Improving Client Web Availability with MONET

David G. Andersen, CMU

Hari Balakrishnan, M. Frans Kaashoek, Rohit Rao, MIT

`http://nms.csail.mit.edu/ron/ronweb/`

Availability We Want

- Carrier Airlines (2002 FAA Fact Book)
  - 41 accidents, 6.7M departures
  - ✔ 99.9993% availability
- 911 Phone service (1993 NRIC report +)
  - 29 minutes per year per line
  - ✔ 99.994% availability
- Std. Phone service (various sources)
  - 53+ minutes per line per year
  - ✔ 99.99+% availability

The Internet Has Only Two Nines

✘ End-to-End Internet Availability: 95% - 99.6%
[Paxson, Dahlin, Labovitz, Andersen]

Insufficient substrate for:

- New / critical apps:
  - Medical collaboration
  - Financial transactions
  - Telephony, real-time services, ...
- Users leave if page slower than 4-8 seconds
[Forrester Research, Zona Research]

MONET: Goals

- Mask Internet failures
  - Total outages
  - Extended high loss periods
- Reduce exceptional delays
  - Look like failures to user
  - Save seconds, not milliseconds

MONET achieves 99.9 - 99.99% availability

(Not enough, but a good step!)

Windows

A fatal exception 0E has occurred at 0028:C00068F8 in PPT.EXE<01> + 000059F8. The current application will be terminated.

* Press any key to terminate the application.
* Press CTRL+ALT+DEL to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue

Windows

A fatal exception 0E has occurred at 0028:C00068F8 in PPT.EXE<01> + 000059F8. The current application will be terminated.

* Press any key to terminate the application.
* Press CTRL+ALT+DEL to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue

Not about client failures...

```
                              Windows

A fatal exception 0E has occurred at 0028:C00068F8 in PPT.EXE<01> +
000059F8.  The current application will be terminated.

* Press any key to terminate the application.
* Press CTRL+ALT+DEL to restart your computer.  You will
  lose any unsaved information in all applications.

                    Press any key to continue
```

Not about client failures...

Nor *fixing* server failures (but understand)

There's another nine hidden in here, but today...

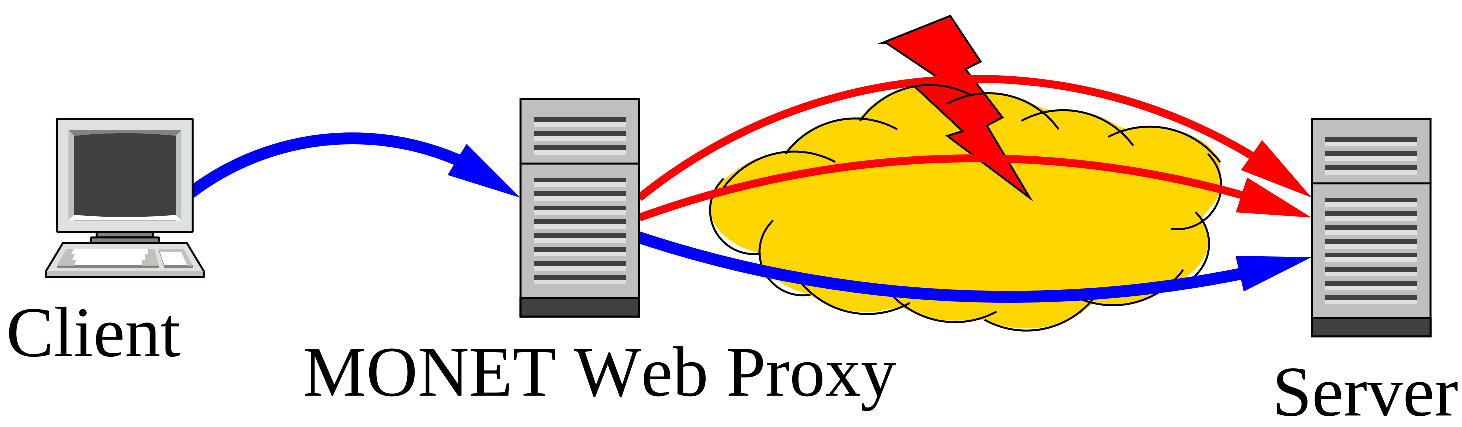
"It's about the network!"

End-to-End Availability: Challenges

- Internet services depend on many components: Access networks, routing, DNS, servers, ...
- End-to-end failures persist despite availability mechanisms for each component.
- Failures unannounced, unpredictable, silent
- Many different causes of failures:
  - Misconfiguration, deliberate attacks, hardware/software failures, persistent congestion, routing convergence

Our Approach

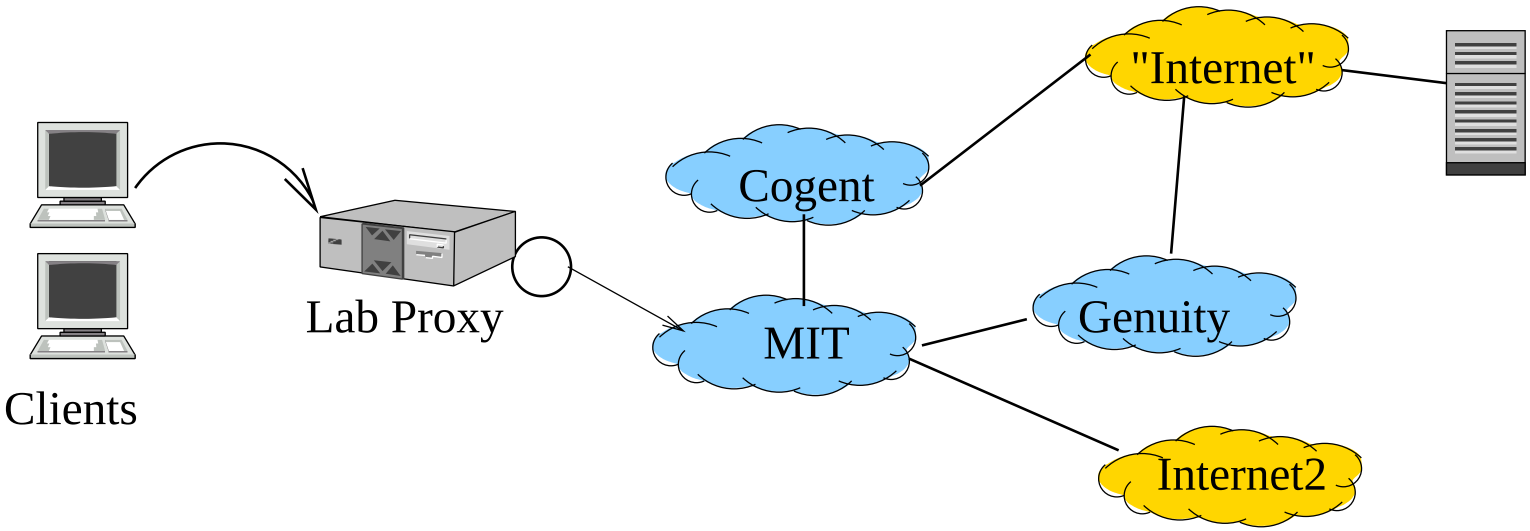
- Expose multiple paths to end system
  - How to get access to them?
- End-systems determine if path works via probing/measurement
  - How to do this probing?
- Let host choose a good end-to-end path



Contributions

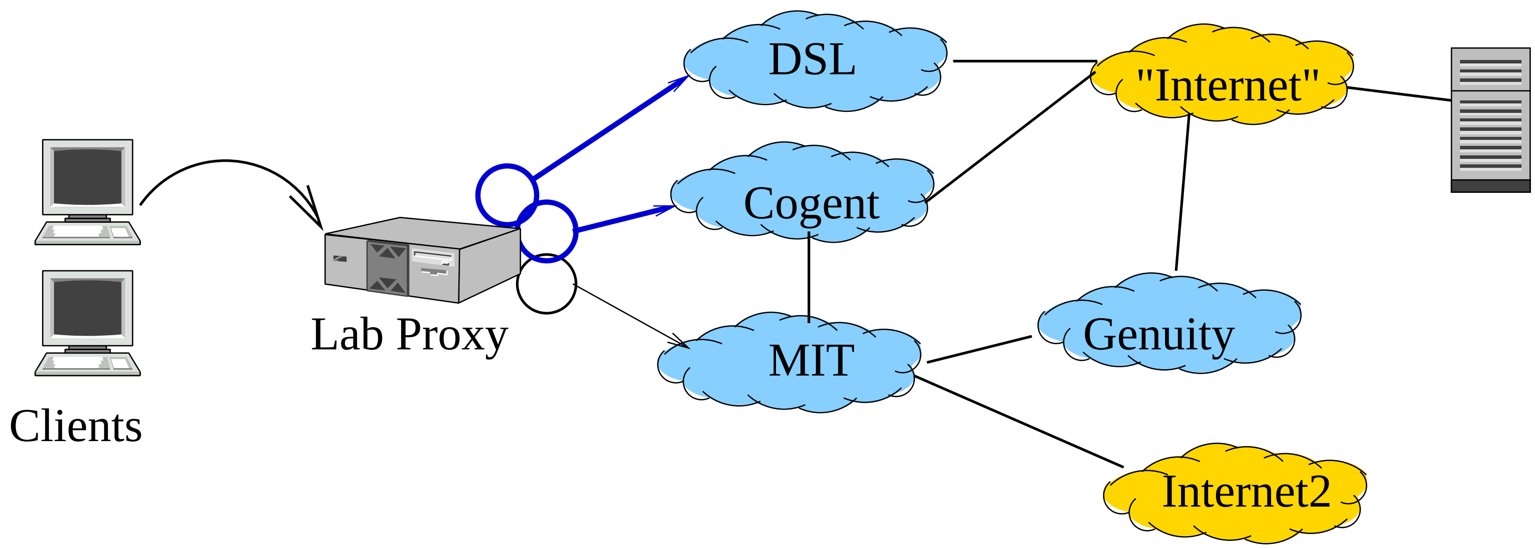
- MONET Web Proxy design and implementation
- Waypoint Selection algorithm explores paths with low overhead
- Evaluation of deployed system with live user traces; roughly order of magnitude availability improvement

MONET: Bypassing Web Failures



- A Web-proxy based system to improve availability
- Three ways to obtain paths

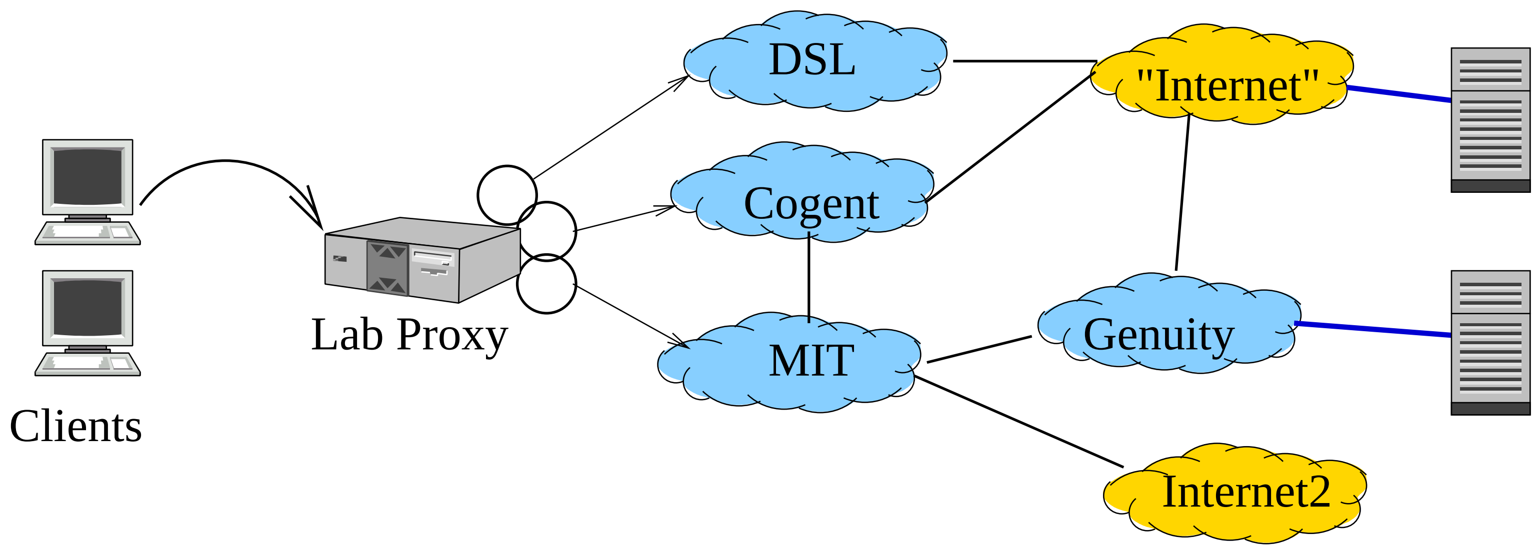
MONET: Obtaining Paths



- 10-50% of failures at client access link

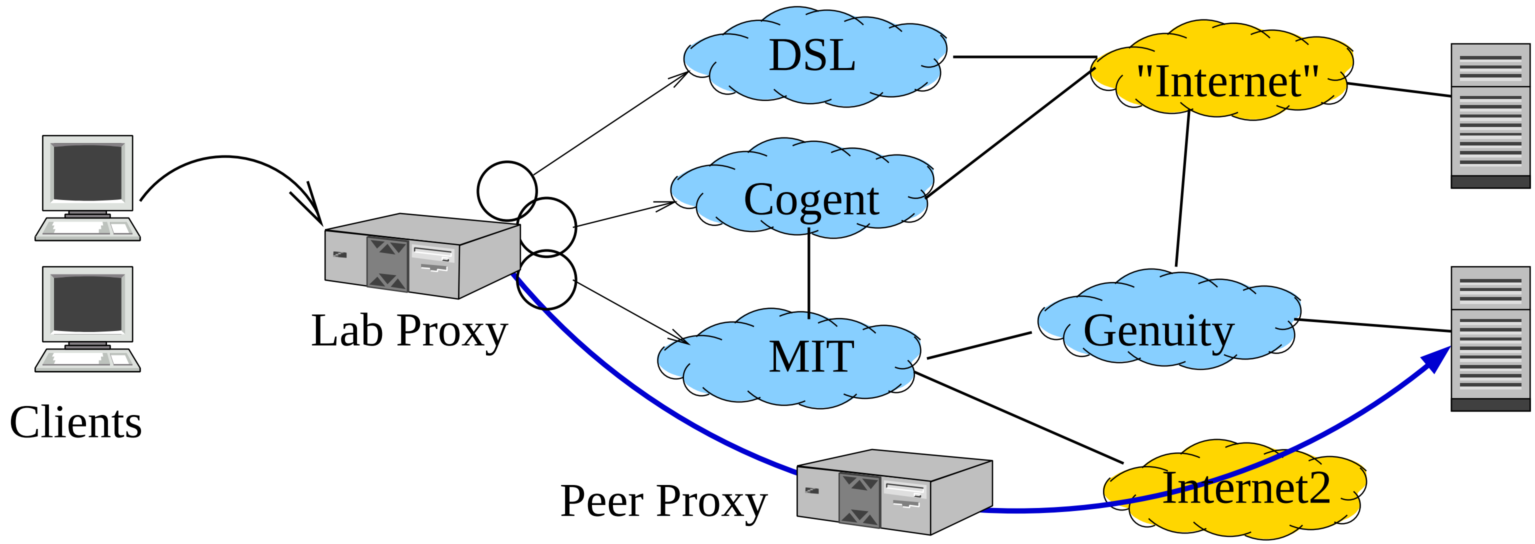
➜ Multihome the *proxy* (no routing needed)

MONET: Obtaining Paths



- 10-50% of failures at client access link
  - ➔ Multihome the *proxy* (no routing needed)
- Many failures at server access link
  - ➔ Contact multiple servers

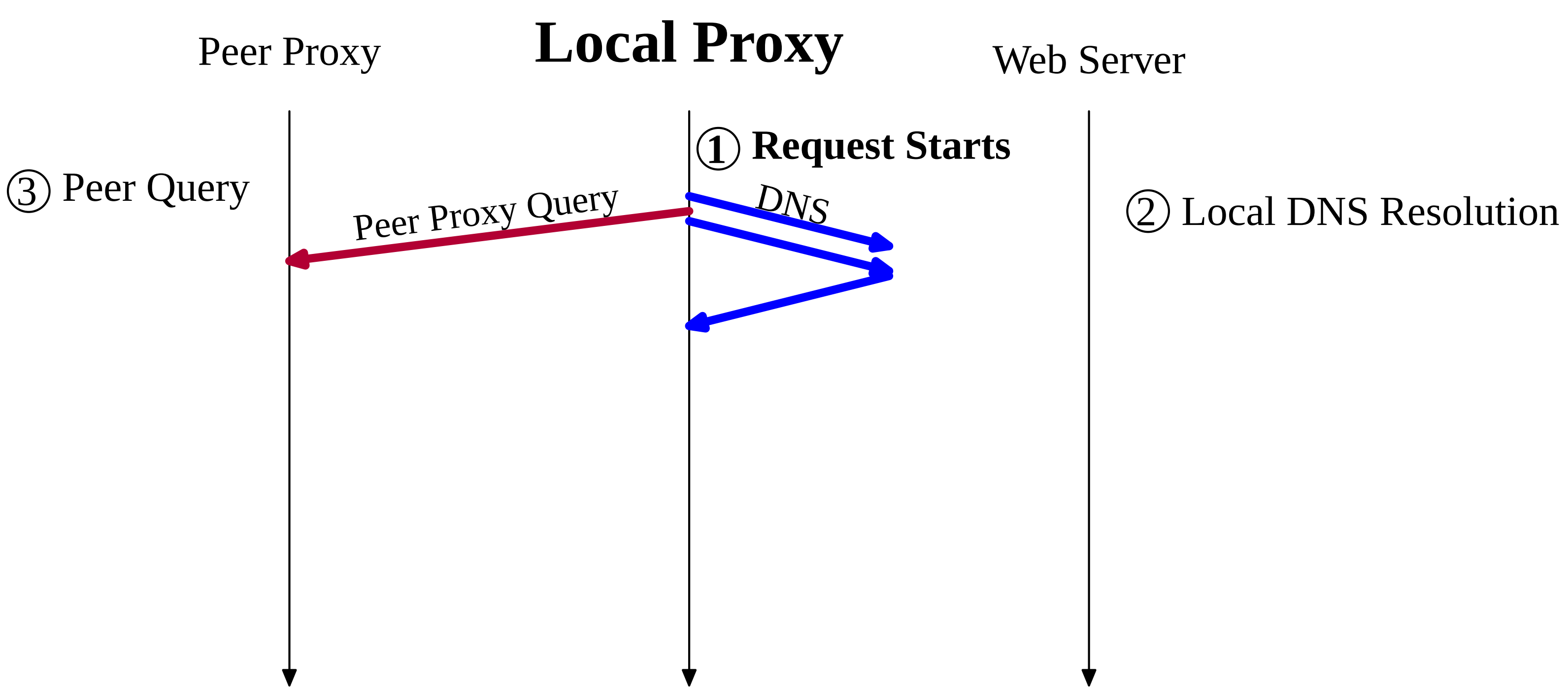
MONET: Obtaining Paths



- 10-50% of failures at client access link
  → Multihome the *proxy* (no routing needed)
- Many failures at server access link
  → Contact multiple servers
- 40-60% failures "in network"→Overlay paths

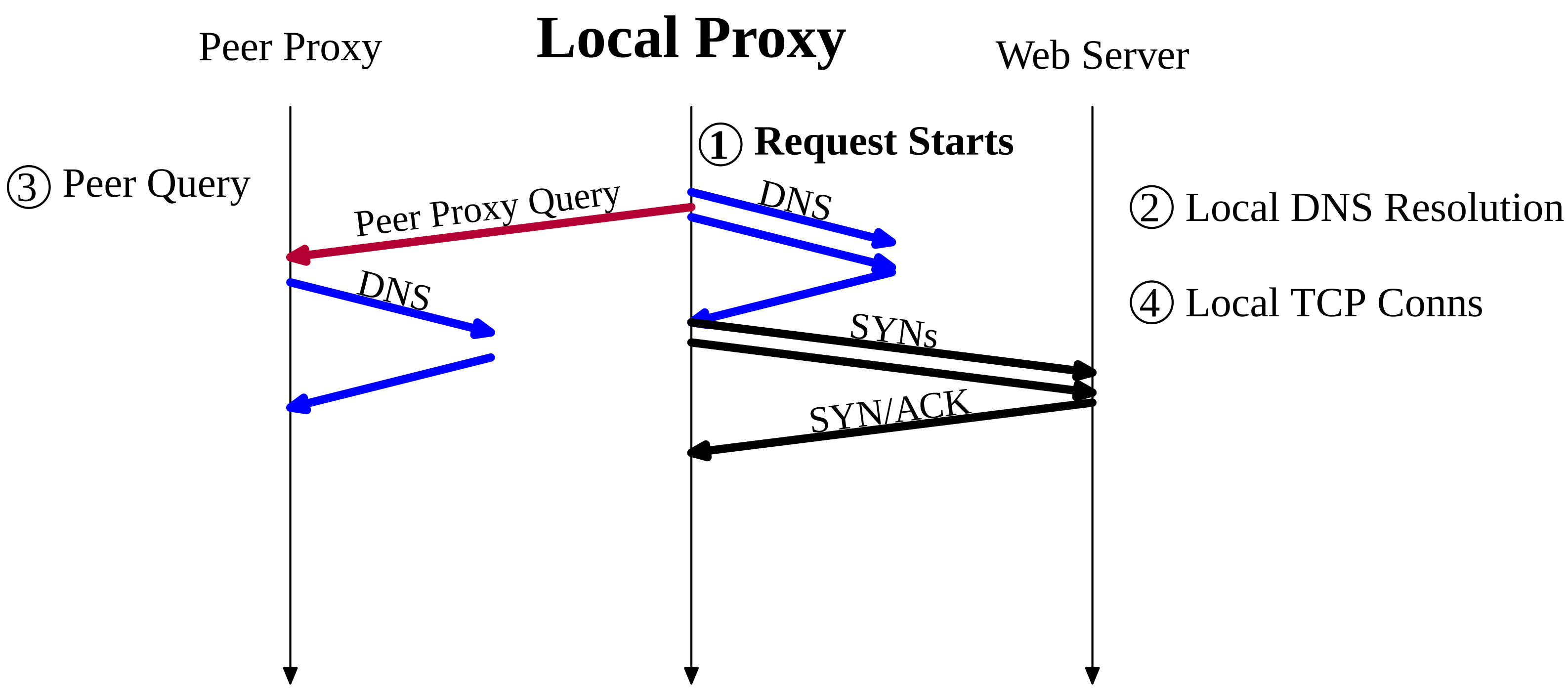
Parallel Connections Validate Paths

Near-concurrent TCP, peer proxy, and DNS queries.



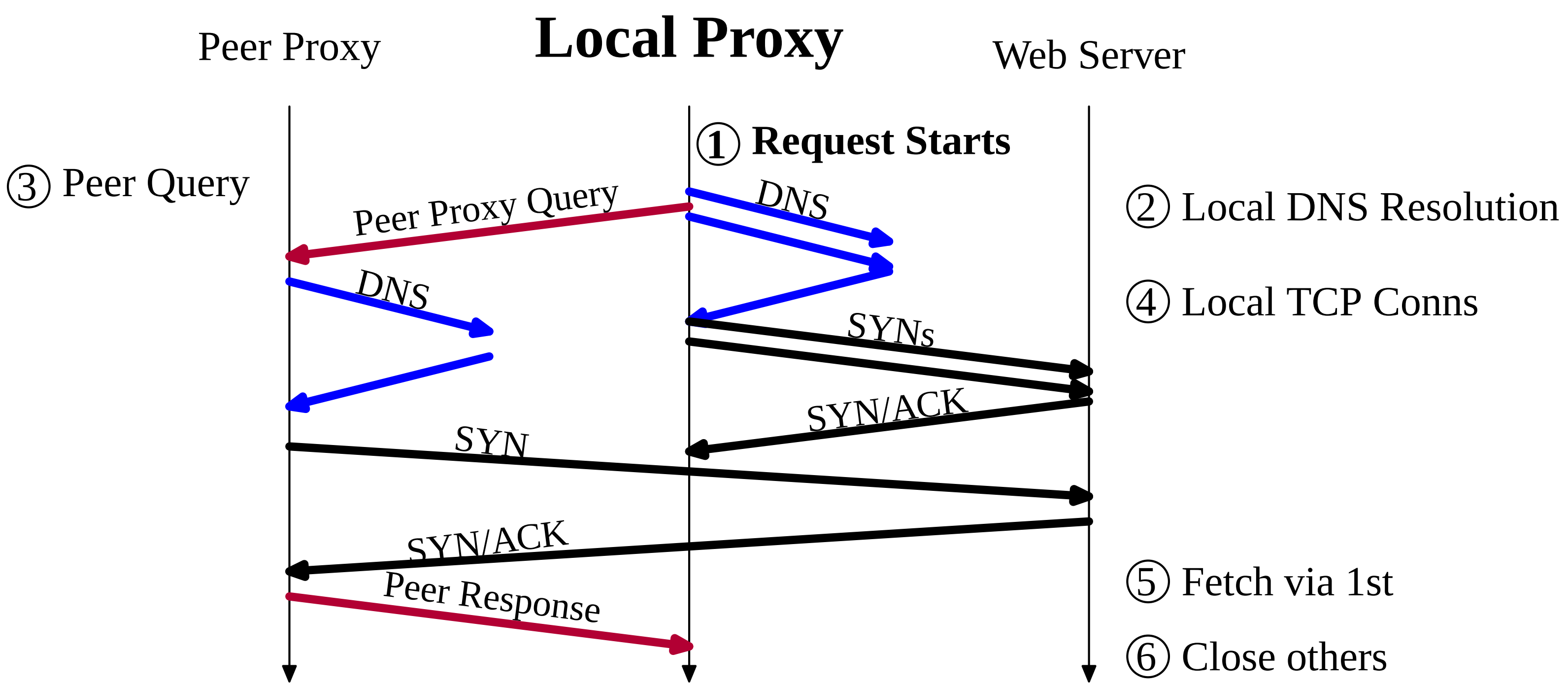
Parallel Connections Validate Paths

Near-concurrent TCP, peer proxy, and DNS queries.



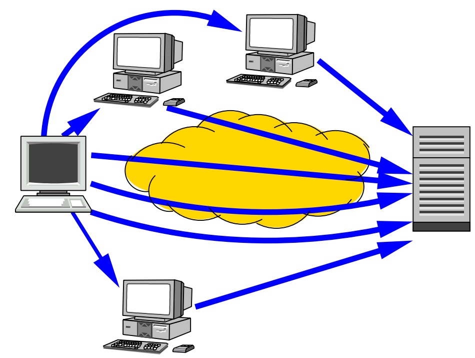
Parallel Connections Validate Paths

Near-concurrent TCP, peer proxy, and DNS queries.



A More Practical MONET

Evaluated MONET tries all combinations:



l local interfaces

p peers

s servers

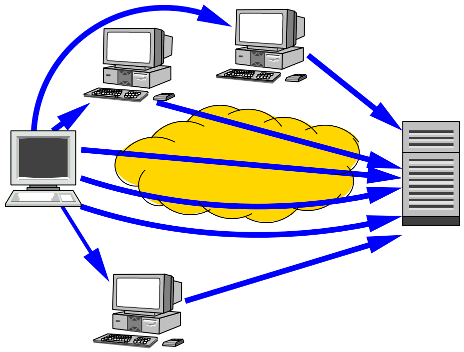
$ls + lps$ paths

$l = 3, p = 3, s = 1 - 8$

Paths = 12 – 96

A More Practical MONET

Evaluated MONET tries all combinations:



l local interfaces

p peers

s servers

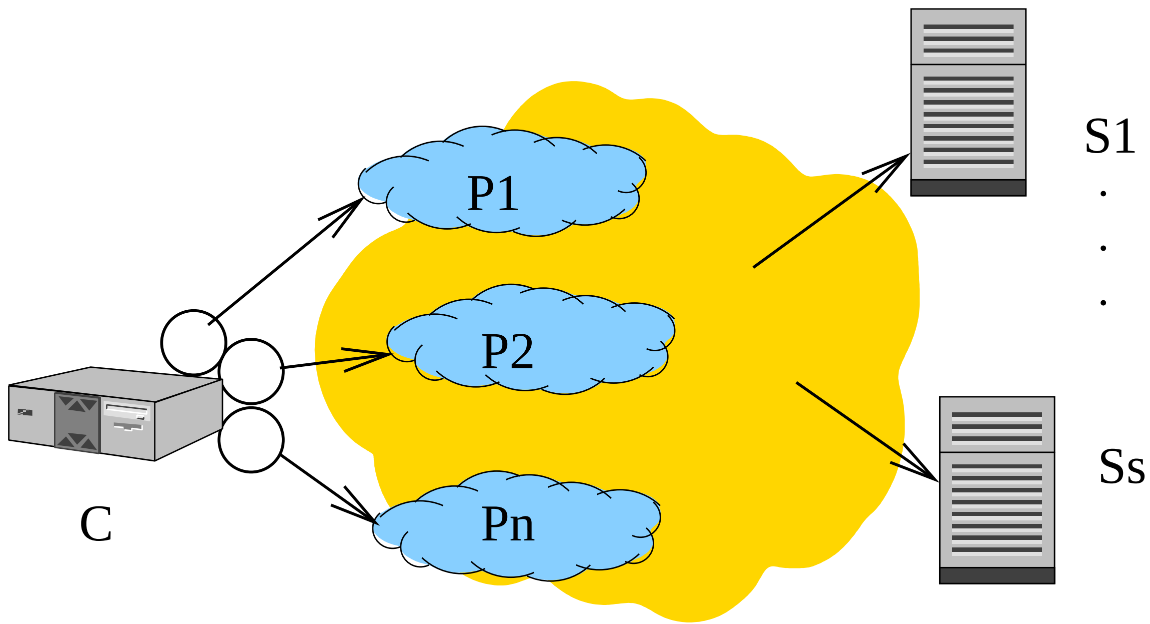
$ls + lps$ paths

$l = 3, p = 3, s = 1 - 8$

Paths = 12 – 96

- Waypoint Selection chooses the right subset
  - What order to try interfaces?
  - How long to wait between tries?

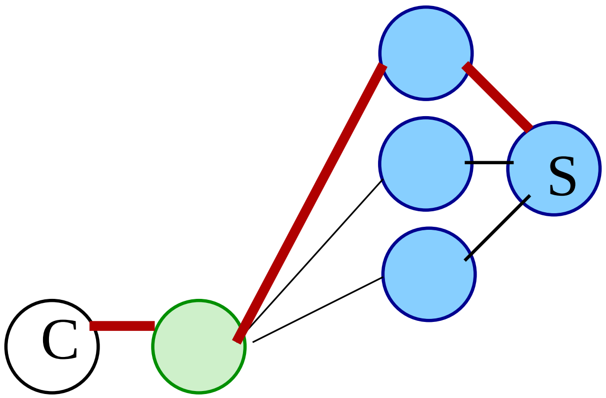
Waypoint Selection Problem



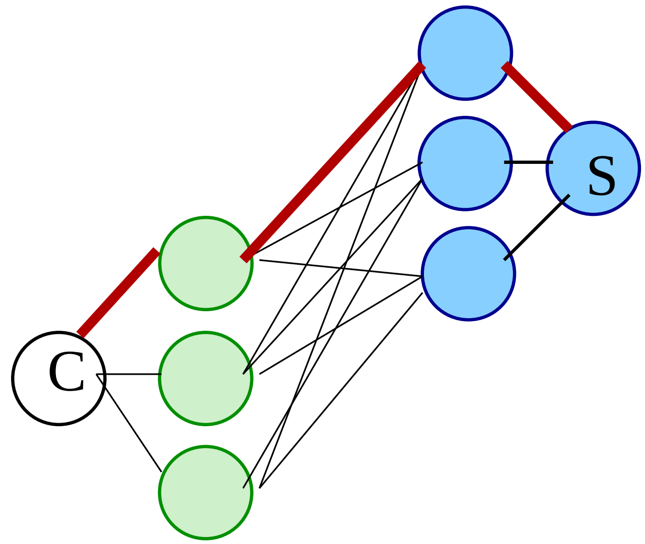
Client C Paths $P_1, \cdots, P_N$ Servers $S_1, ..., S_s$

- Find good order of the $s * N$ P_x, S_y pairs.
- Find delay between each pair.

Waypoint Selection

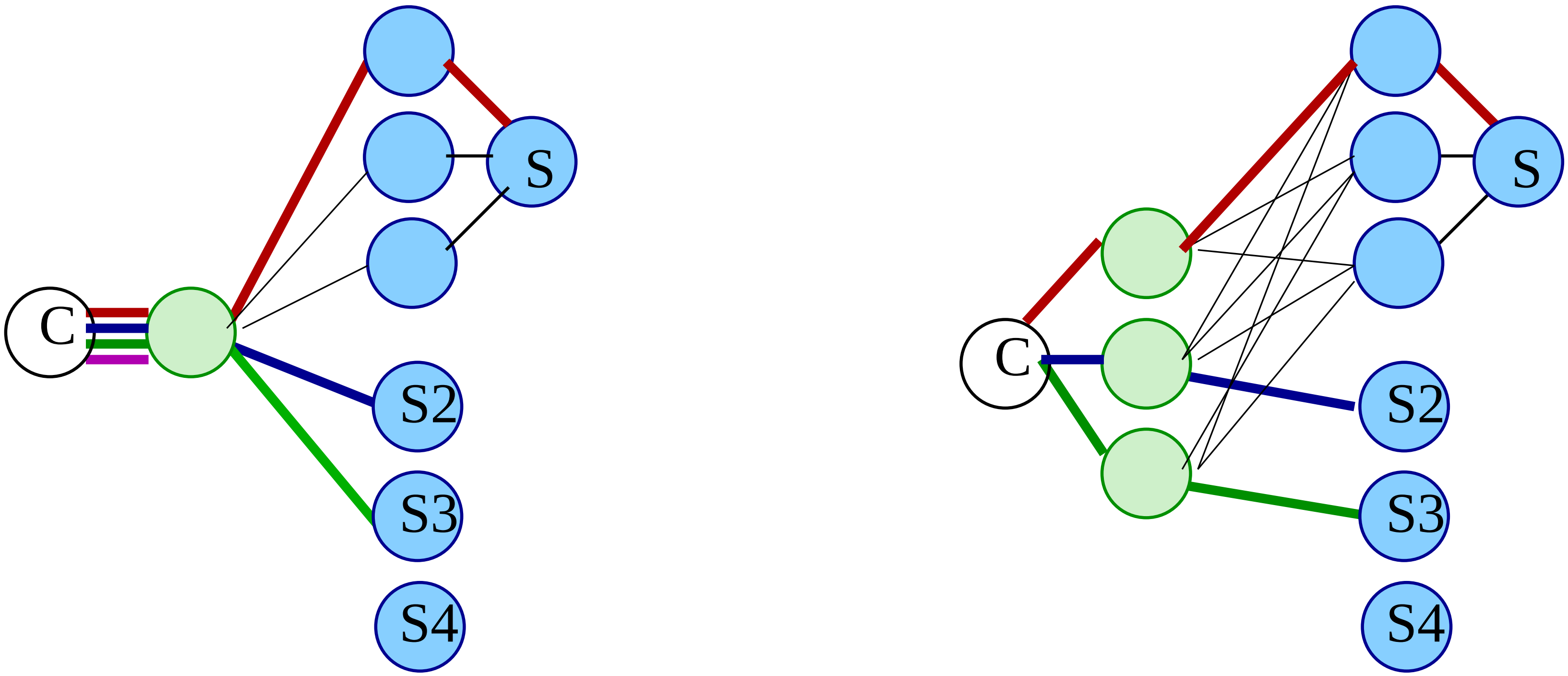


Server Selection



Waypoint Selection

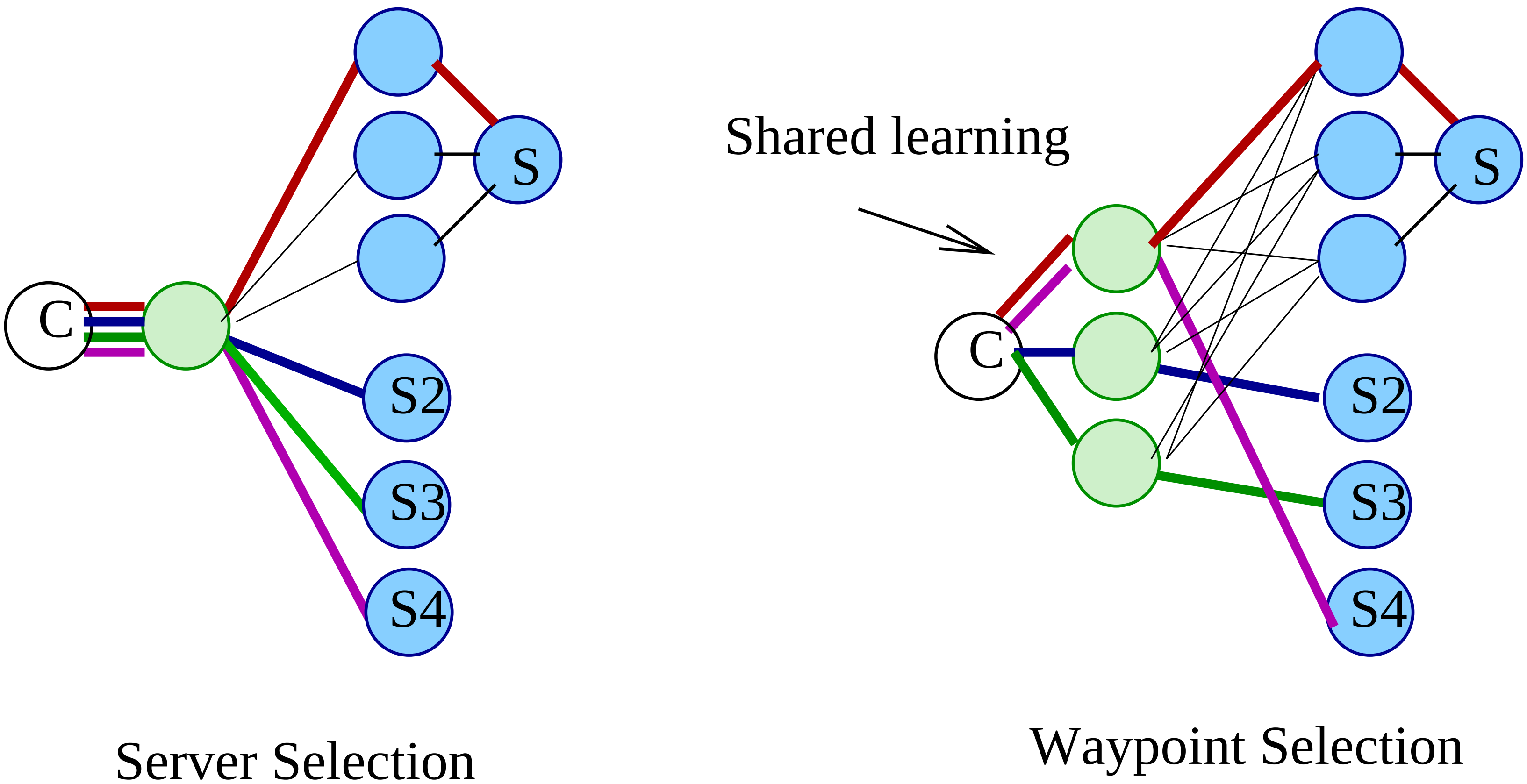
Waypoint Selection



Server Selection

Waypoint Selection

Waypoint Selection



- History teaches about *paths*, not just servers

➔ Better initial guess (ephemeral...)

Using Waypoint Results to Probe

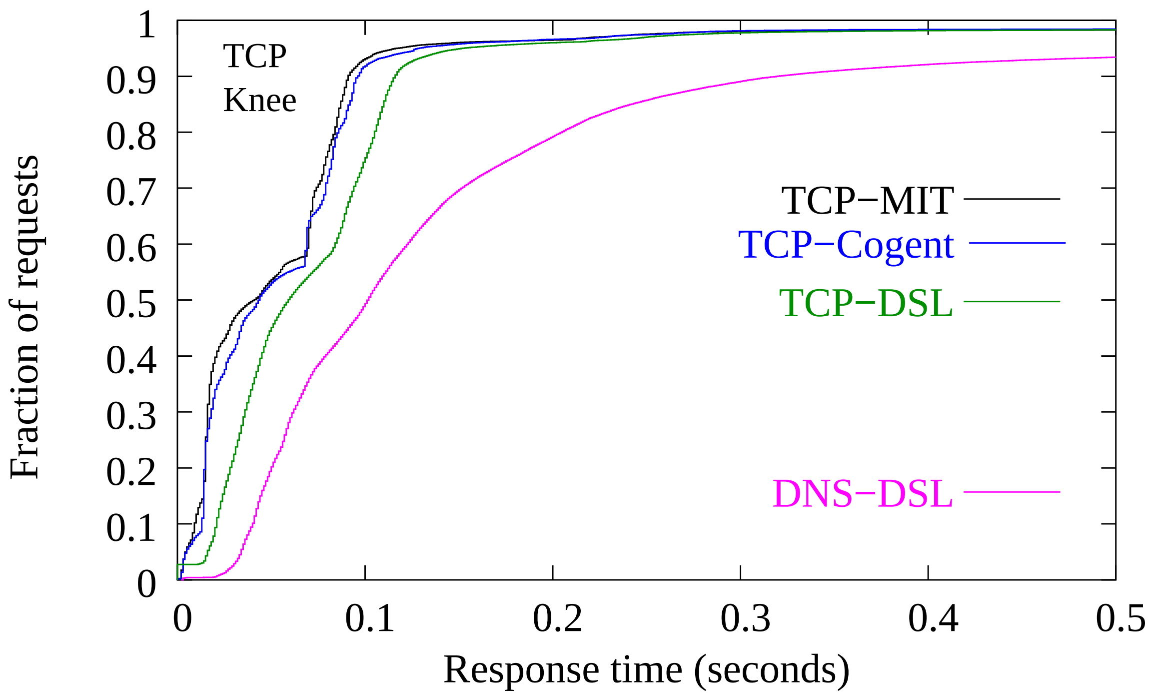
- DNS: Current best + random interface
- TCP: Current best path (int or peer)
- 2nd TCP w/5% chance via random path
- Pass results back to waypoint algorithm

Using Waypoint Results to Probe

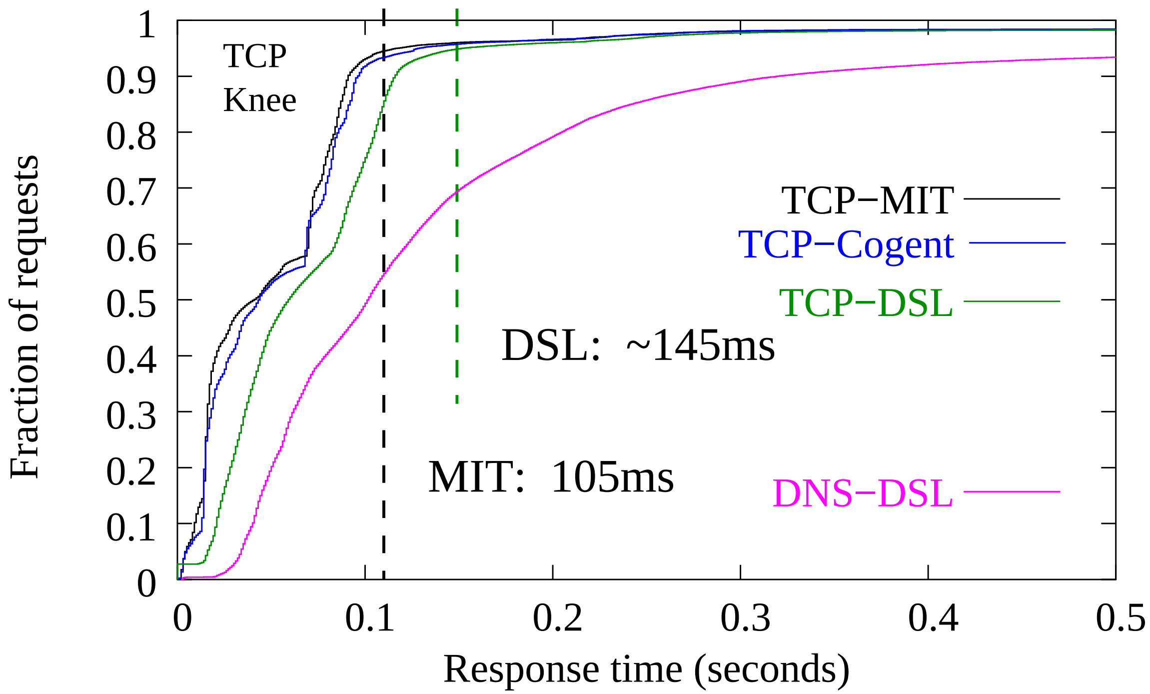
- DNS: Current best + random interface
- TCP: Current best path (int or peer)
- 2nd TCP w/5% chance via random path
- Pass results back to waypoint algorithm
- While no response within *thresh*
  - connect via next best
  - increase *thresh*

➔What information affects *thresh*?

TCP Response Time Knee



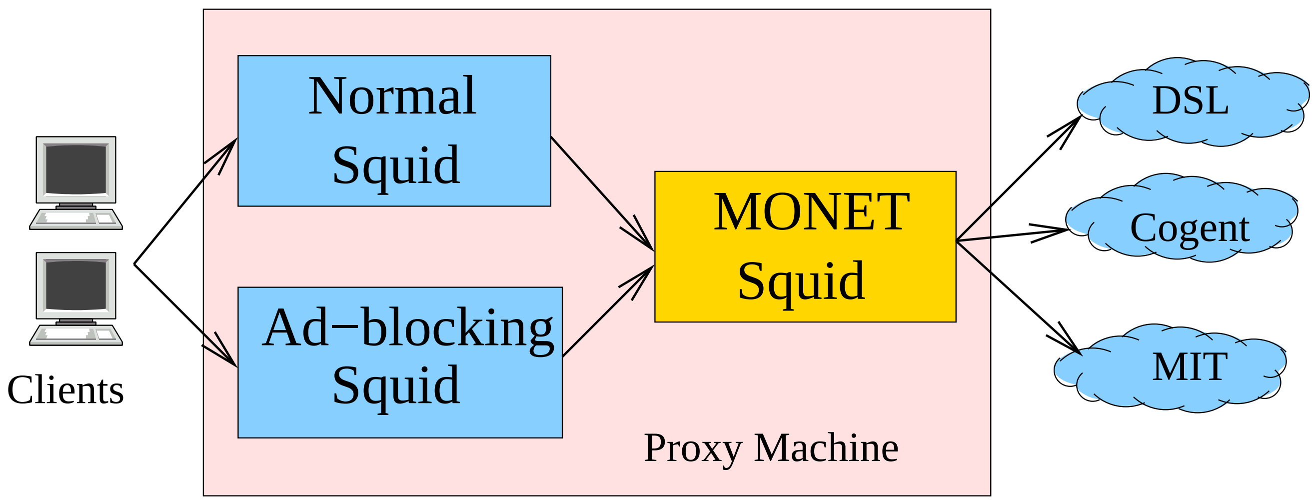
TCP Response Time Knee



- When to probe - right after knee
- Small extra latency → much less overhead

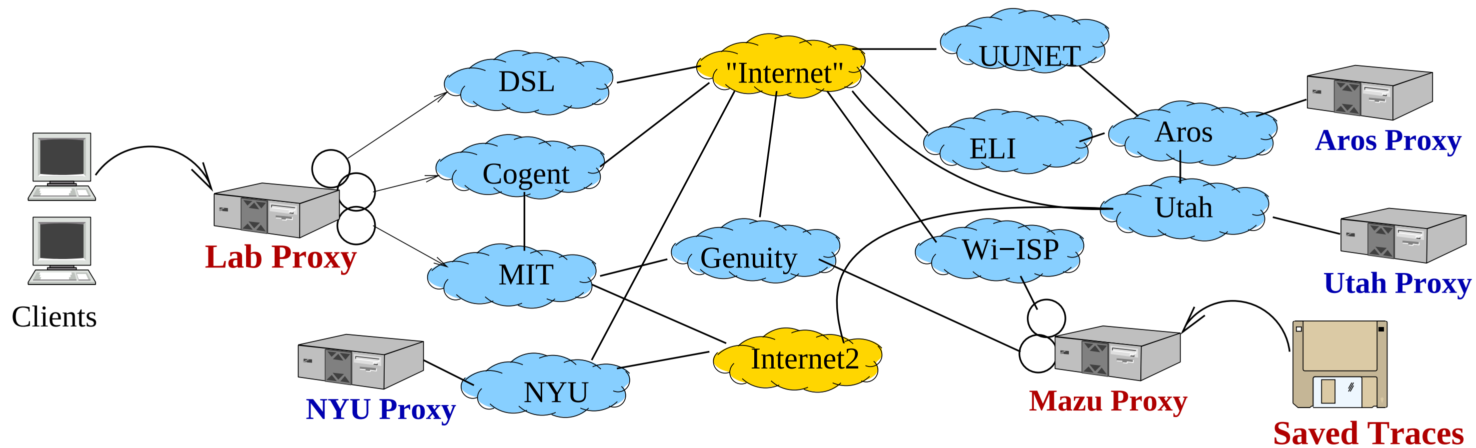
Two ways to approximate the knee in the paper

Implementation



- Squid Web proxy + parallel DNS resolver
- Front-end squids mask back-end failures (Ad-blocking squid as bribe)
- Choose outbound link with FreeBSD / Mac OS X `ipfw` or Linux policy routing

6-site MONET Deployment



- Two years, ~ 50 users/week
- Primary traces at MIT, replay at Mazu
- Three peer proxies: NYU, Utah, Aros
- Focus on 1 Dec 2003 – 27 Jan 2004
- Record everything

Measurement Challenges

- Invalid DNS responses (packet traces)
- Invalid IPs (0.0.0.0, 127.0.0.1, ...)
- Anomalous servers - discard 90% SYNs, etc.
- Implementation and design flaws
  - Network anomalies hit corner cases (Must avoid correlated measurement & network failures!)
- Identify, automate detection, iterate...

Excluded consistently anomalous services.

MIT Trace Statistics

Request type	Count
Client object fetch	2.1M
Cache misses	1.3M
Data fetch size	28.5 Gb
Cache hit size	1 Gb
TCP Connections	616,536
DNS lookups	82,957

137,341 Sessions - first req to a server after 60+ idle seconds (avoids bias)

Characterizing Failures

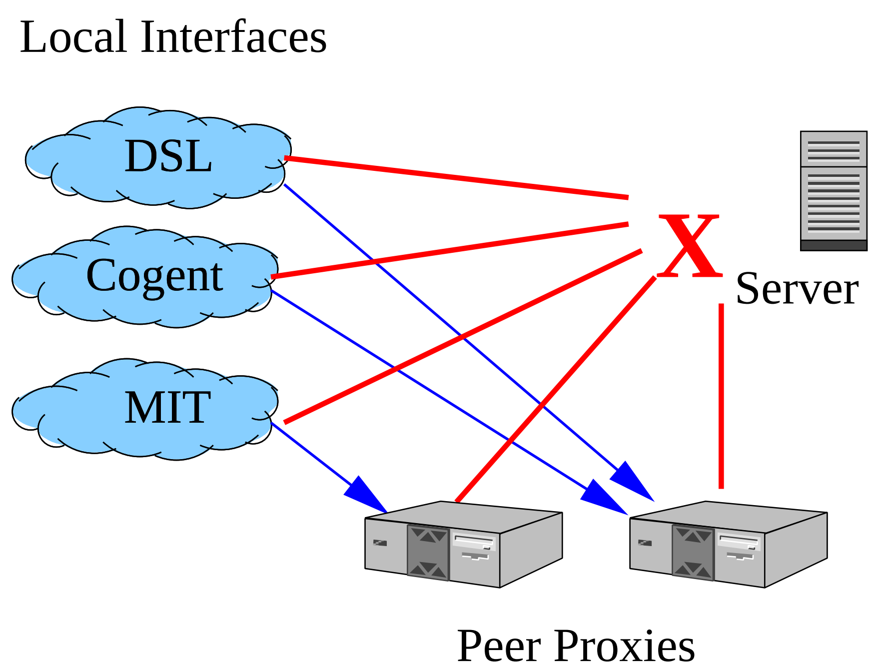
DNS

Server unreach

Server RST

Client access

Wide-area



2+ peers reachable

no peer or link could reach server

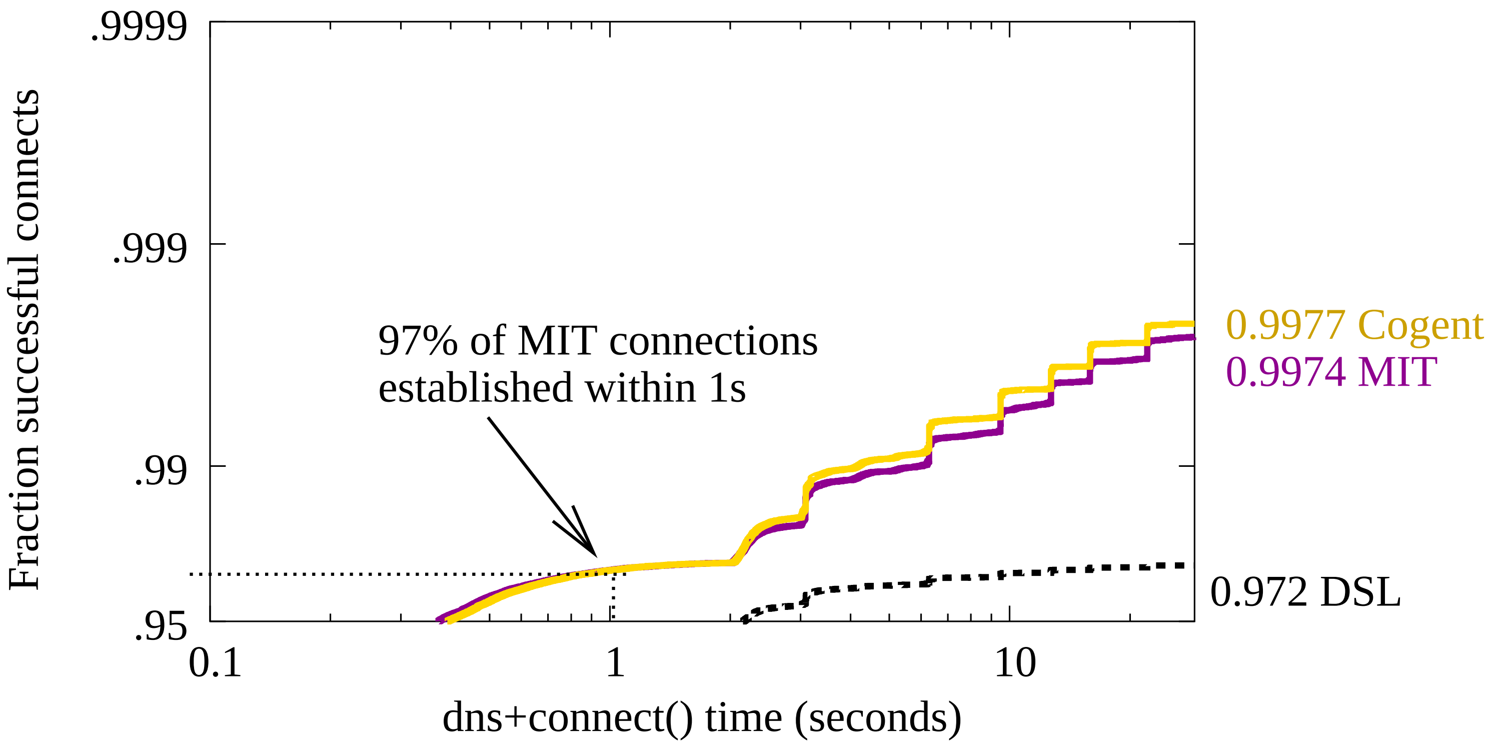
(40% unreachable during post-analysis)

Failure Breakdown

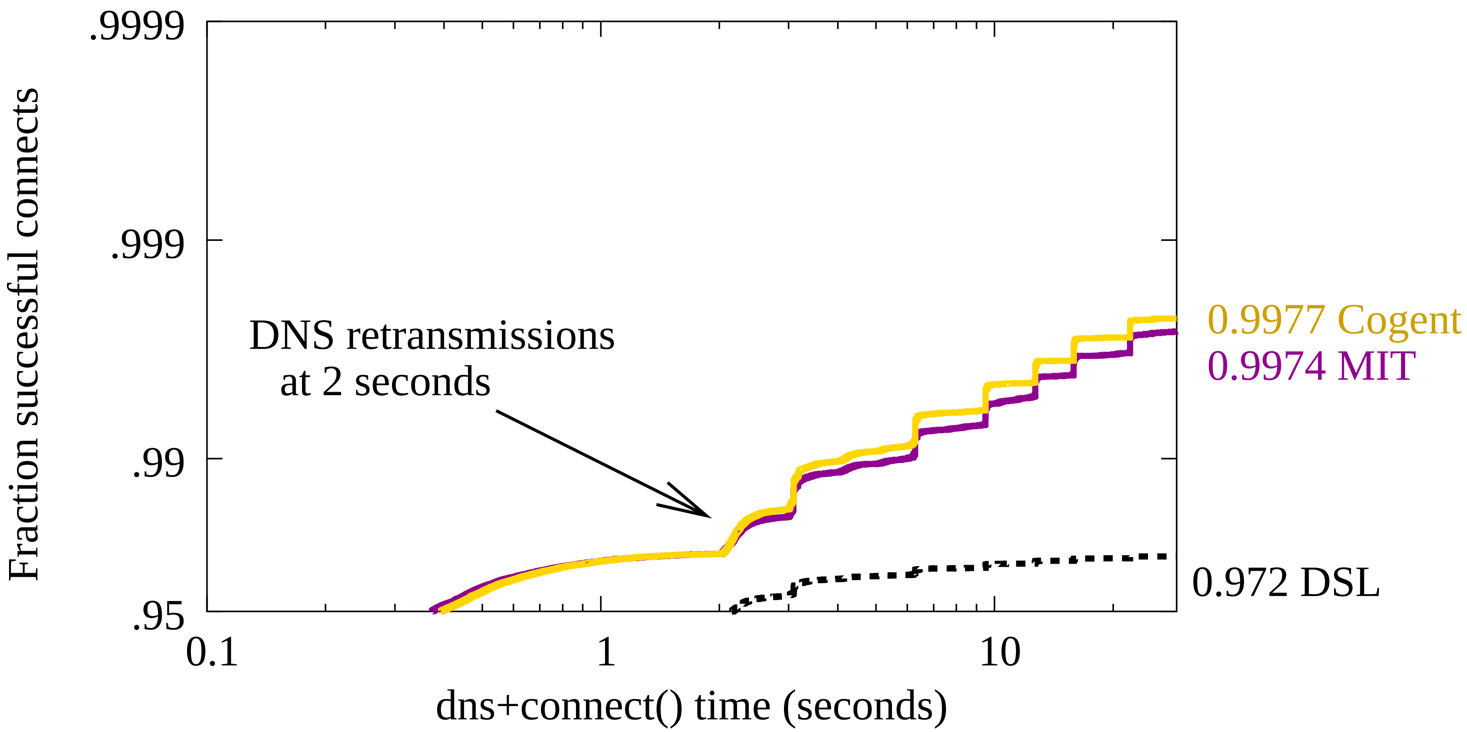
		MIT 137,612 sessions		
Failure Type	**Srv**	**MIT**	**Cog**	**DSL**
DNS	1			
Srv. Unreach	173			
Srv. RST	50			
Client Access		152	14	2016
Wide-area		201	238	1828
Availability		99.6%	99.7%	97%

Factor out server failures—until they use MONET!

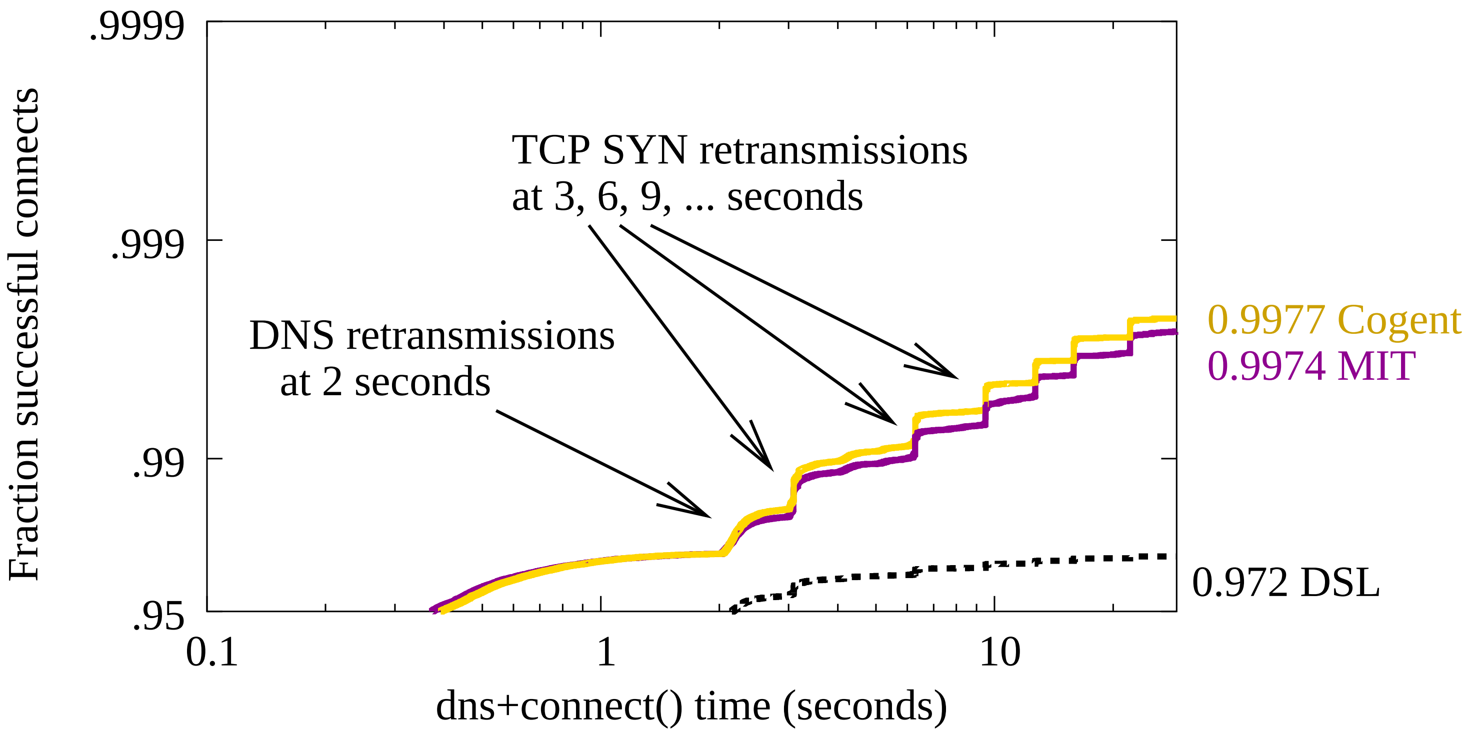
Single Link Availability



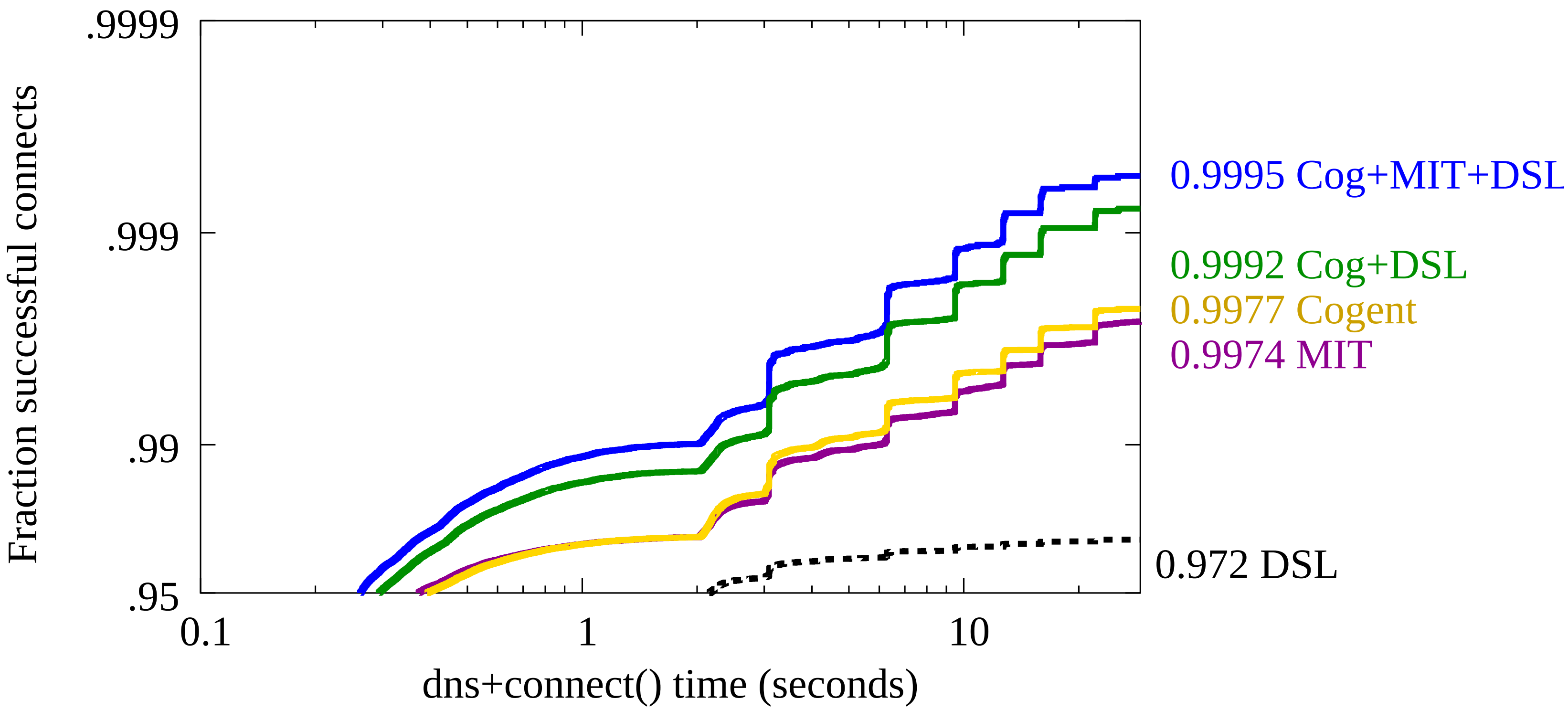
Single Link Availability



Single Link Availability

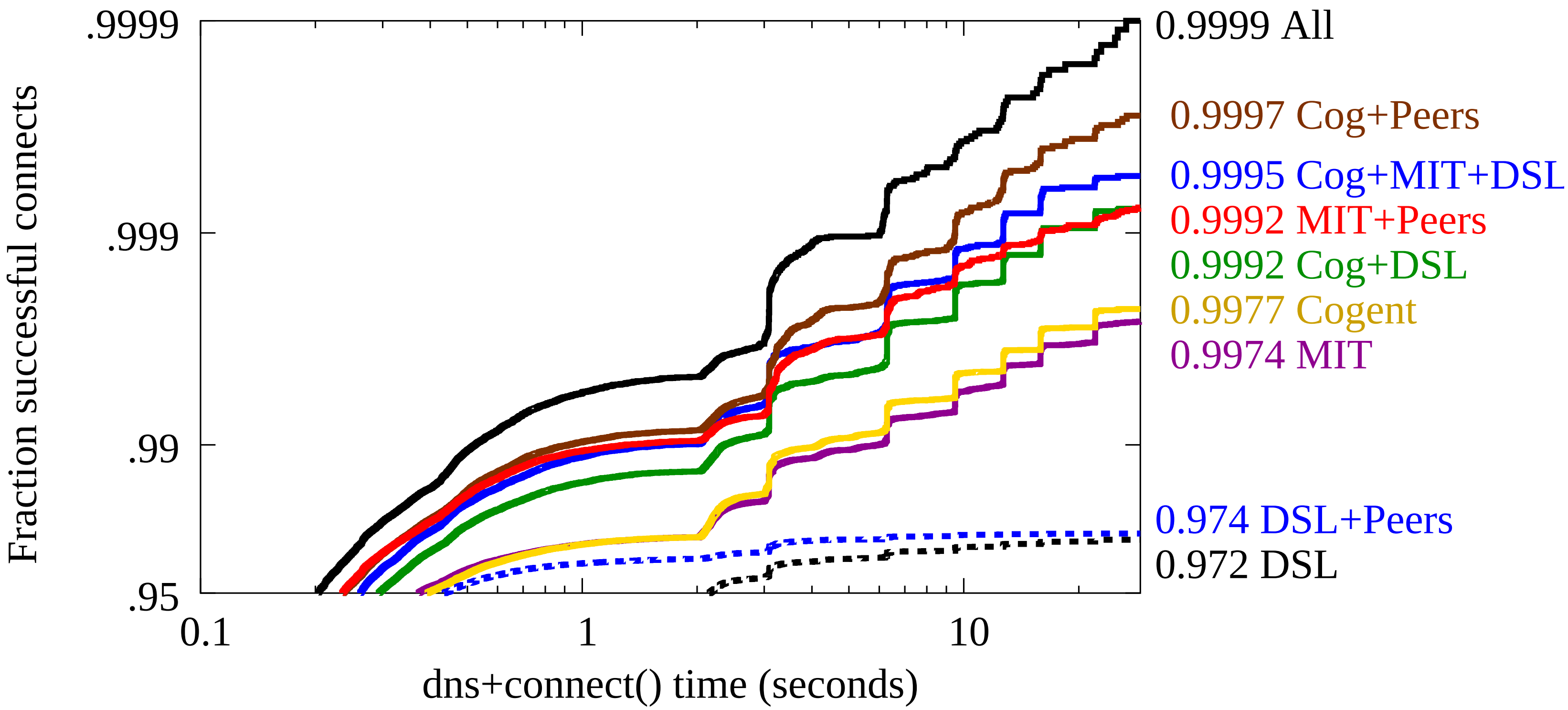


Combined Link Availabilitgy



- Cheap DSL augments 100Mbit link

MONET Achieves 4 Nines



- Cheap DSL augments 100Mbit link
- Overlays + reliable link *very* good

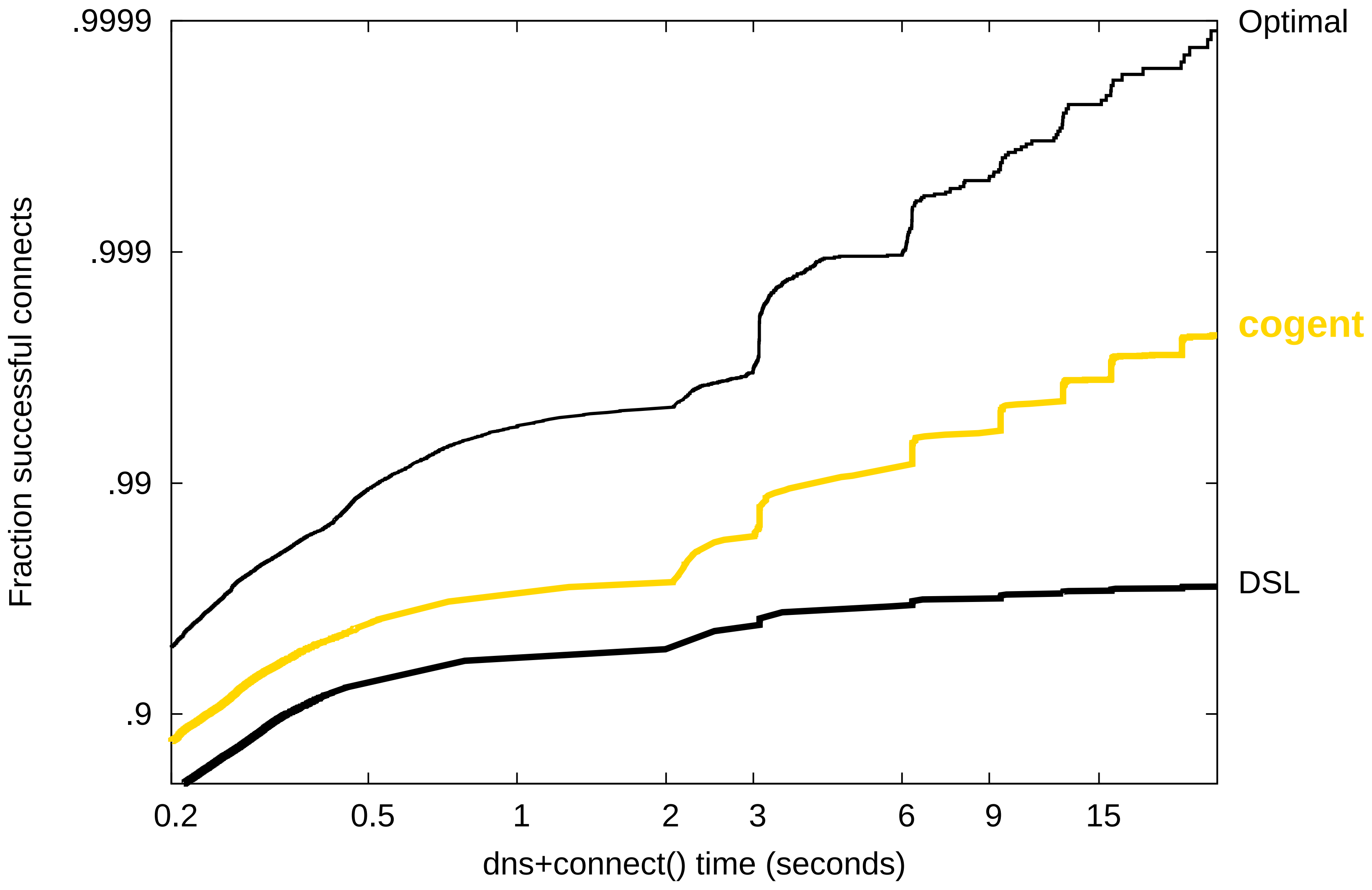
MONET with Low Overhead

How do the practical MONETs compare?

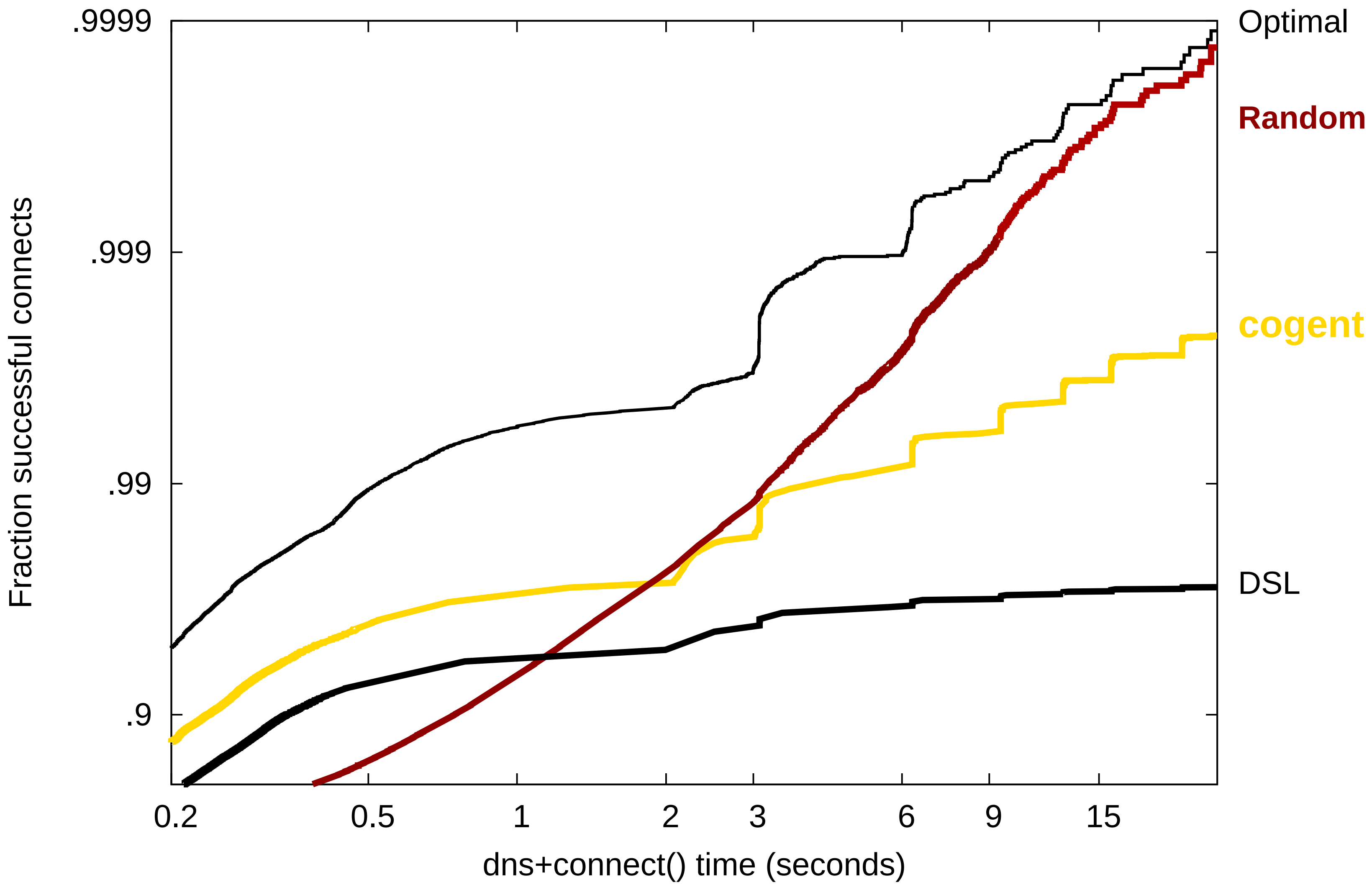
- Optimal, Liveness, Random
- Post-best:
  - Analyze trace, determine single "best" interface to always use first
  - While no response within *thresh*
    - connect via random interface or peer
    - increase *thresh*

(Requires omniscience, but quasi-realistic).

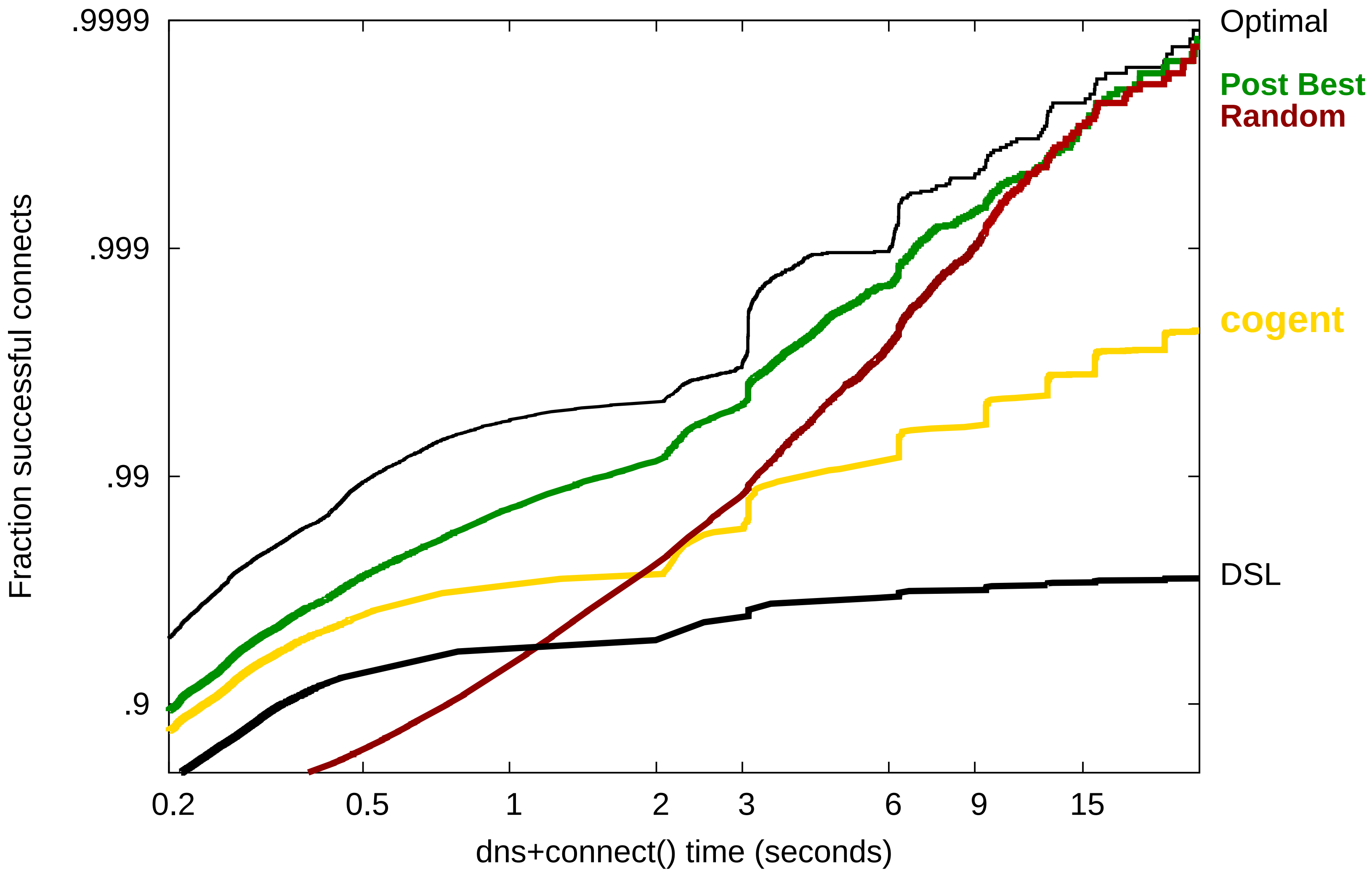
Achievable Resilience



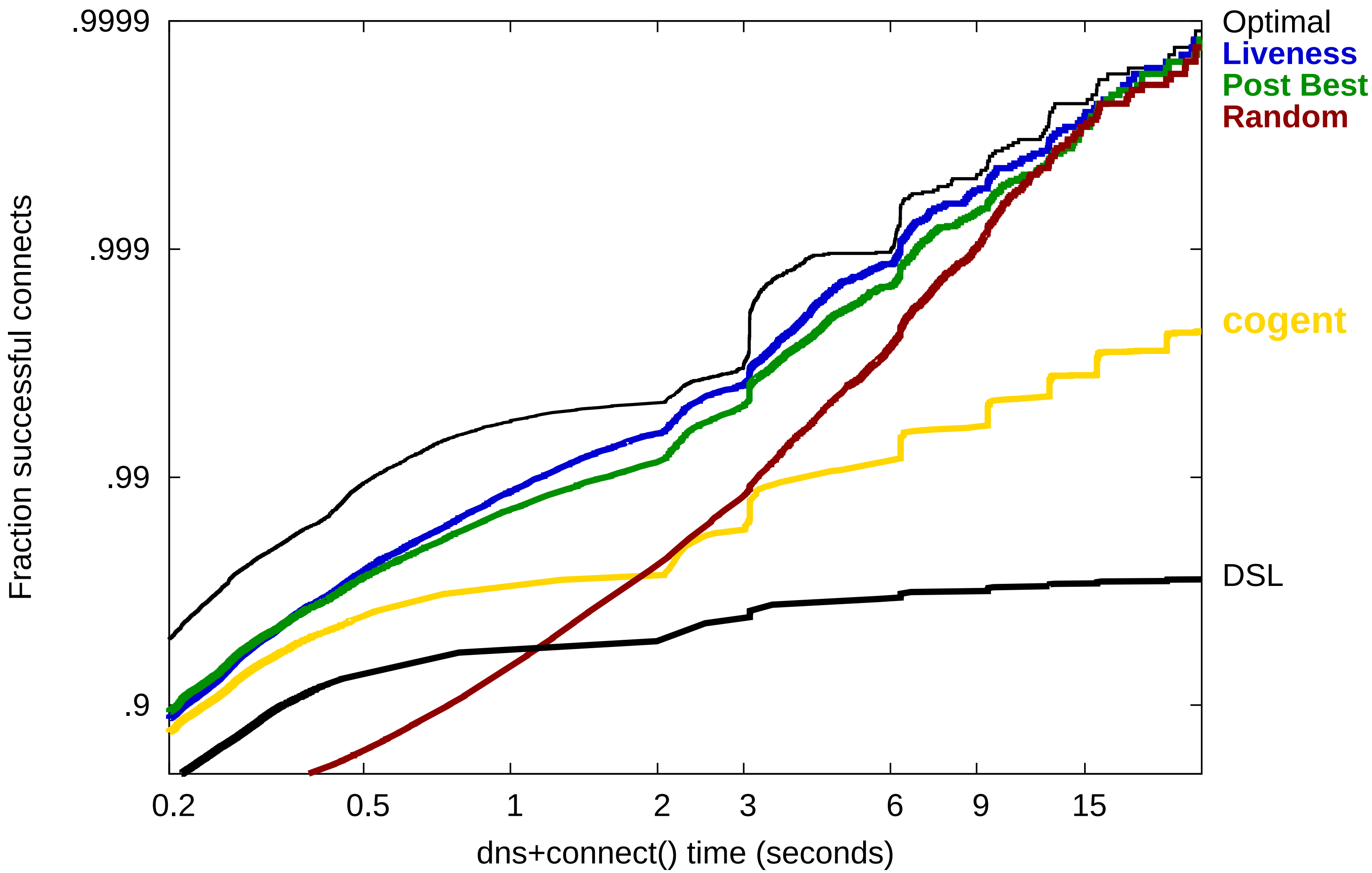
Achievable Resilience



Achievable Resilience



Achievable Resilience



- 10% more SYNs (< 1% packets), near optimal

What we didn't talk about

- Discounted server failures: Some servers *really* bad.
- Paper: MONET + Replicated services
  - A more reliable subset of servers
  - Presumably, operators care more...

✔ 8x better availability *including* server failures.

Related Work

- SOSR (OSDI'04) - single-hop NAT-based overlay routing.
  Probing-based study
- Akella et al. multihoming
  Akamai-based study

→ Similar underlying network performance.

- Commercial products (Stonesoft, Sockeye, ...)
  Tactics, performance, formalize problem
- Content Delivery Networks
  MONET improves availability

Summary

- Expose multiple paths to end-system
  - Choose one that works end-to-end
- Necessary location for availability engineering
- Multihoming *without* routing support
- Resilience achievable with low overhead
- Experience w/2 year deployment and 100s of users: Avoids 90% of failures to reliable sites

`http://nms.lcs.mit.edu/ron/ronweb/`

Bulk Transfers

- Use application knowledge
  - Static objects only
  - HTTP parallel transfers (“Paraloaders”)
- Dykes et al. server selection + our tests
  - First-response SYN effective
- Mid-stream failover
  - SCTP, Migrate, Host ID schemes, others..
  - Range requests / app-specific tactics

TCP_CONTROL_DEFER socket option

- Switch to new server if SYN lost

  Still works if SYN delayed > 3 seconds

- Avoid 3-way handshake completion

  for all but one connection

Time	source	dest	Type
54:31	**client.3430**	> **server-A.80**	SYN
54:34	client.3430	> server-A.80	SYN
		...	
55:05	client.3430	> server-A.80	SYN
55:17	**client.3432**	> **server-B.80**	SYN

Characterizing Failures

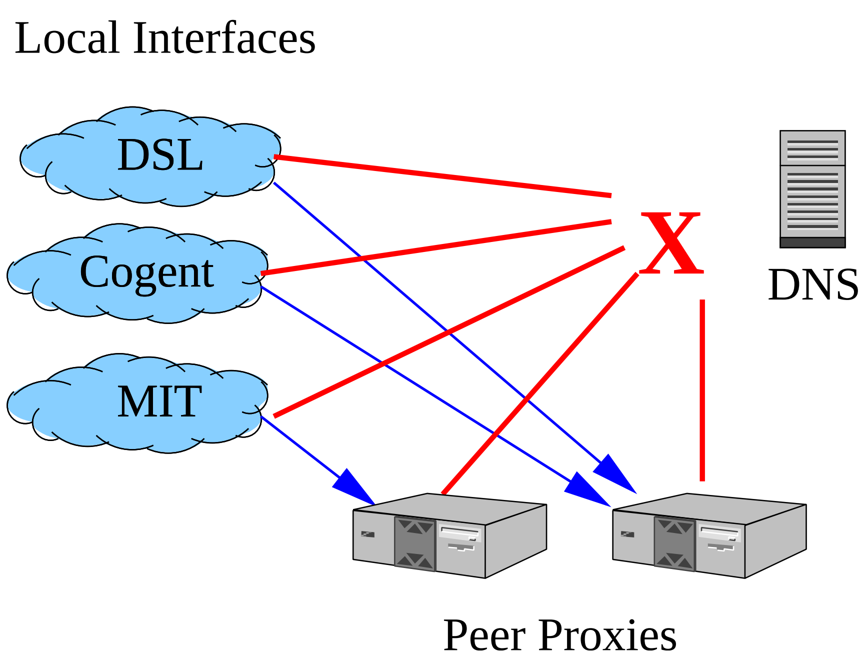
DNS

Server unreach

Server RST

Client access

Wide-area



Peers reachable

no peer or interface could resolve DNS.

Characterizing Failures

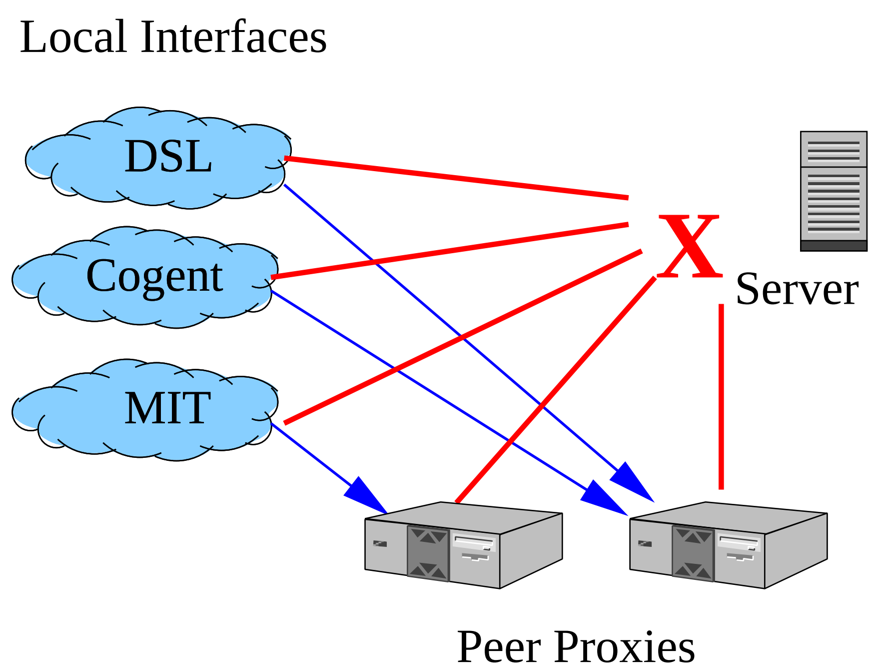
DNS

Server unreach

Server RST

Client access

Wide-area



2+ peers reachable

no peer or link could reach server

(40% unreachable during post-analysis)

Characterizing Failures

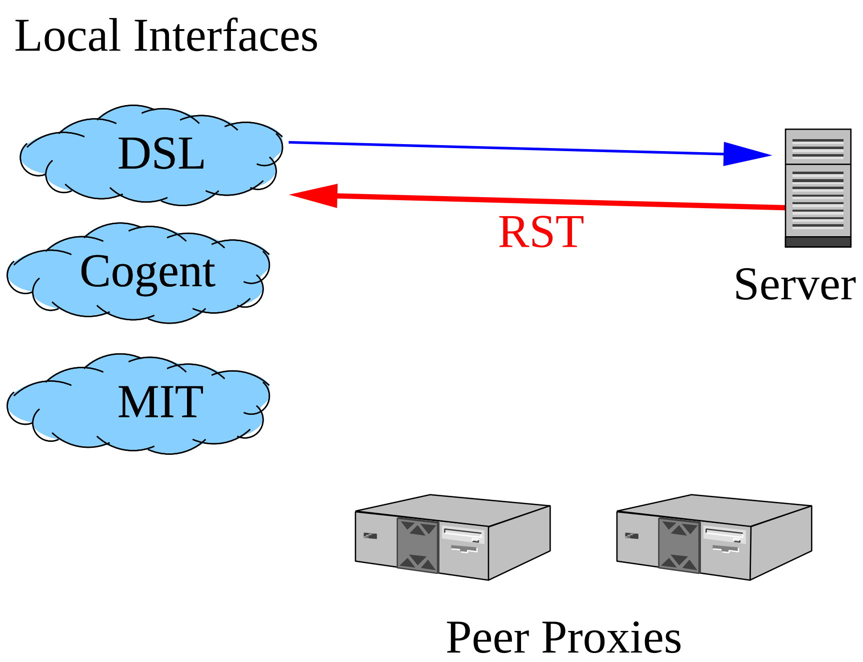
DNS

Server unreach

Server RST

Client access

Wide-area



Server refused TCP connections

Network OK end-to-end.

Characterizing Failures

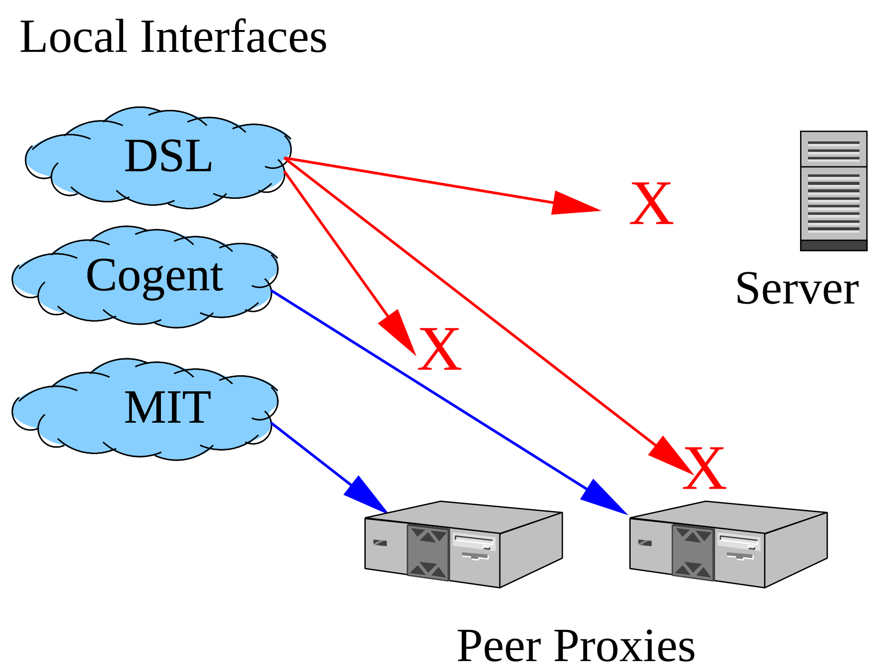
DNS

Server unreach

Server RST

Client access

Wide-area



No peers, DNS or server reachable via one link.

Peers and server working via other links.

Characterizing Failures

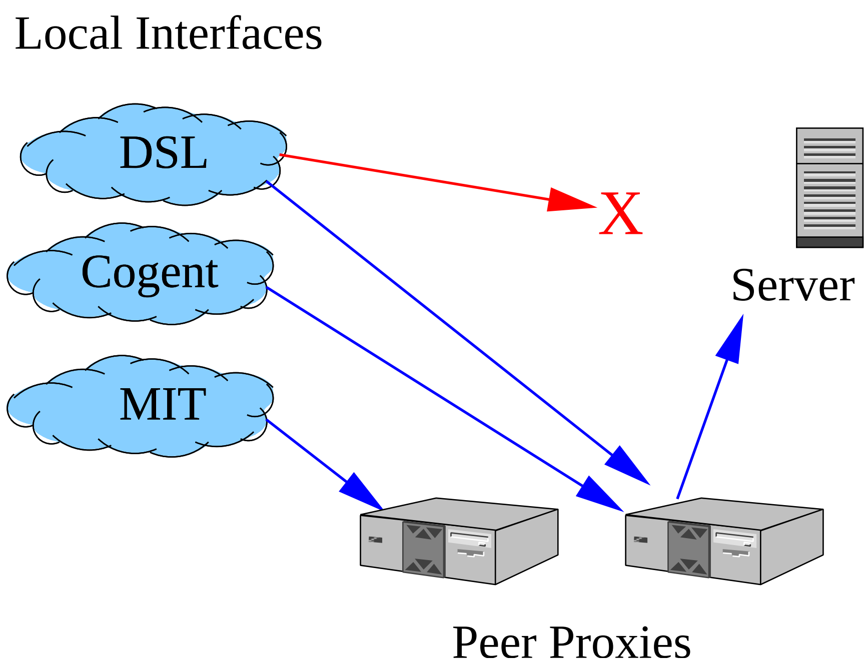
DNS

Server unreach

Server RST

Client access

Wide-area



Server not reachable via one link. That link can reach peers.

Server reachable via peer or other link.

Measurement

Packet-level traces at each node:

- TCP to server, all DNS lookups
- UDP overlay queries

Application traces:

- Proxy request parameters, TCP sessions, DNS queries, overlay queries
- DNS server query log

Sliding-window join links application logs to local and remote packet logs.

When to probe: Practical Solution

Conservative estimator from aggregate connection behavior:

- *rttest* - expected `connect()` time

$$rttest \leftarrow q * rttest + (1 - q) * rtt$$

- *rttdev* - average linear deviation ($> \sigma$)
- *thresh* $= rttest + 4 * rttdev$

✔ Easily computed, little state, effective