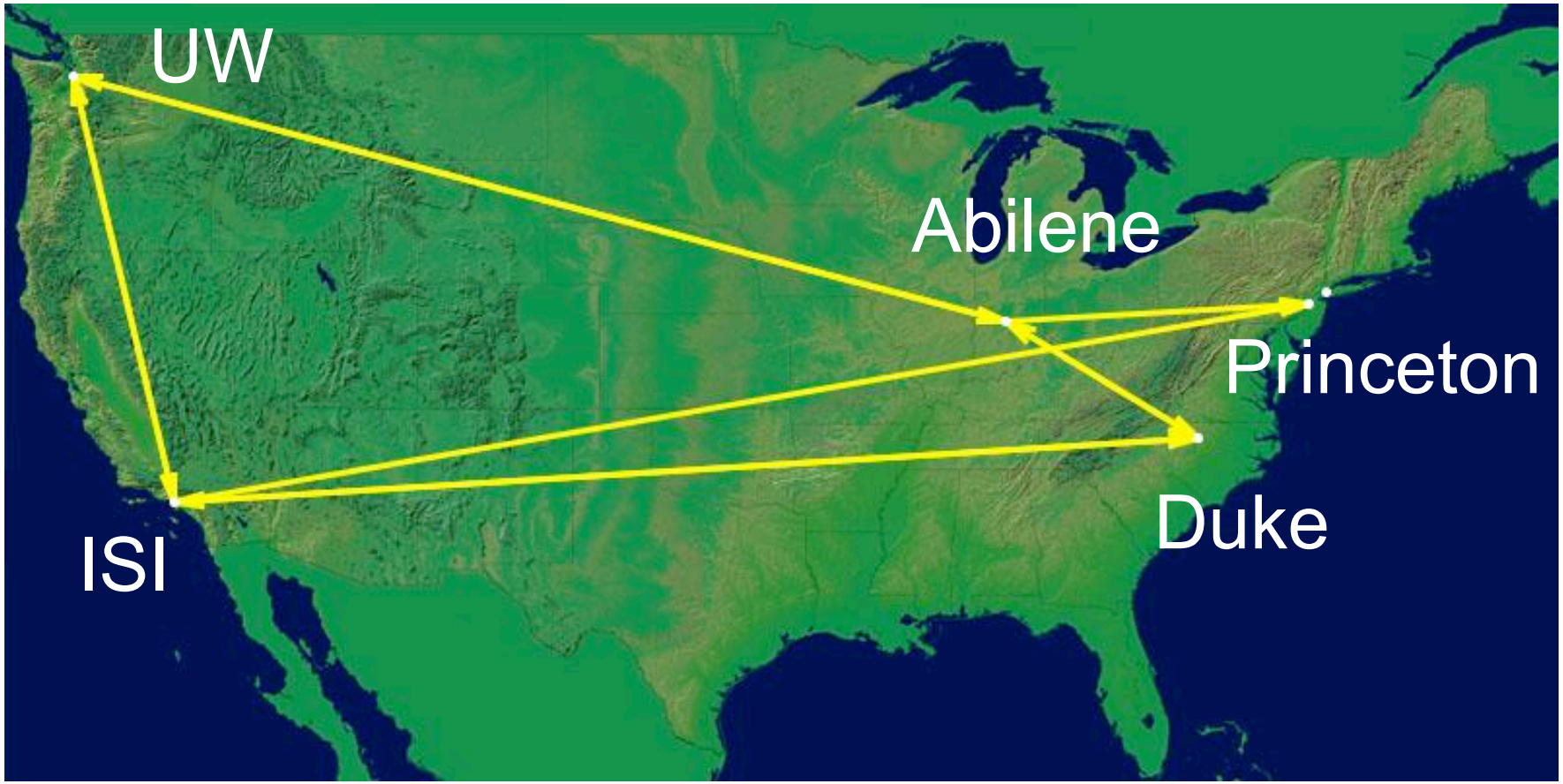
(2) Verify the topology
- GetPath(*u,w*) and GetPath(*w,v*)

Mesh (cont)


Using A+B
Using A
Peering Graph from RouteViews+Planetlab
algorithm1
algorithm2
Reduction (%) in the number of edges
Number of nodes
0
10
20
30
40
50
60
70
80
90
100
0
100
200
300
400
500
600
700
800
900
1000

BuildMesh on PlanetLab

Mesh with *N*=5 nodes (PU, Duke, ISI, UW, Abilene)



Todo

- Modify RON, ESM, DHT to use underlay
- Discriminate among nodes in transit ASes
- Build sparser mesh(es) on top of AS-based mesh
- Develop better cost/benefit model
- Get more BGP feeds (or fake it)
- Implement finer-grain resolutions
- Implement an “ IPv*N* ” service