Democratizing content distribution

Michael J. Freedman

New York University

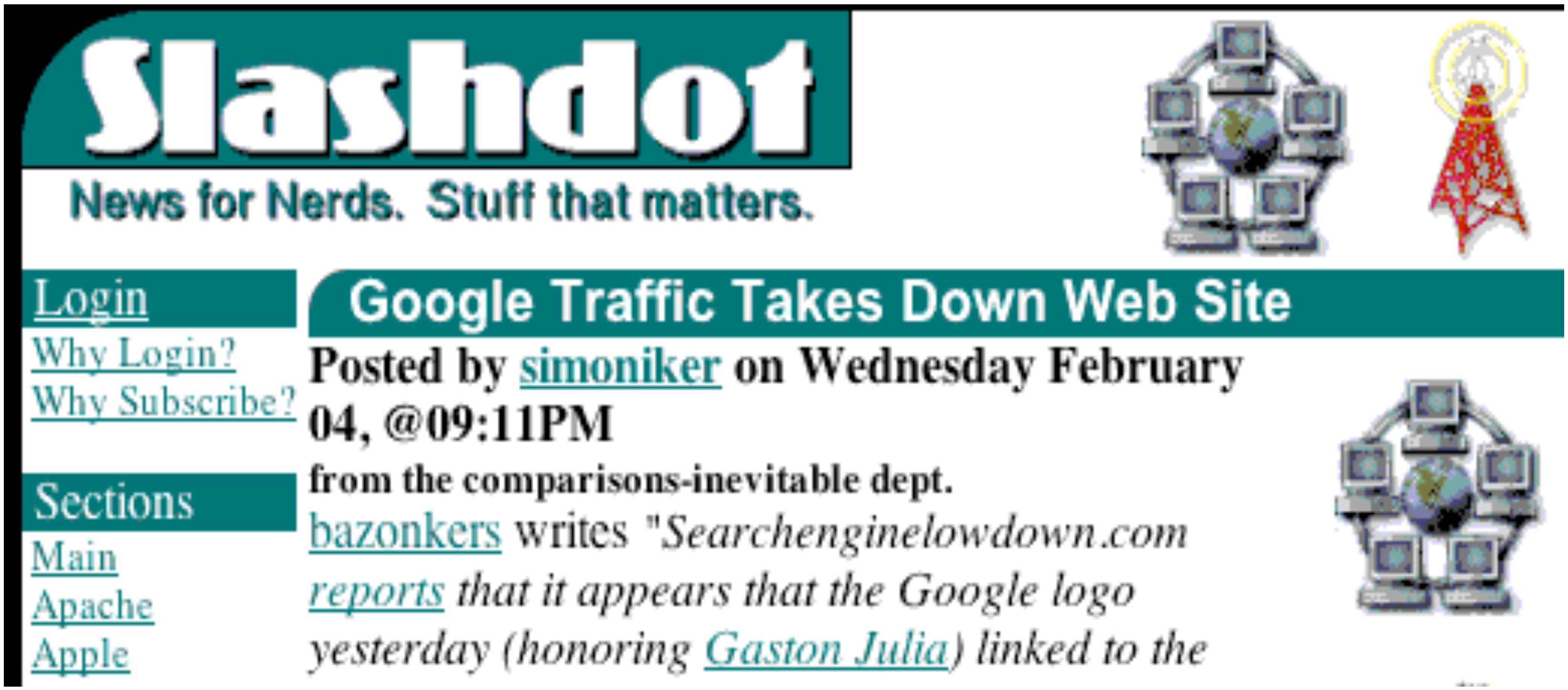
Primary work in collaboration with:
Martin Casado, Eric Freudenthal, Karthik Lakshminarayanan, David Mazières

Additional work in collaboration with:
Siddhartha Annapureddy, Hari Balakrishnan, Dan Boneh, Nick Feamster, Scott Garriss, Yuval Ishai, Michael Kaminsky, Brad Karp, Max Krohn, Nick McKeown, Kobbi Nissim, Benny Pinkas, Omer Reingold, Kevin Shanahan, Scott Shenker, Ion Stoica, and Mythili Vutukuru

Overloading content publishers

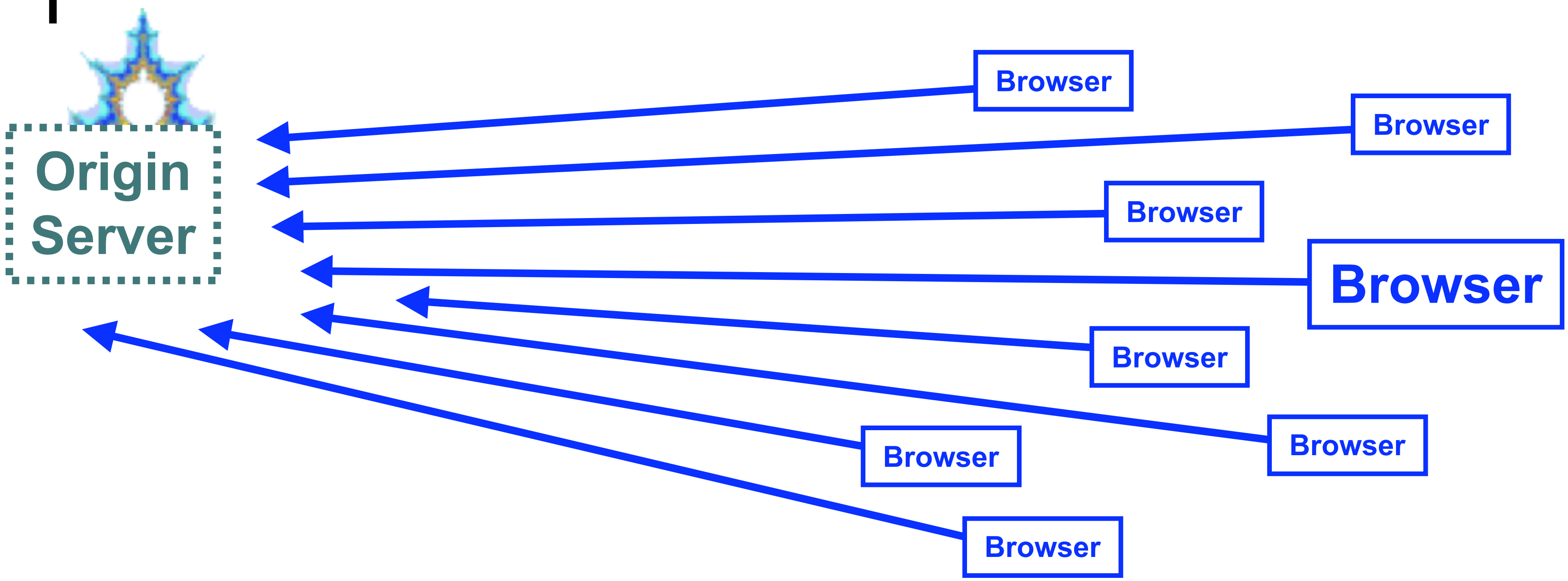
- Feb 3, 2004: Google linked banner to “julia fractals”
- Users clicked onto University of Western Australia web site
- …University’s network link overloaded, web server taken down temporarily…

Adding insult to injury…



- Next day: Slashdot story about Google overloading site
- …UWA site goes down again

Insufficient server resources

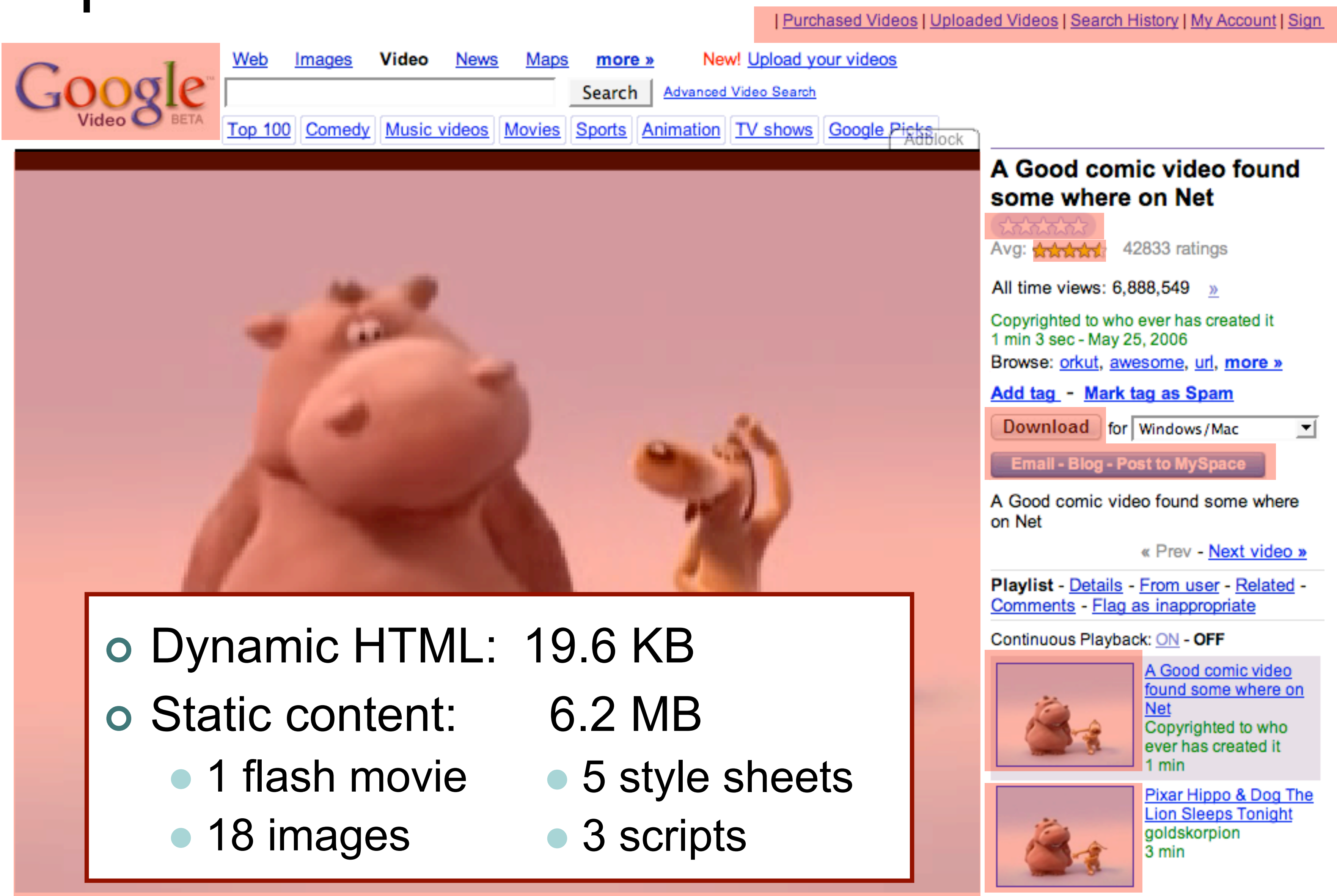


- Many clients want content
- Server has insufficient resources
- Solving the problem requires more resources

Serving large audiences possible…

- Where do their resources come from?
  - Must consider two types of content separately
    - Static
    - Dynamic

Static content uses most bandwidth



- Dynamic HTML: 19.6 KB
- Static content: 6.2 MB
  - 1 flash movie
  - 18 images
  - 5 style sheets
  - 3 scripts

Serving large audiences possible…

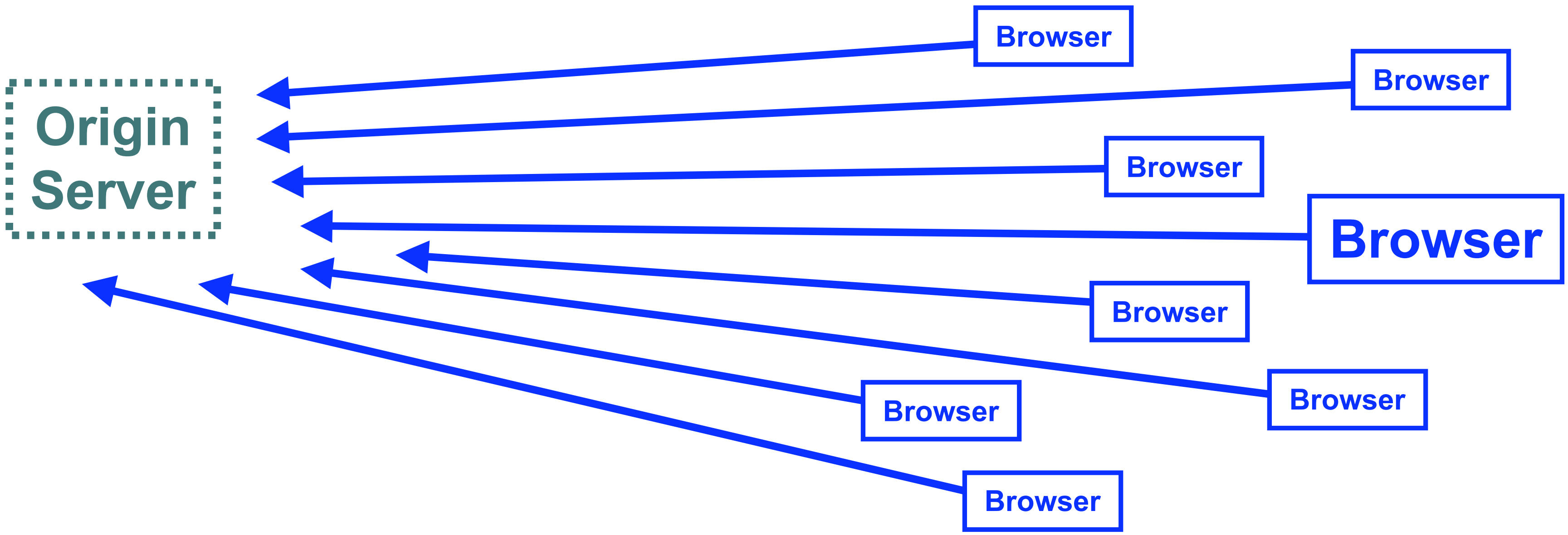
- How do they serve static content?

Content distribution networks (CDNs)

Centralized CDNs

- Static, manual deployment
- Centrally managed
- Implications:
  - Trusted infrastructure
  - Costs scale linearly

Not solved for little guy



- Problem:
  - Didn’t anticipate sudden load spike (flash crowd)
  - Wouldn’t want to pay / couldn’t afford costs

Leveraging cooperative resources

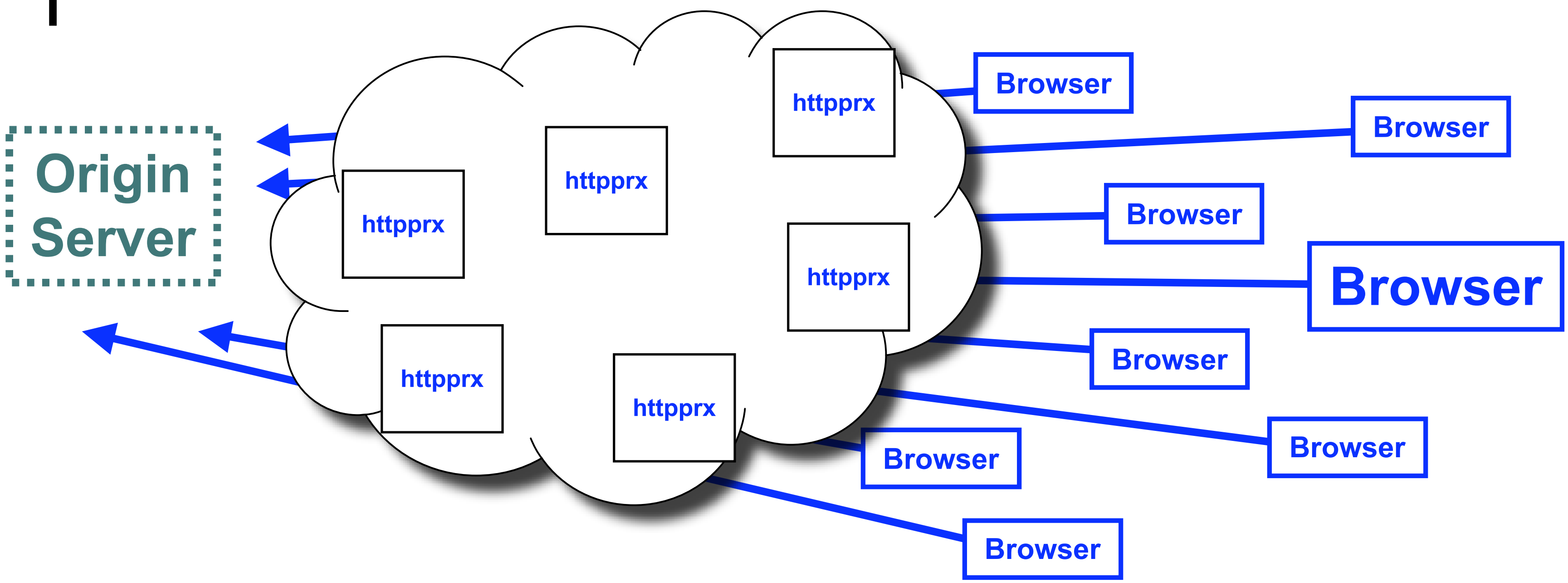
- Many people want content
- Many willing to mirror content
  - e.g., software mirrors, file sharing, open proxies, etc.
- Resources are out there

…if only we can leverage them

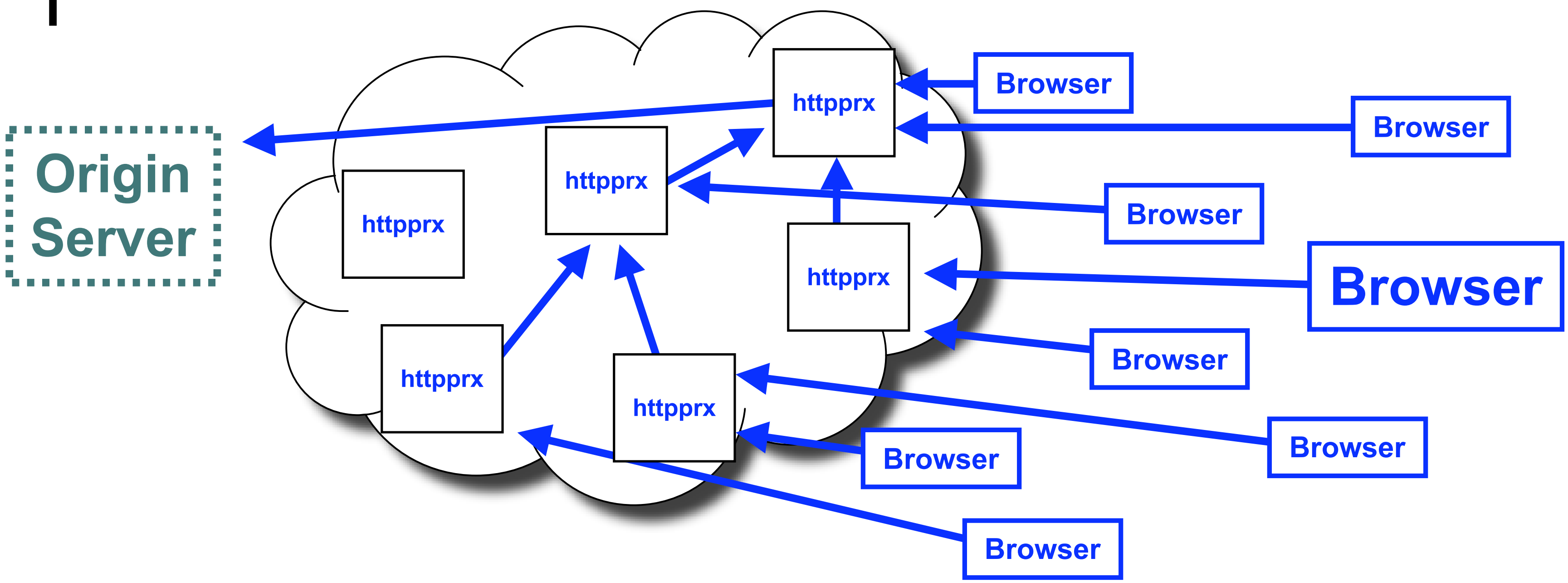
- Contributions

Theme throughout talk: How to leverage previously untapped resources to gain new functionality

Proxies absorb client requests



Proxies absorb client requests



- Reverse proxies handle all client requests
- Cooperate to fetch content from one another

A comparison of settings

Centralized CDNs

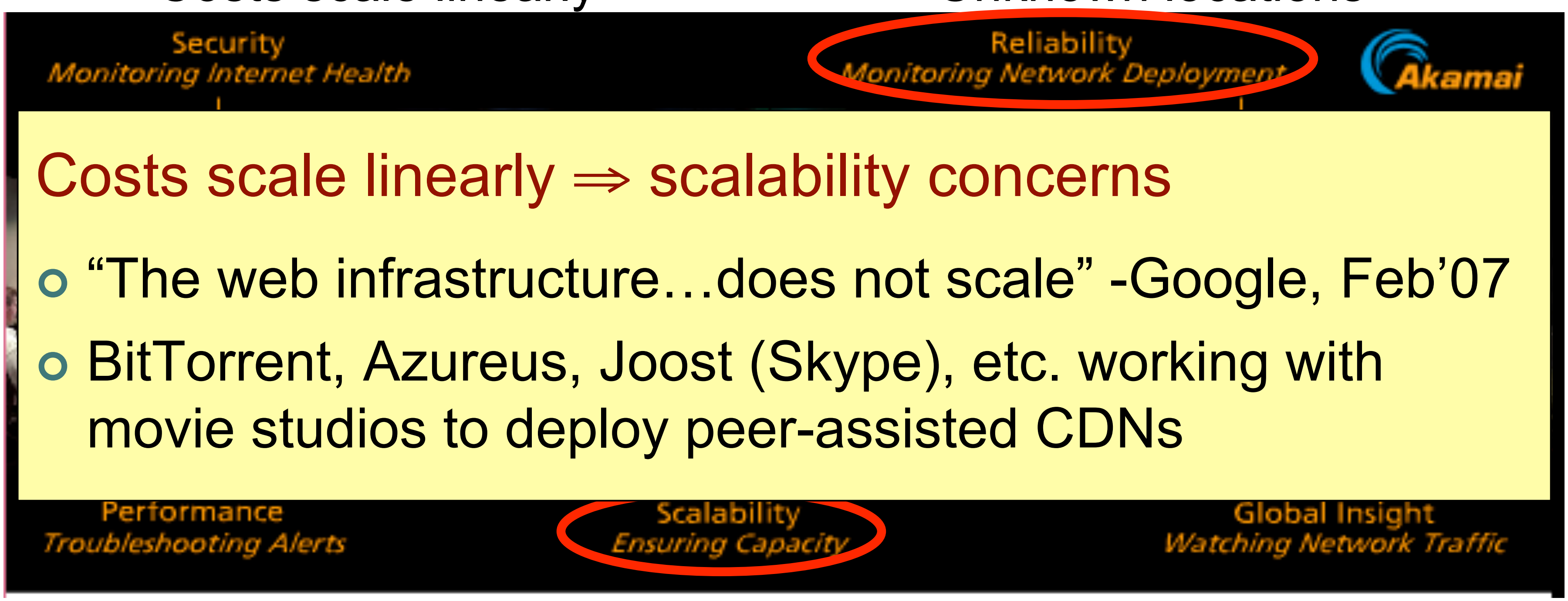
- Static, manual deployment
- Centrally managed
- Implications:
  - Trusted infrastructure
  - Costs scale linearly

Decentralized CDNs

- Use participating machines
- No central operations
- Implications:
  - Less reliable or untrusted
  - Unknown locations

A comparison of settings

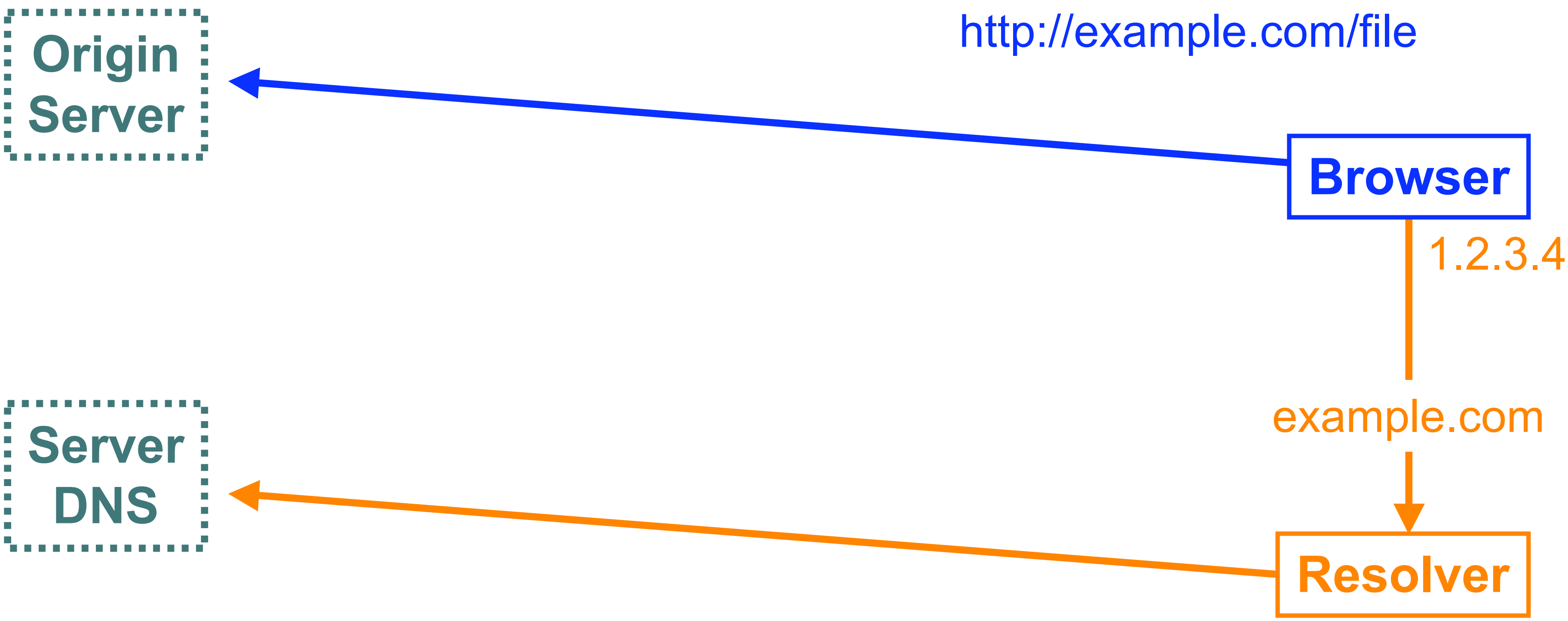
Centralized CDNs	Decentralized CDNs
Static, manual deployment	Use participating machines
Centrally managed	No central operations
Implications:	Implications:
• Trusted infrastructure	• Less reliable or untrusted
• Costs scale linearly	• Unknown locations



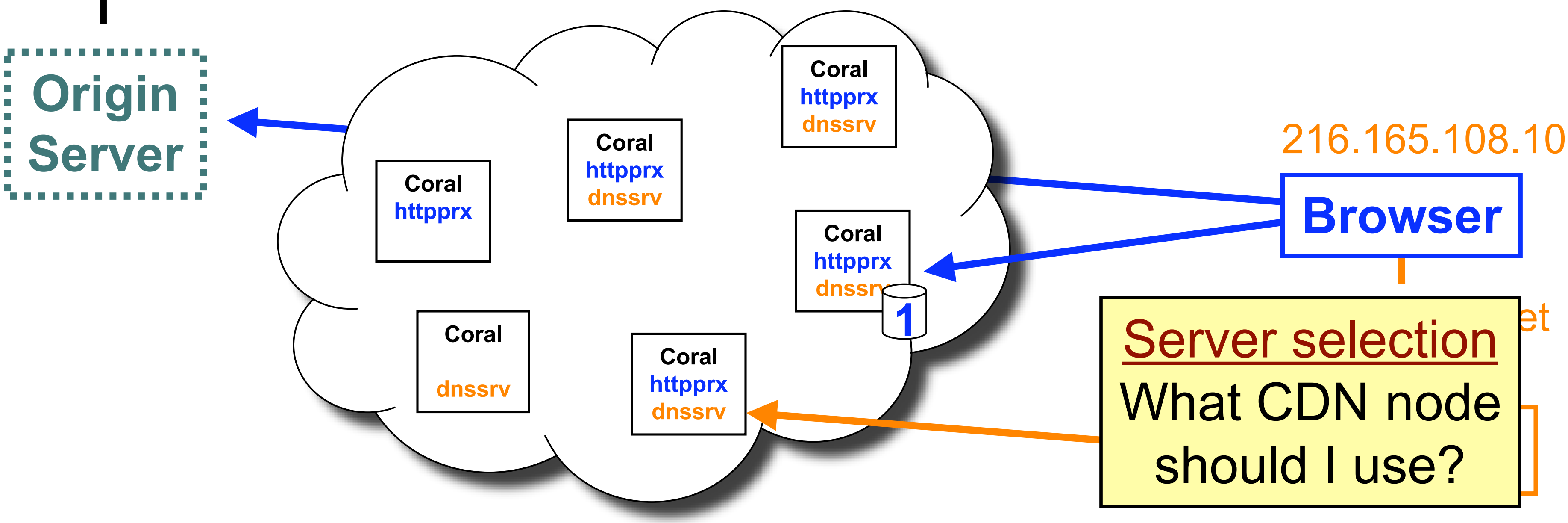
Costs scale linearly ⇒ scalability concerns

- “The web infrastructure…does not scale” -Google, Feb’07
- BitTorrent, Azureus, Joost (Skype), etc. working with movie studios to deploy peer-assisted CDNs

Getting content



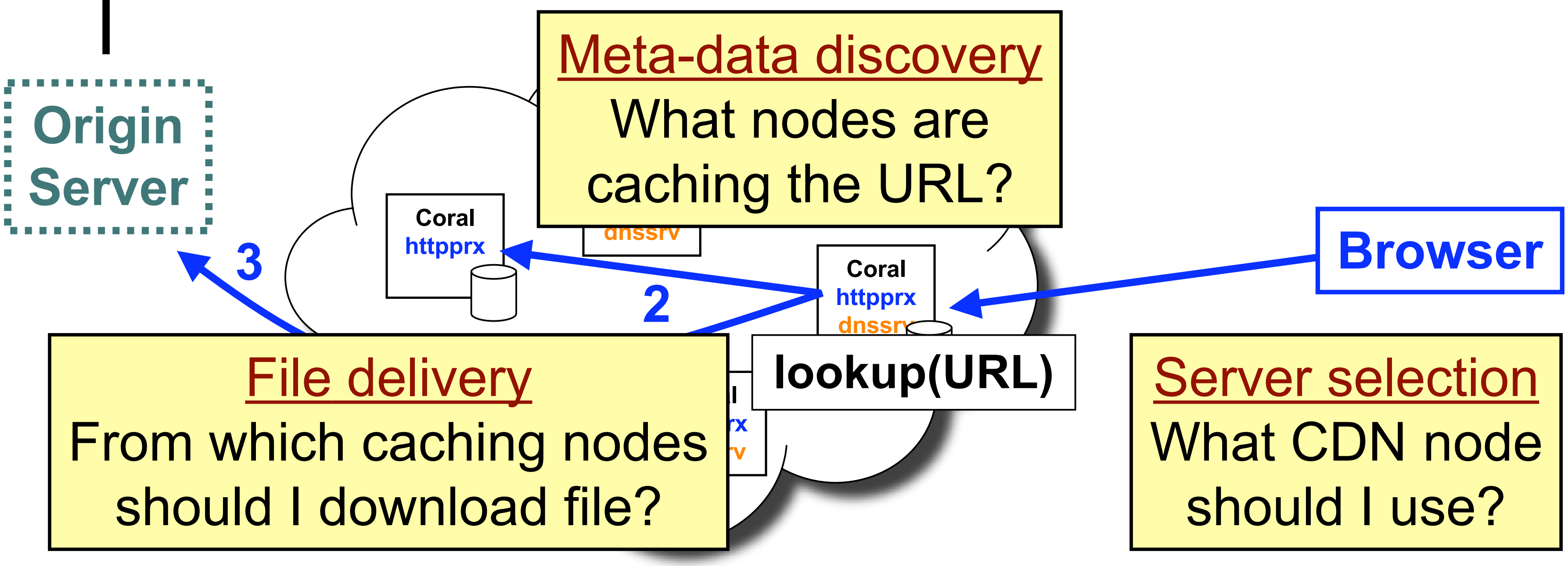
Getting content with CoralCDN



- Participants run CoralCDN software, no configuration
- Clients use CoralCDN via modified domain name

  example.com/file → example.com.nyud.net:8080/file

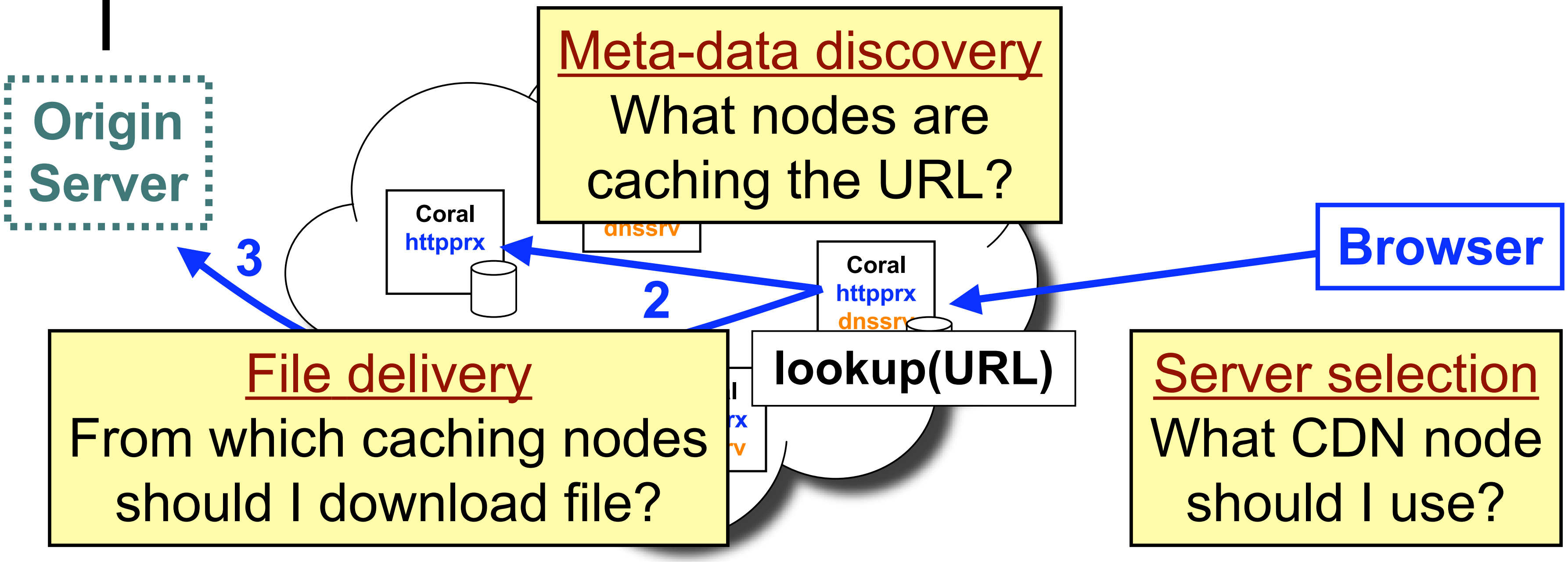
Getting content with CoralCDN



- Participants run CoralCDN software, no configuration
- Clients use CoralCDN via modified domain name

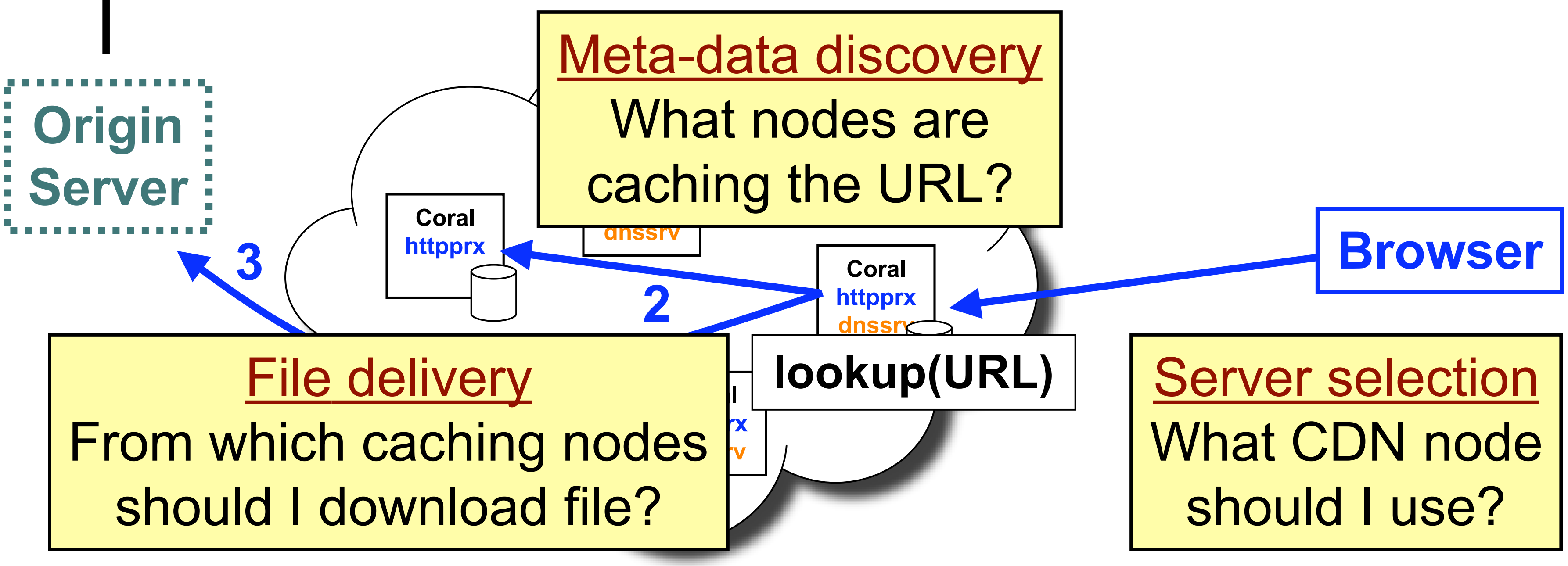
  example.com/file → example.com.nyud.net:8080/file

Getting content with CoralCDN



- Goals
  - Reduce load at origin server
  - Low end-to-end latency
  - Self-organizing

Getting content with CoralCDN

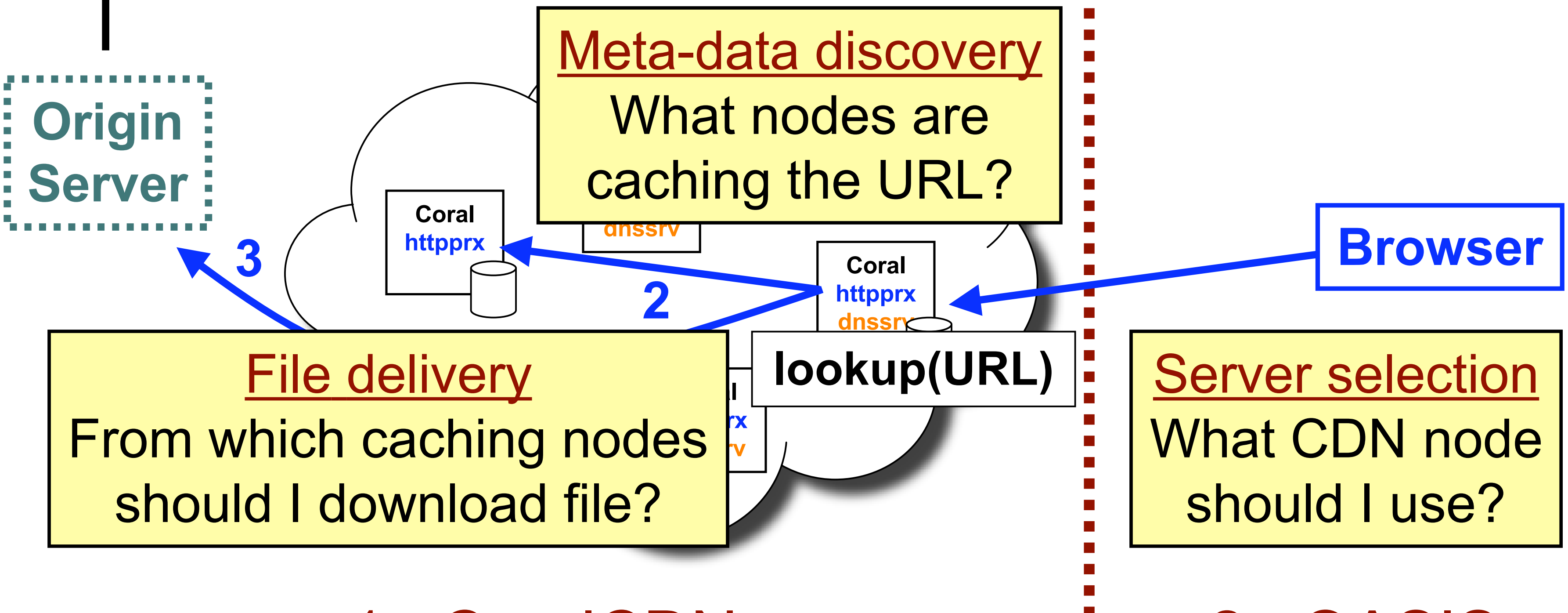


- Why participate?
  - Ethos of volunteerism
  - Cooperatively weather peak loads spread over time
  - Incentives: Better performance when resources scarce

This talk

[IPTPS '03] [NSDI '04]

[NSDI '06]



1. CoralCDN
2. OASIS

3. Using these for measurements: Illuminati [NSDI '07]
4. Finally, adding security to leverage more volunteers

“ Real deployment ”

Powered By PLANETLAB

- Currently deployed on 300-400 PlanetLab servers
  - CoralCDN running 24 / 7 since March 2004

- An open CDN for *any* URL:

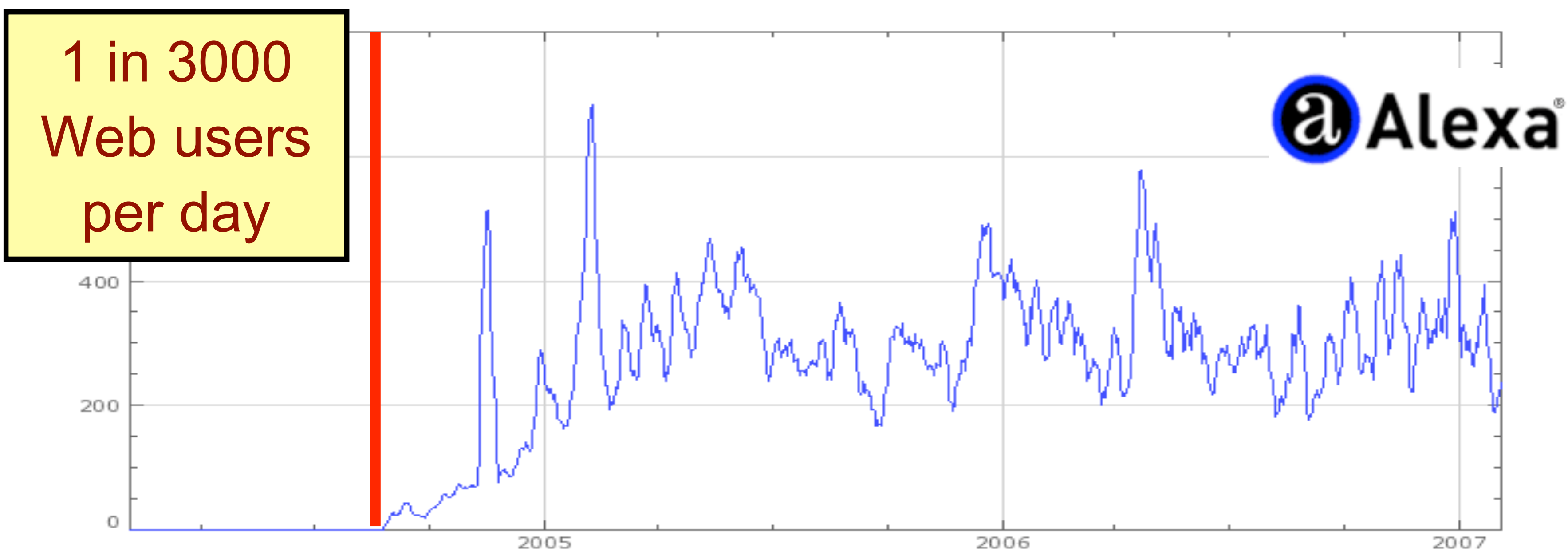
  example.com/file → example.com.nyud.net:8080/file

“ Real deployment ”

Powered By PLANETLAB

- Currently deployed on 300-400 PlanetLab servers
  - CoralCDN running 24 / 7 since March 2004
- An open CDN for *any* URL:

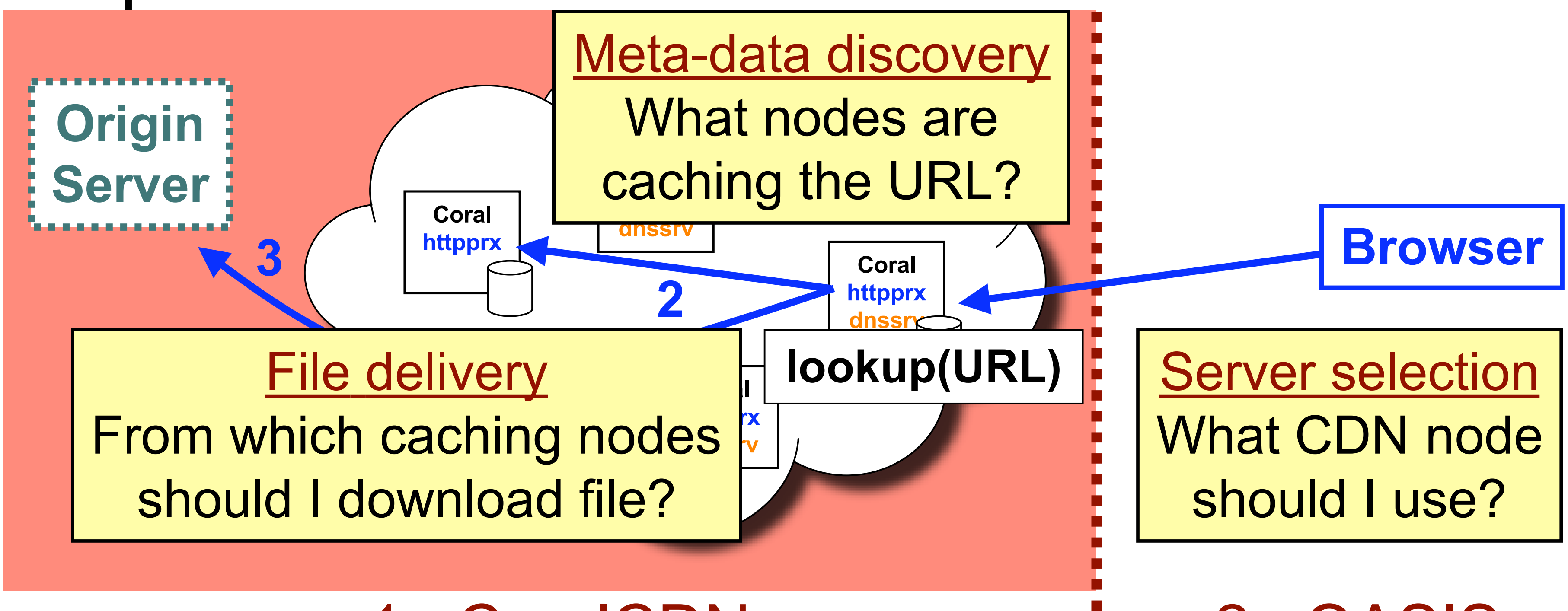
  example.com/file → example.com.nyud.net:8080/file



This talk

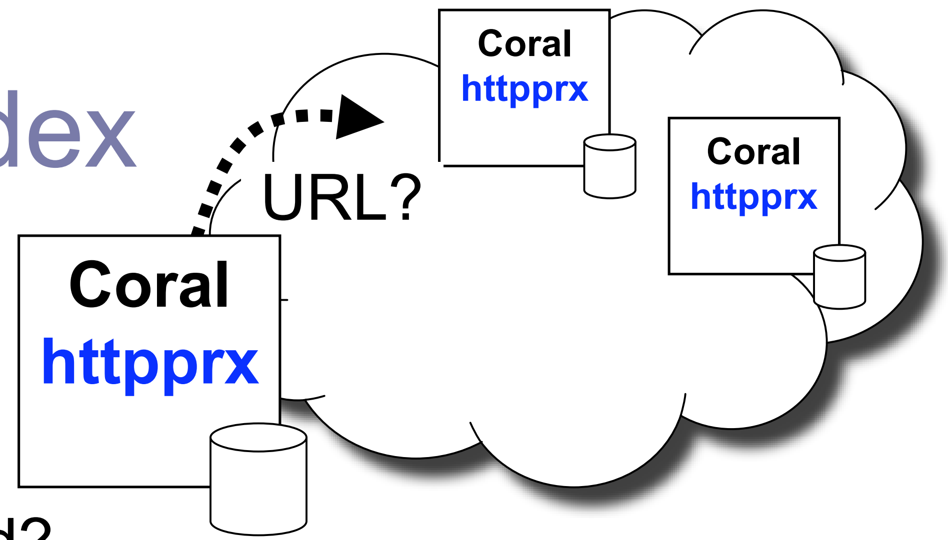
[IPTPS '03]
[NSDI '04]

[NSDI '06]



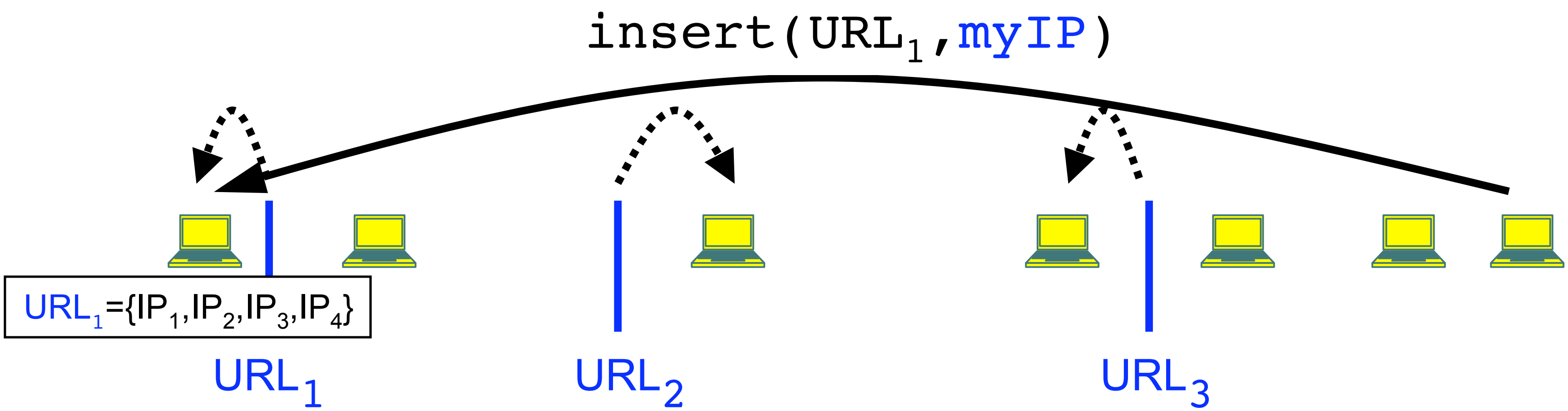
1. CoralCDN
2. OASIS
3. Using these for measurements: Illuminati [NSDI '07]
4. Finally, adding security to leverage more volunteers

We need an index



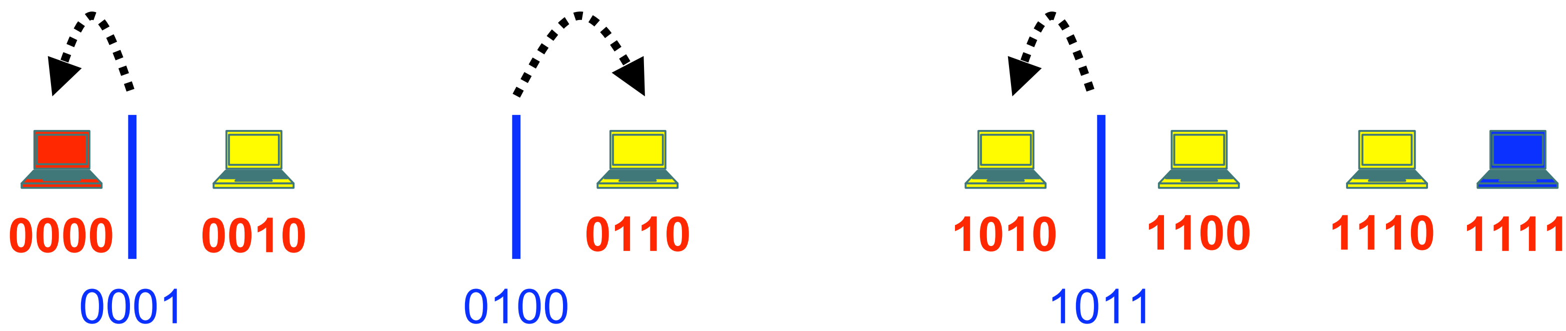
- Given a URL:
  - Where is the data cached?
  - Map name to location: URL ⇒ {IP_1, IP_2, IP_3, IP_4}
  - `lookup(URL)` ⇒ Get IPs of caching nodes
  - `insert(URL,myIP,TTL)` ⇒ Add me as caching URL for TTL seconds
- Can't index at central servers
  - No individual machines reliable or scalable enough
- Need to distribute index over participants

Strawman: distributed hash table (DHT)



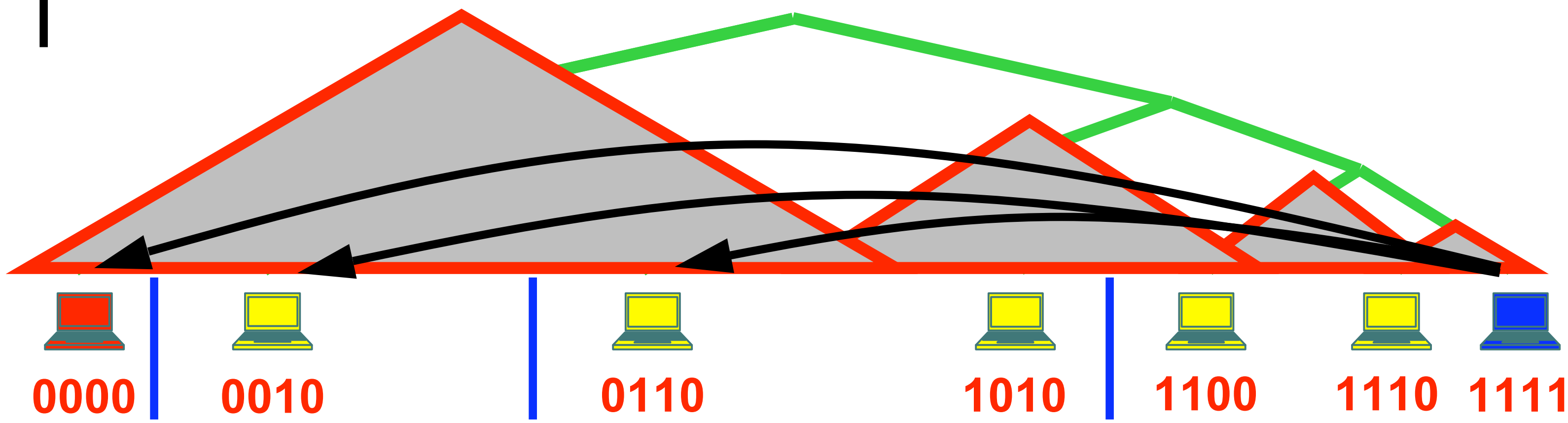
- Use DHT to store mapping of URLs (keys) to locations
- DHTs partition key-space among nodes
- Contact appropriate node to lookup/store key
  - Blue node determines red node is responsible for URL
  - Blue node sends lookup or insert to red node

Strawman: distributed hash table (DHT)



- Partitioning key-space among nodes
  - Nodes choose random identifiers: hash(IP)
  - Keys randomly distributed in ID-space: hash(URL)
  - Keys assigned to node nearest in ID-space
    - Minimizes *XOR(hash(IP),hash(URL))*

Strawman: distributed hash table (DHT)



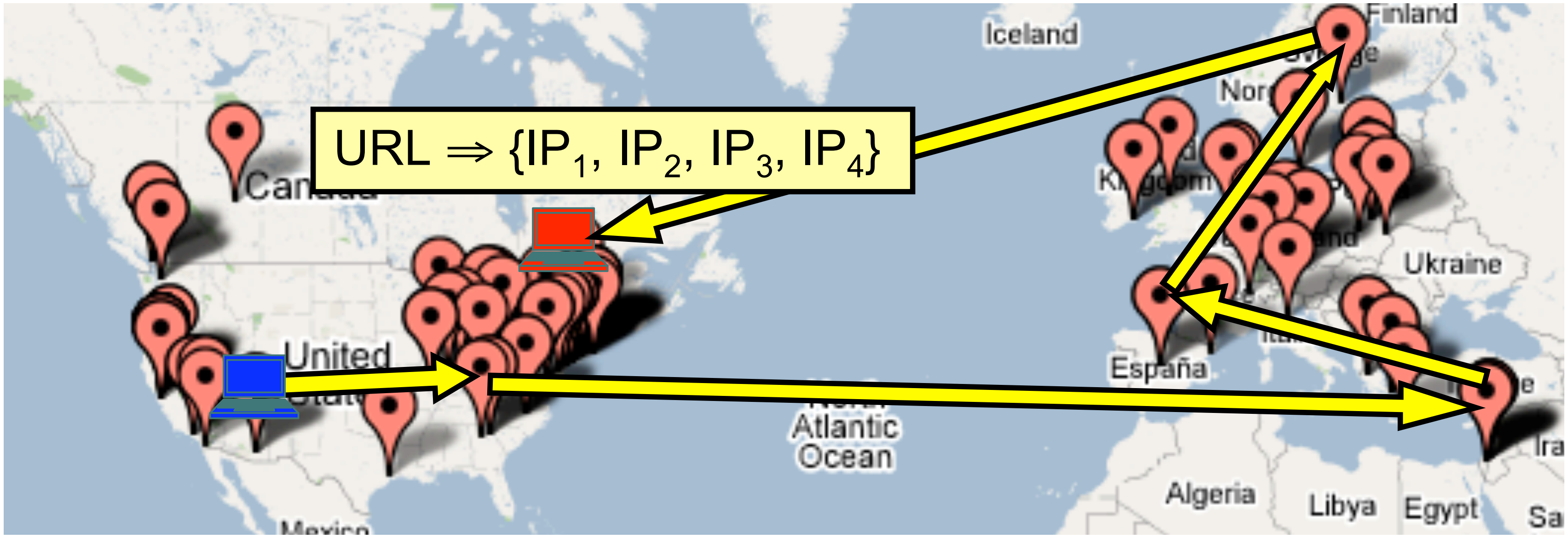
- Provides "efficient" routing with small state

  If *n* is # nodes, each node:

  - Monitors *O(log n)* peers
  - Discovers closest node (and URL map) in *O(log n)* hops
  - Join/leave requires O(log n) work

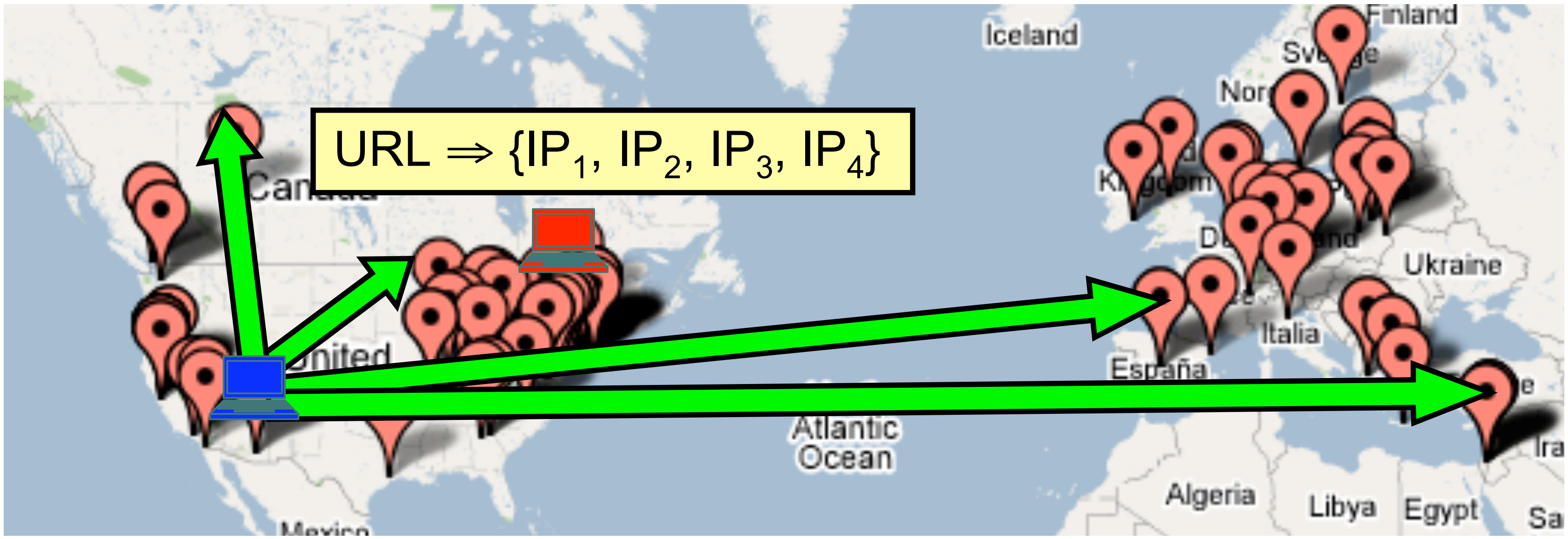
- Spread ownership of URLs evenly across nodes

Is this index sufficient?



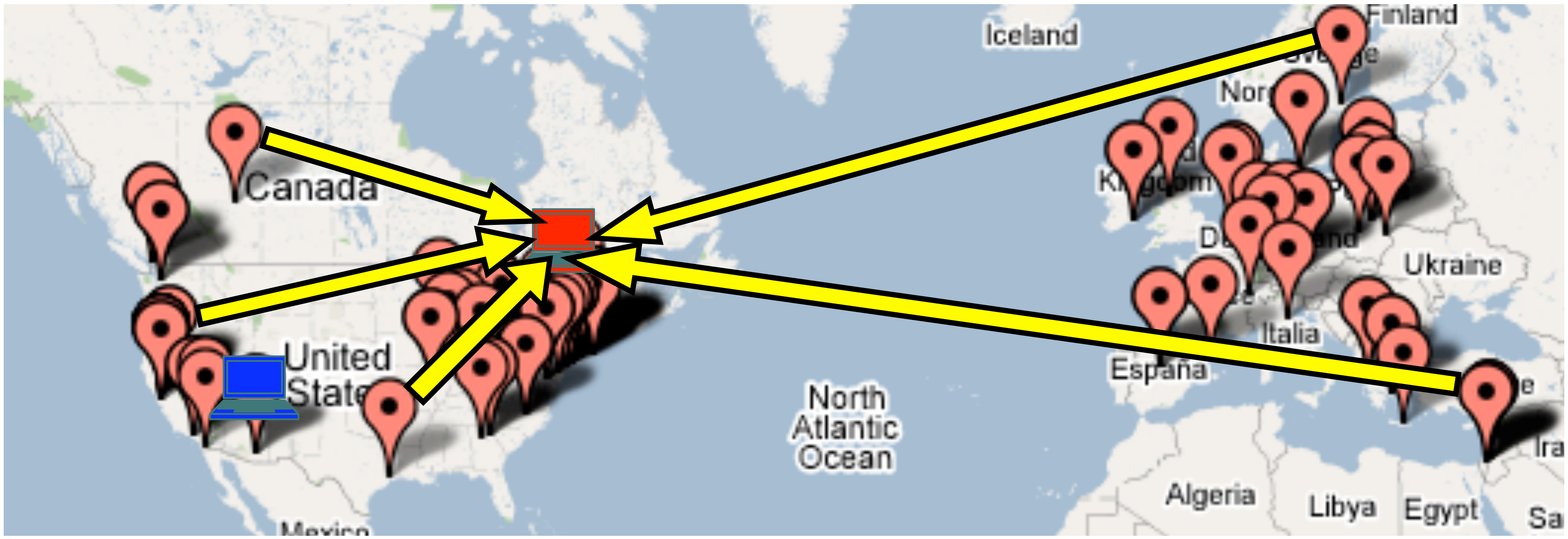
- **Problem:** Random routing

Is this index sufficient?



- Problem: Random routing
- Problem: Random downloading

Is this index sufficient?



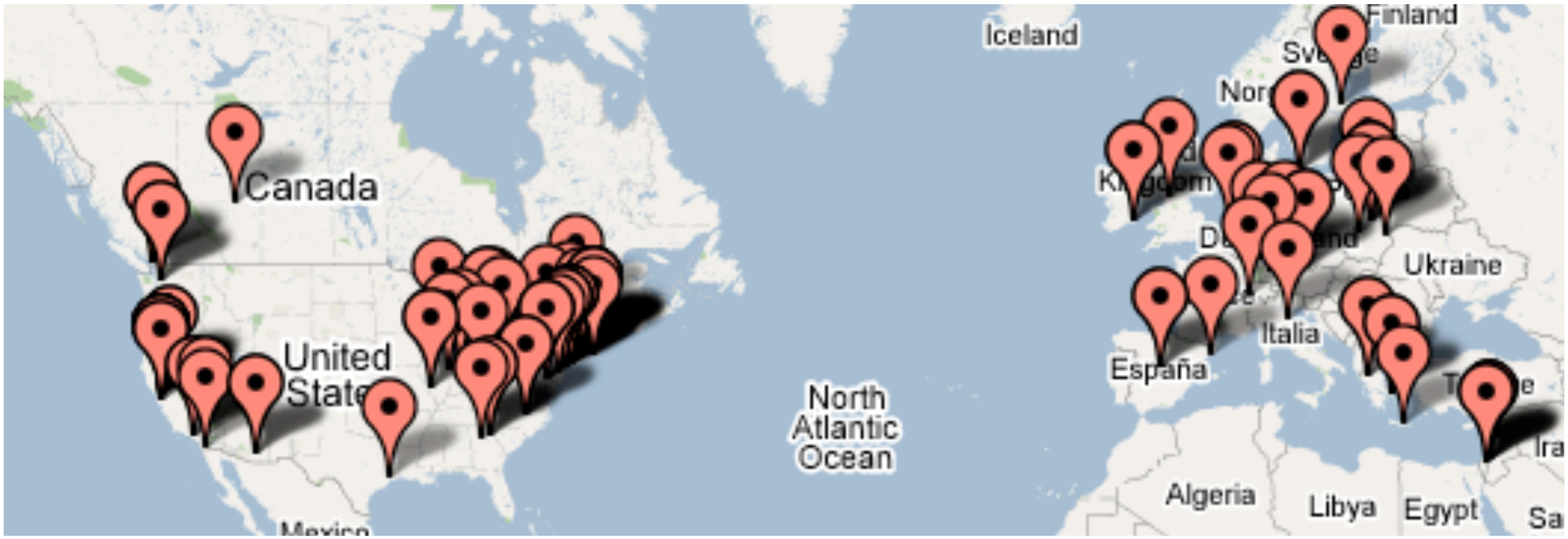
- Problem: Random routing
- Problem: Random downloading
- Problem: No load-balancing for single item
  - All `insert` and `lookup` go to same closest node

Don't need hash-table semantics

- DHTs designed for hash-table semantics
  - Insert and replace: URL ⇒ IP_{last}
  - Insert and append: URL ⇒ {IP_1, IP_2, IP_3, IP_4}

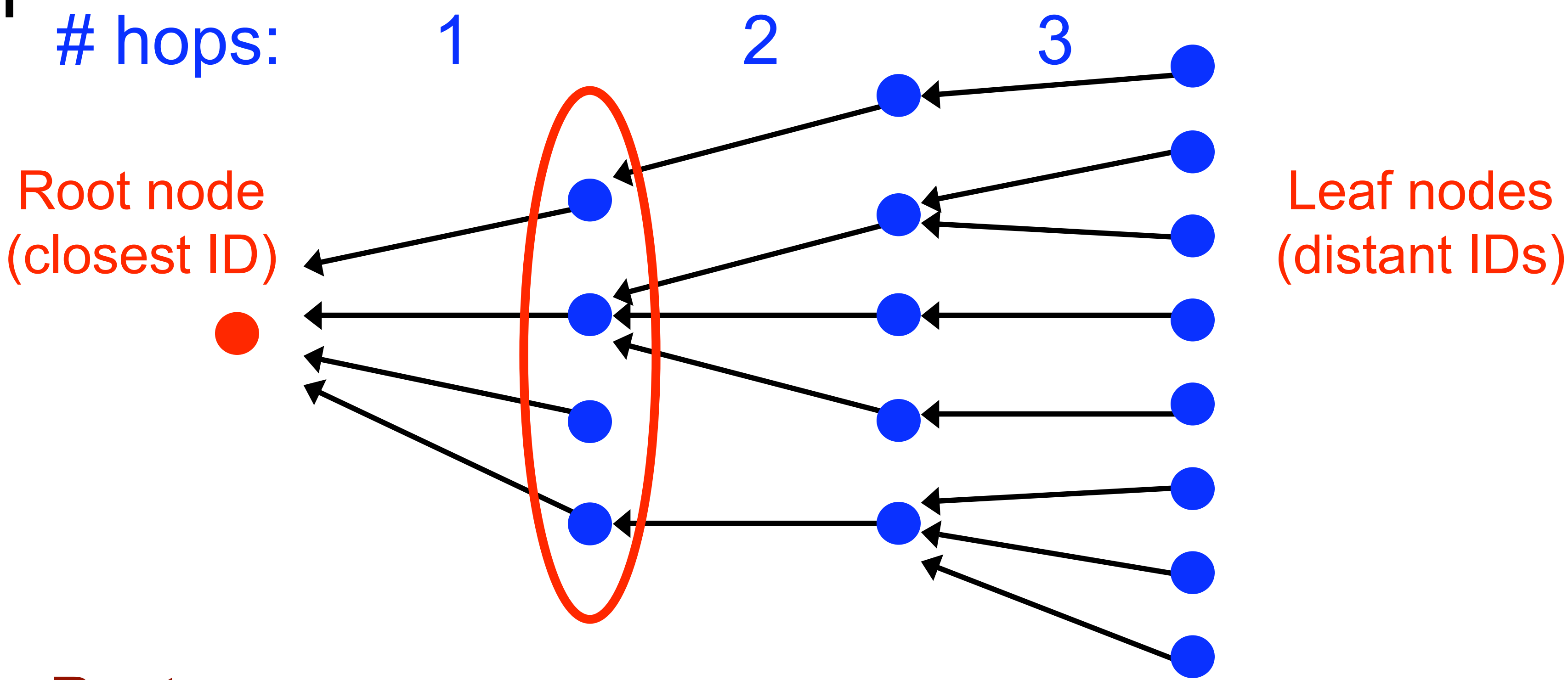
- We only need few values
  - `lookup`(URL) ⇒ {IP_2, IP_4}
  - Preferably ones close in network

Next…



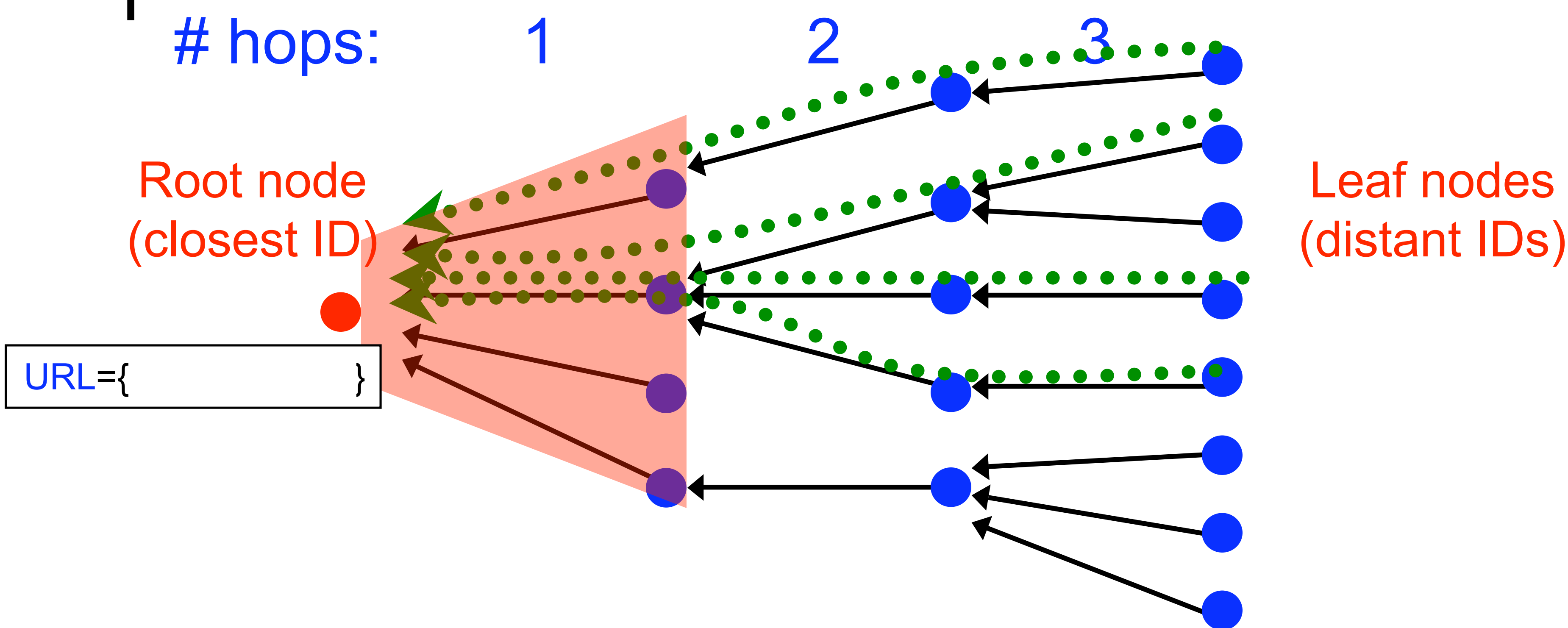
- Solution: Bound request rate to prevent hotspots
- Solution: Take advantage of network locality

Prevent hotspots in index



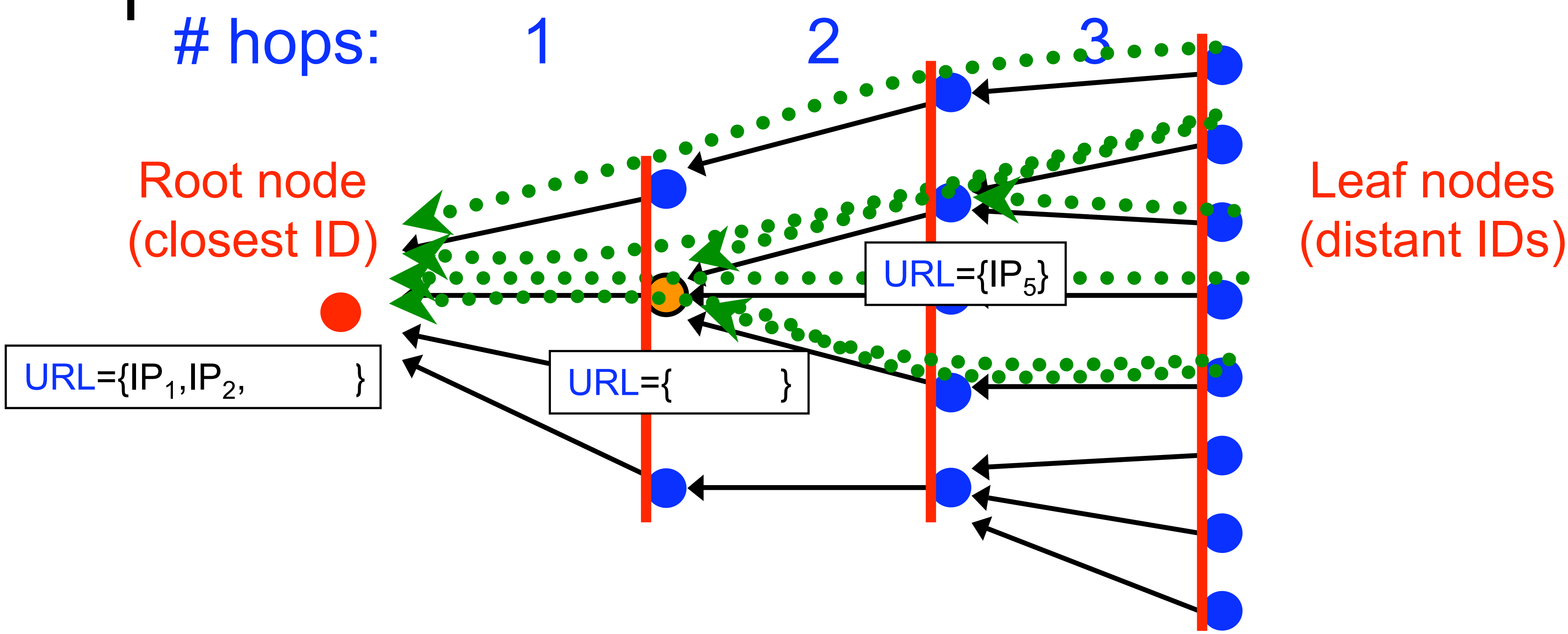
- Route convergence
  - *O(log n)* nodes are 1 hop from root

Prevent hotspots in index



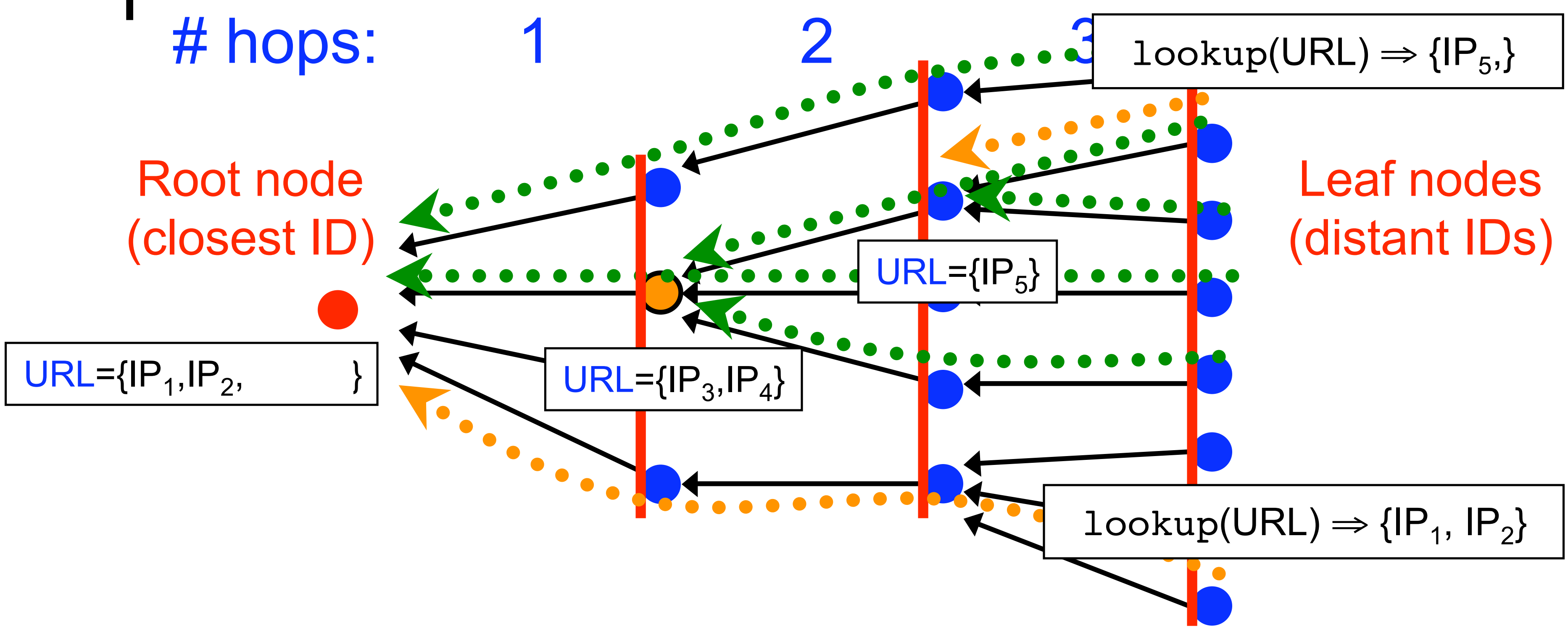
- Route convergence
  - *O(log n)* nodes are 1 hop from root
- Request load increases exponentially towards root

Rate-limiting requests



- Bound rate of inserts towards root
  - Nodes leak through at most β inserts per min per URL
- Locations of popular items pushed down tree
  - Refuse if already storing max # “fresh” IPs per URL

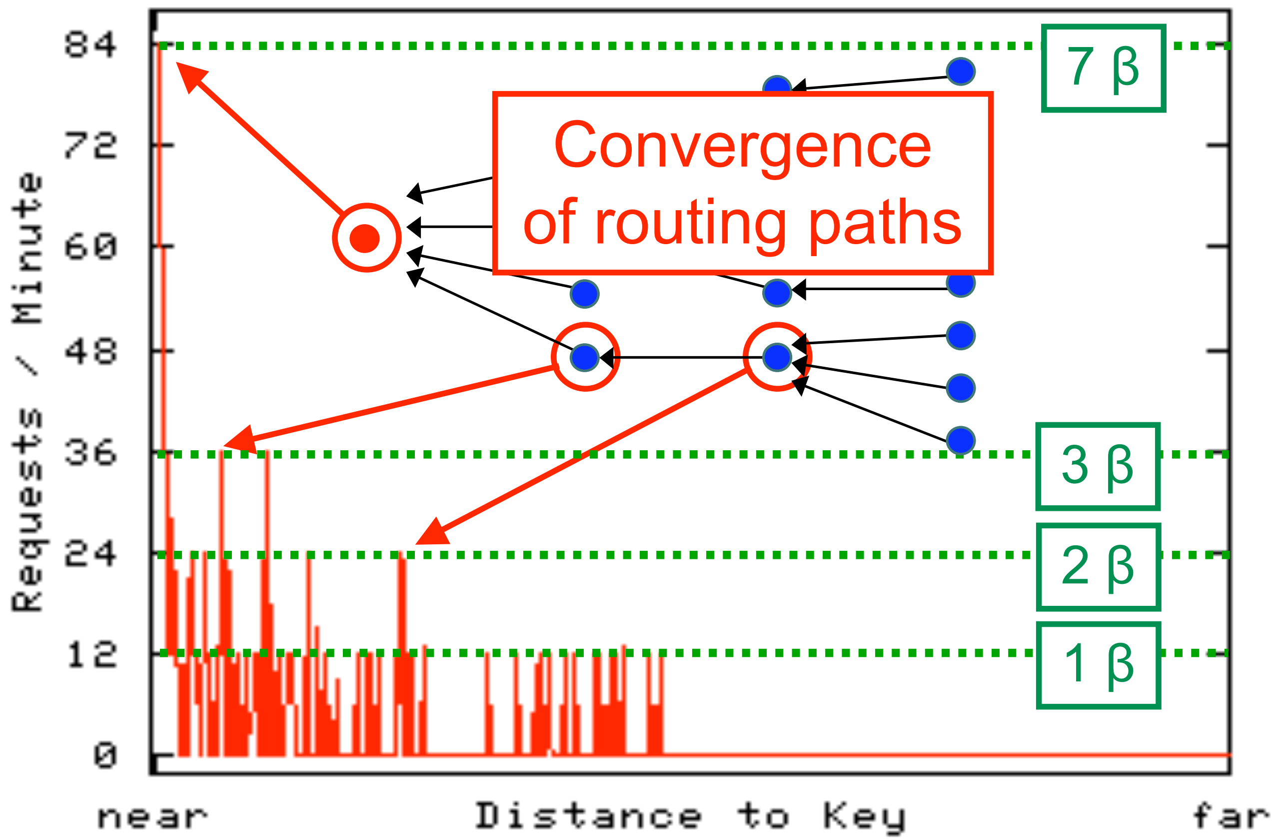
Rate-limiting requests



- **High load:** Most stored on path, few on root

Theorem: Fixing *b* bits per hop, root receives $\beta \cdot (2^b - 1) \cdot \left\lceil \frac{\log_{b+1} n}{b} \right\rceil$ insertion requests per time period

Wide-area results follow analytics

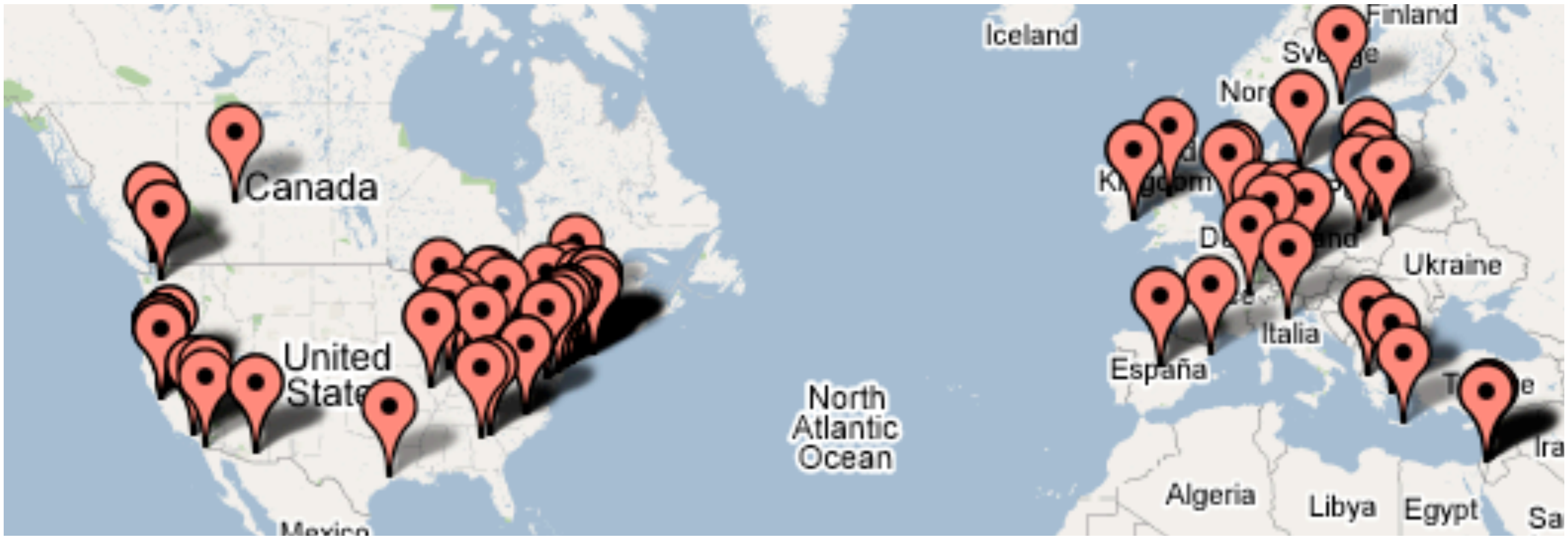


494 nodes on PlanetLab

$$\lceil \log_2(494) \rceil = 9$$

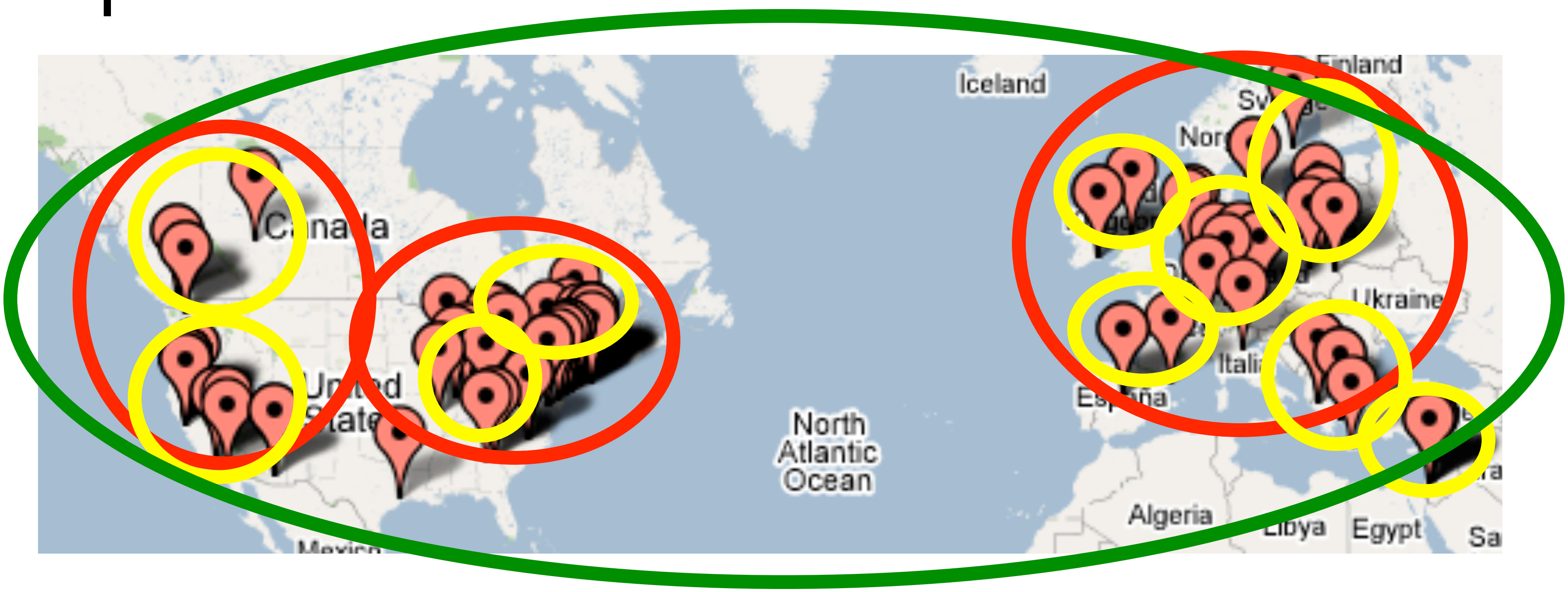
- Nodes aggregate request rate: ~12 million / min
- Rate-limit per node (β): 12 / min
- Requests at closest fan-in from 7 others: 83 / min

Next…



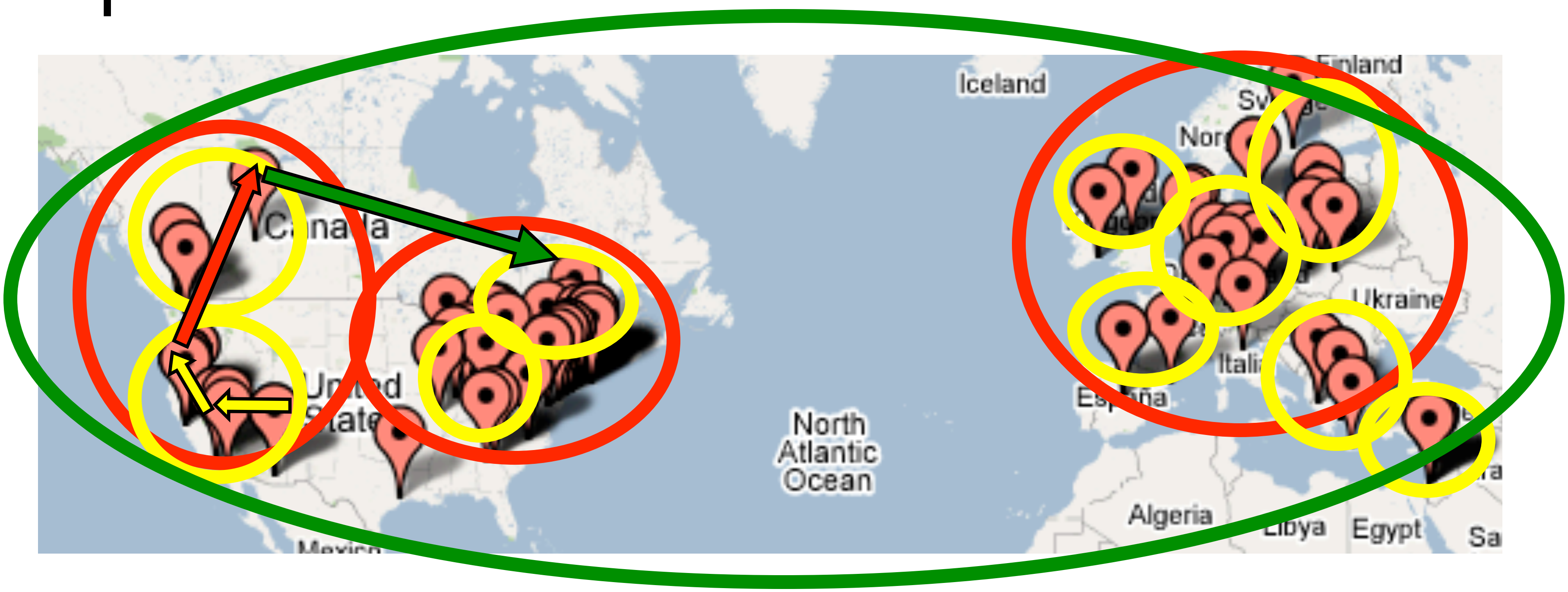
- Solution: Bound request rate to prevent hotspots
- Solution: Take advantage of network locality

Cluster by network proximity



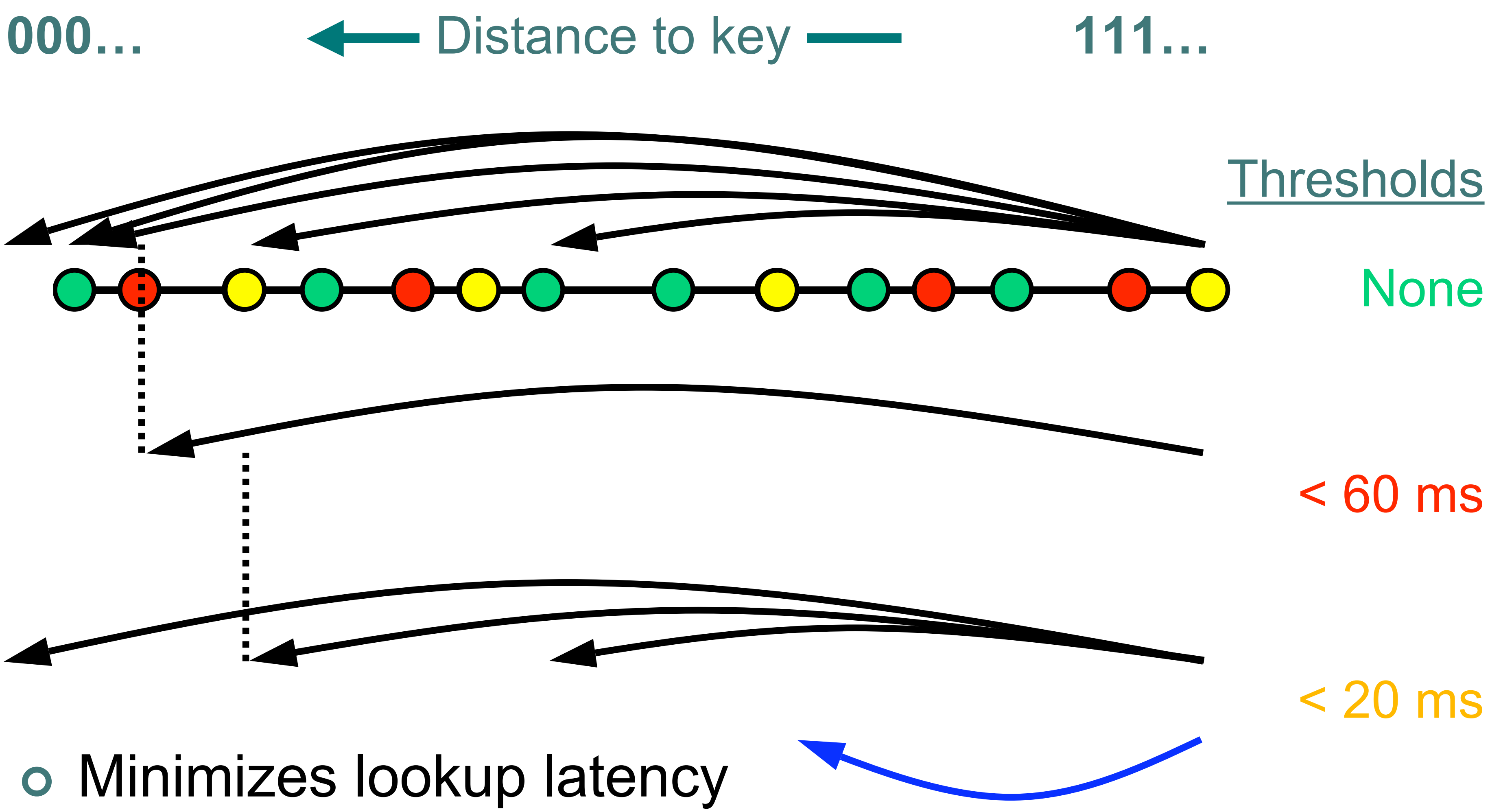
- Organically cluster nodes based on RTT
- Hierarchy of clusters of expanding diameter
- Lookup traverses up hierarchy
  - Route to node nearest ID in each level

Cluster by network proximity



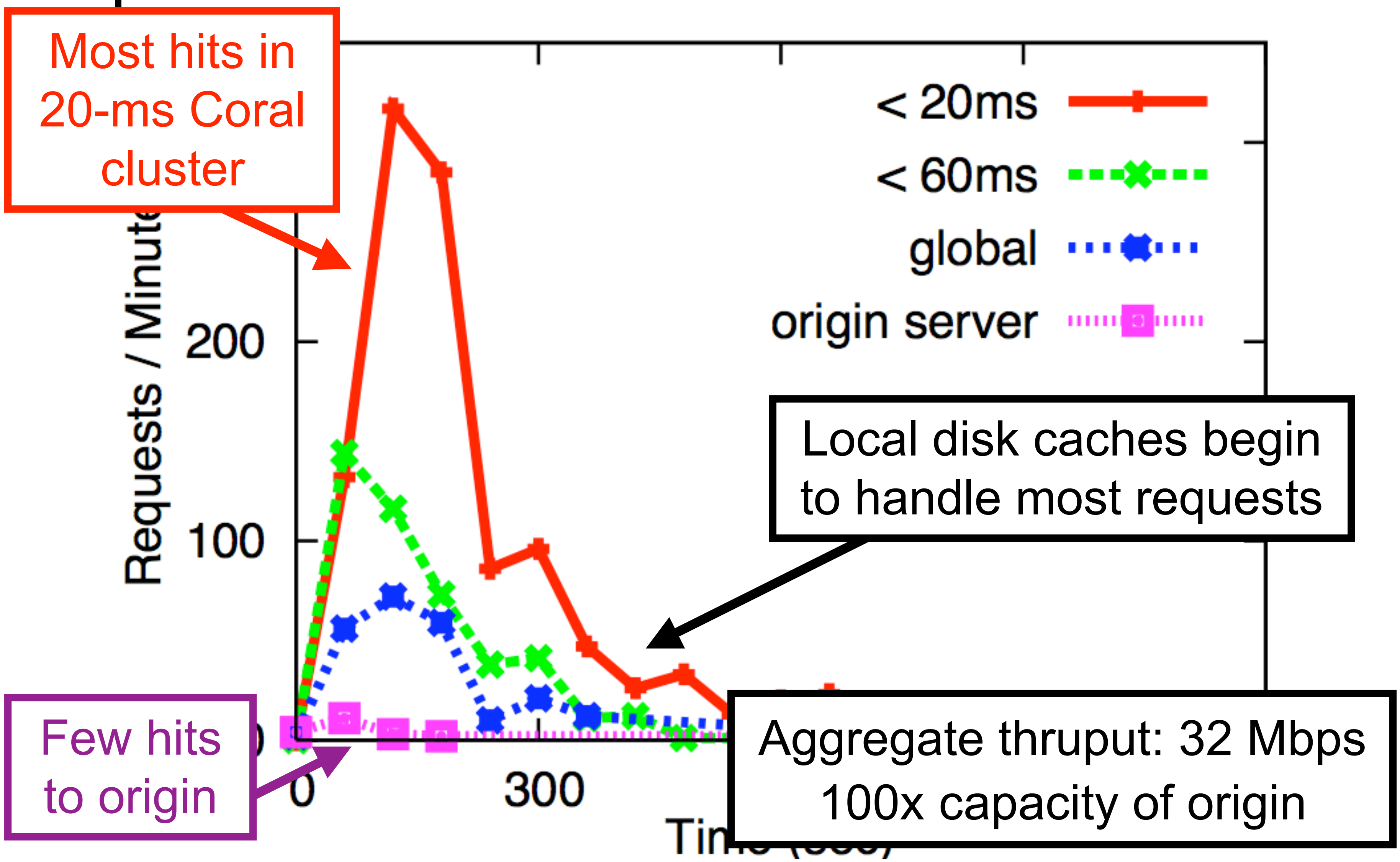
- Organically cluster nodes based on RTT
- Hierarchy of clusters of expanding diameter
- Lookup traverses up hierarchy
  - Route to node nearest ID in each level

Preserve locality through hierarchy

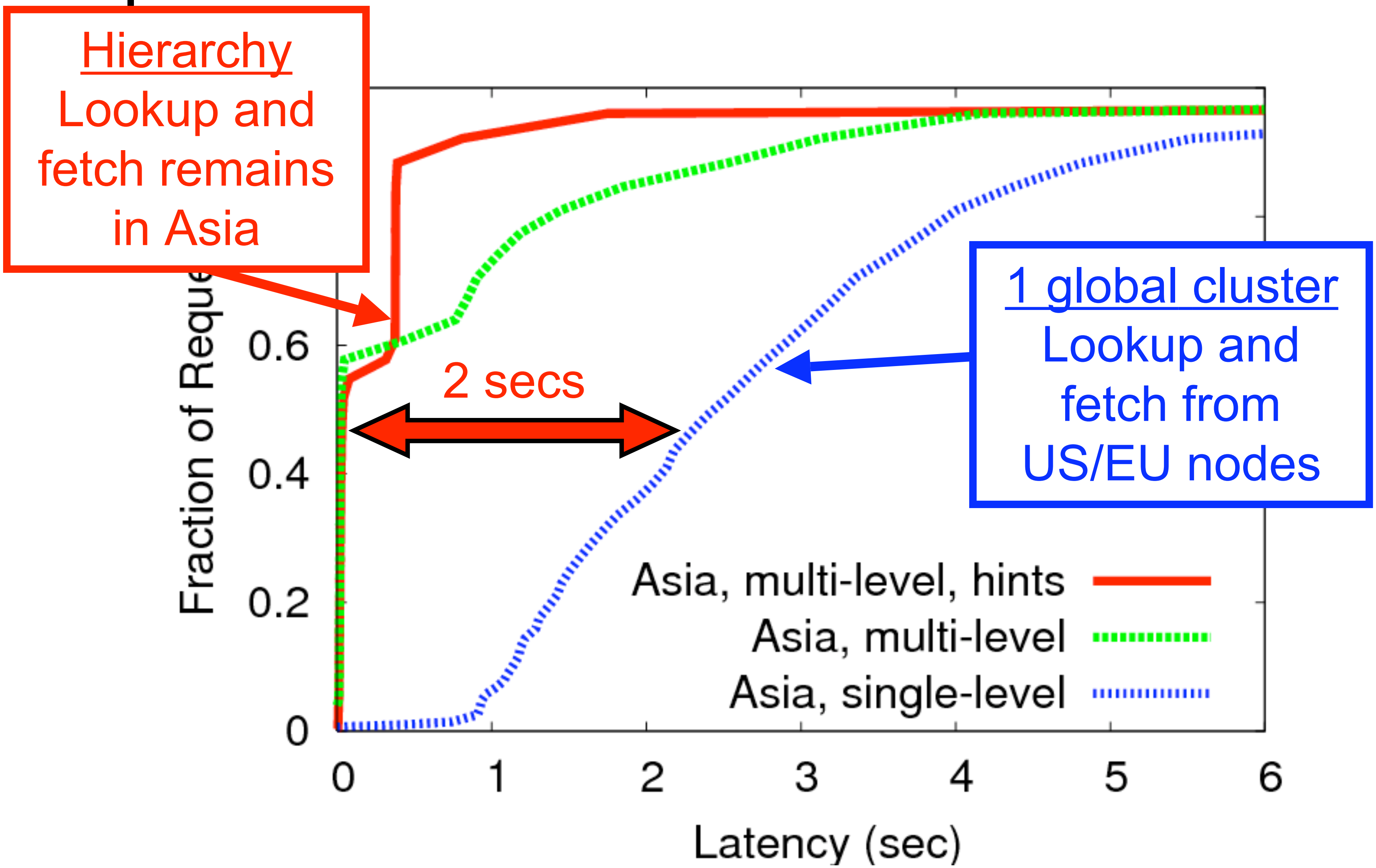


- Minimizes lookup latency
- Prefer values stored by nodes within faster clusters

Reduces load at origin server



Clustering benefits e2e latency



CoralCDN's deployment

Powered By PLANETLAB

- Deployed on 300-400 PlanetLab servers
- Running 24 / 7 since March 2004

Current daily usage

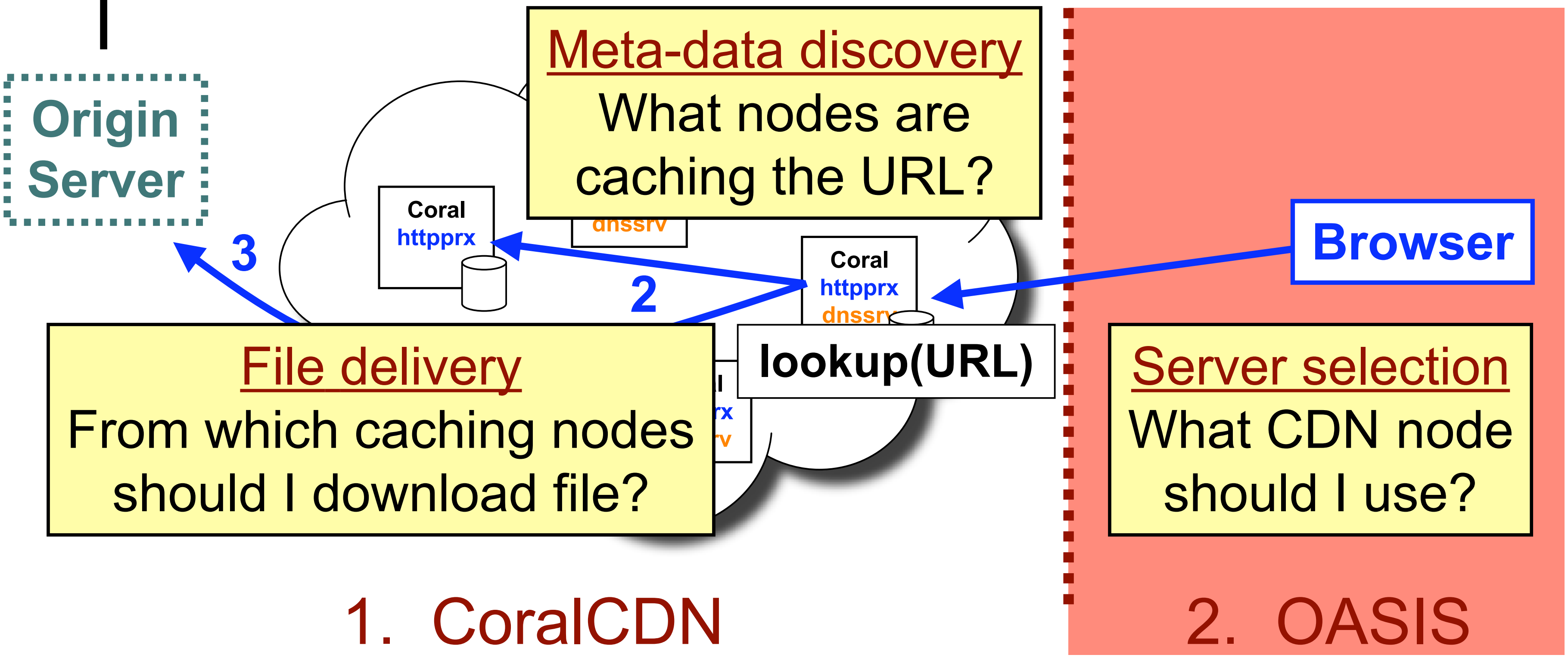
- 20-25 million HTTP requests
- 1-3 terabytes of data
- 1-2 million unique client IPs
- 20K-100K unique servers contacted (Zipf distribution)

- Varied usage
  - Servers to withstand high demand
  - Portals such as Slashdot, digg, …
  - Clients to avoid overloaded servers or censorship

This talk

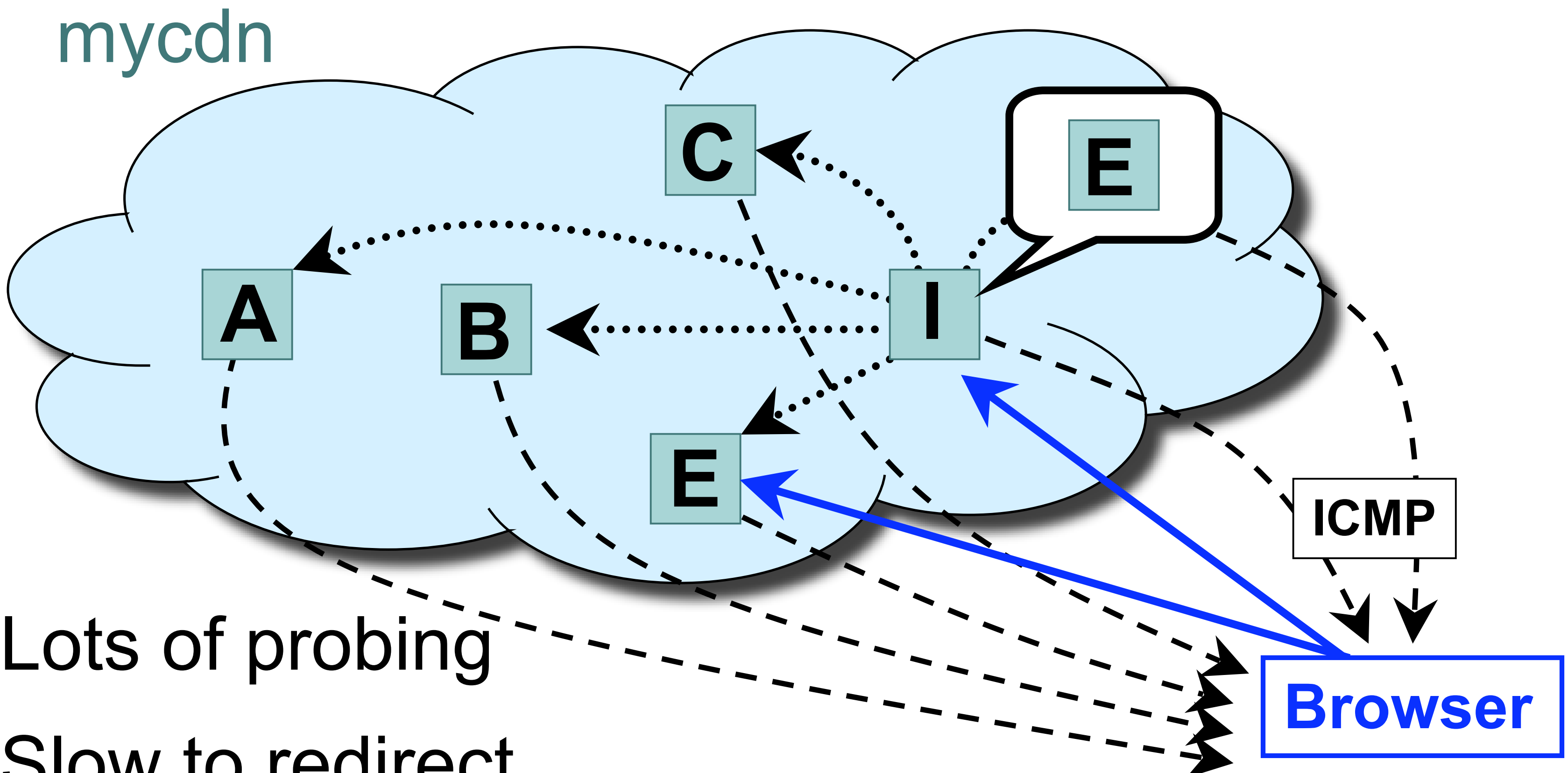
[IPTPS '03]
[NSDI '04]

[NSDI '06]



3. Using these for measurements: Illuminati [NSDI '07]
4. Finally, adding security to leverage more volunteers

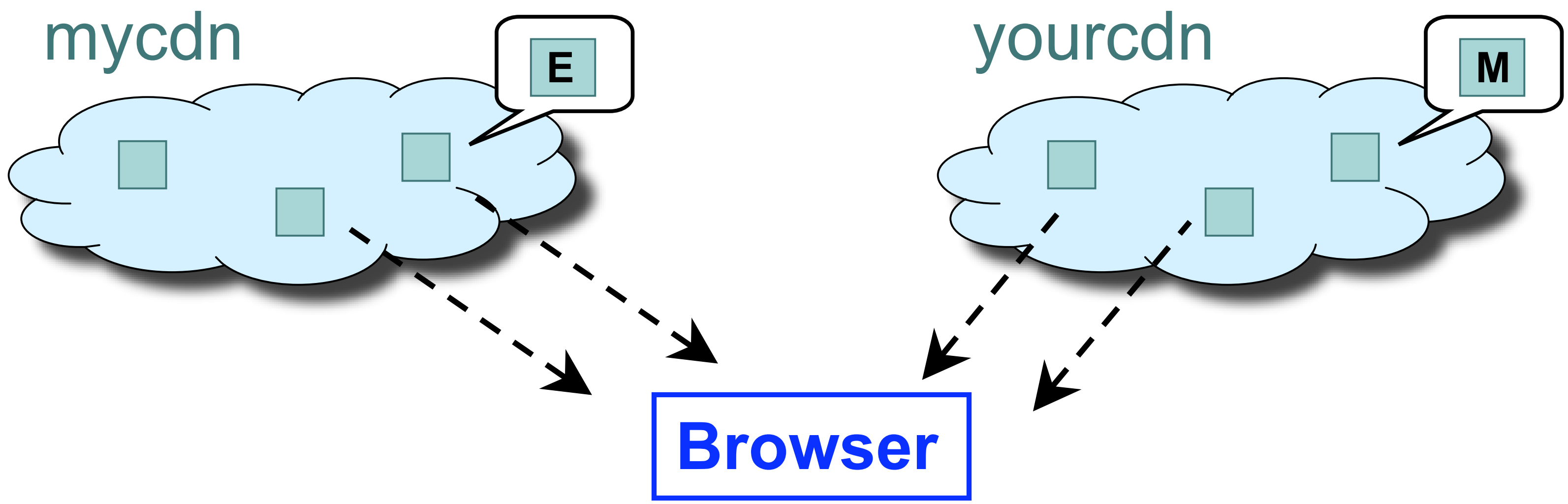
Strawman: probe to find nearest



- Lots of probing
- Slow to redirect
  - Negates goal of faster e2e download

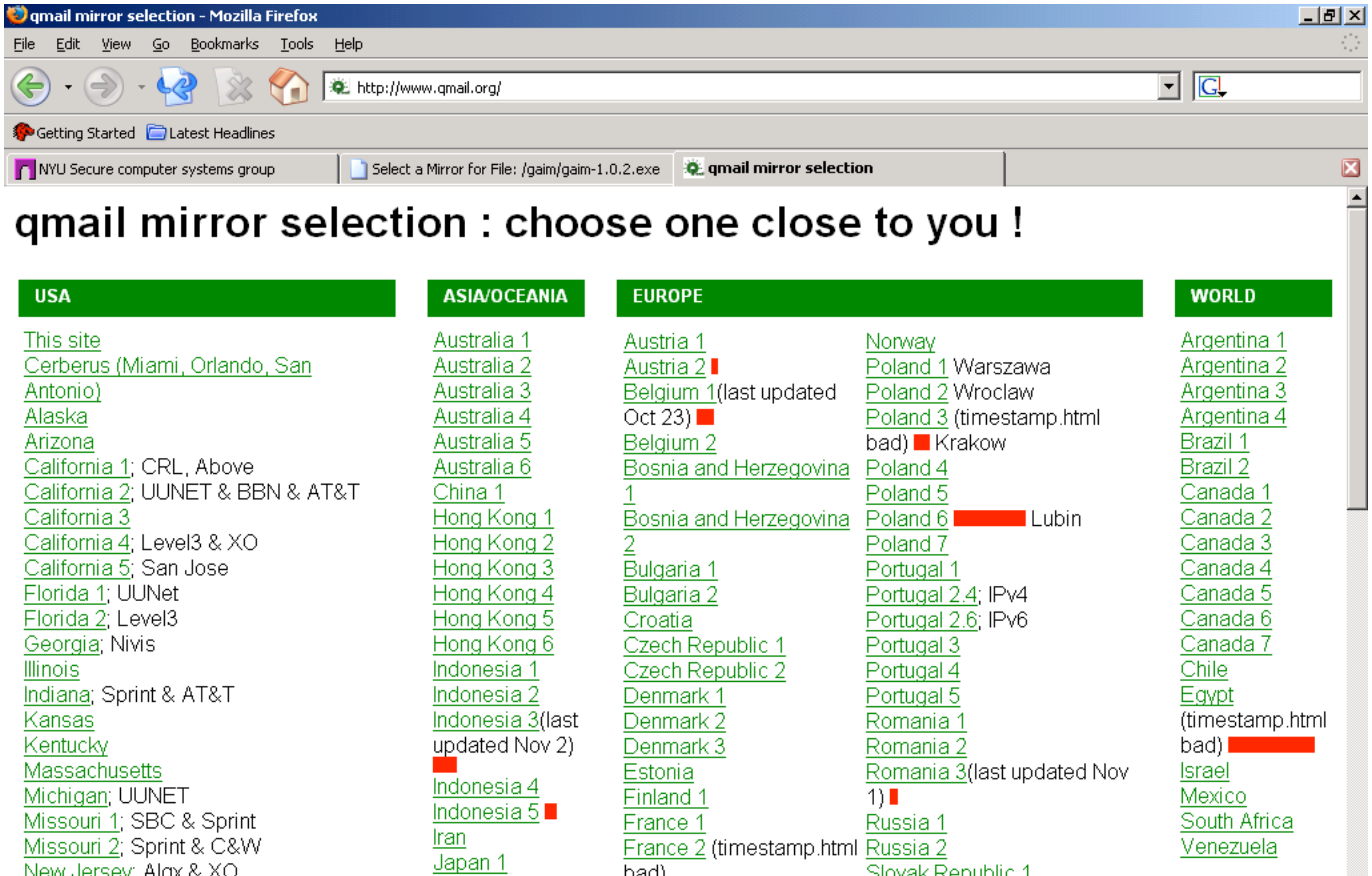
⇒ Cache after first lookup?

What about *yourcdn*?

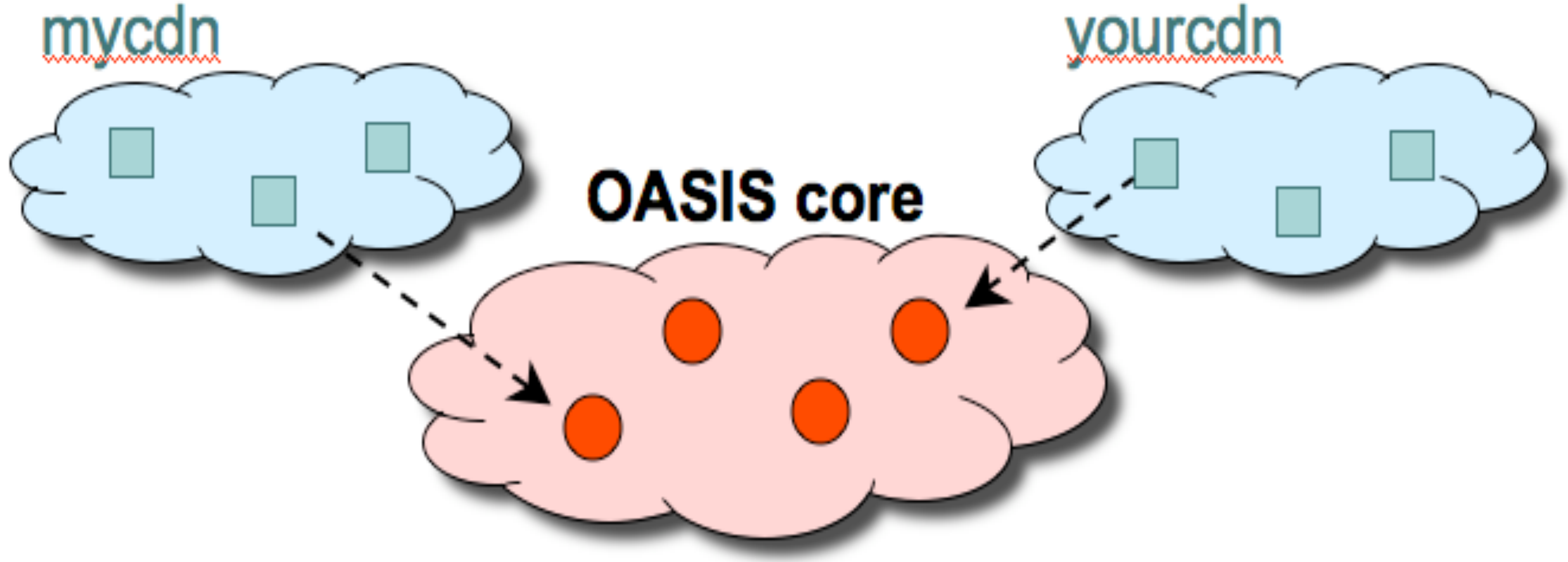


- Lots of probing
- Slow to redirect
- Every service pays same cost

Whither server-selection?

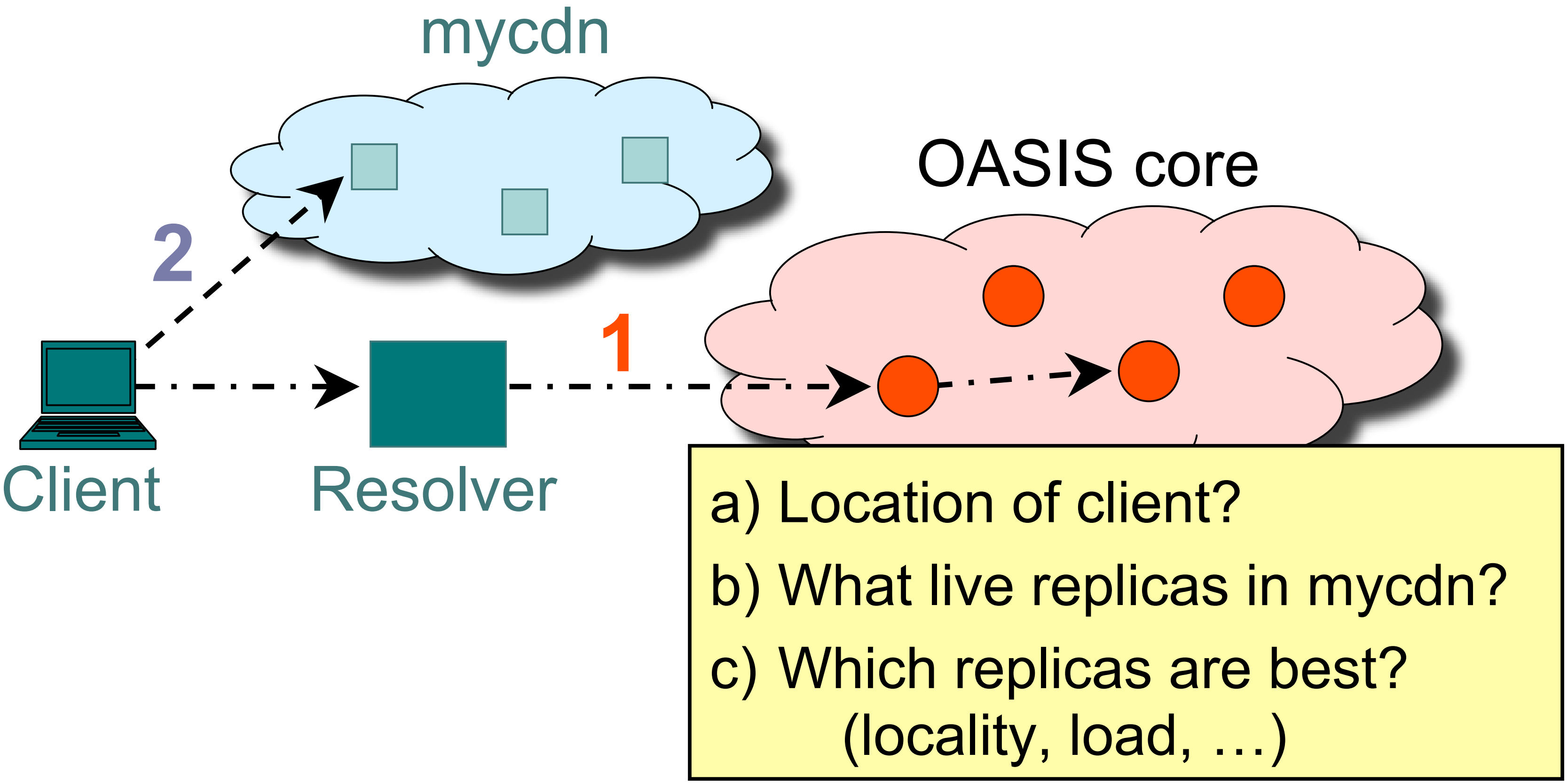


Goal: Knew answer without probing on critical path



- OASIS: a shared server-selection infrastructure
  - Amortize measurement cost over services' replicas
    - Total of ~20 GB/week, not per service
    - More nodes ⇒ higher accuracy and lower cost each
  - In turn, services benefit from functionality

If had a server-selection infrastructure…



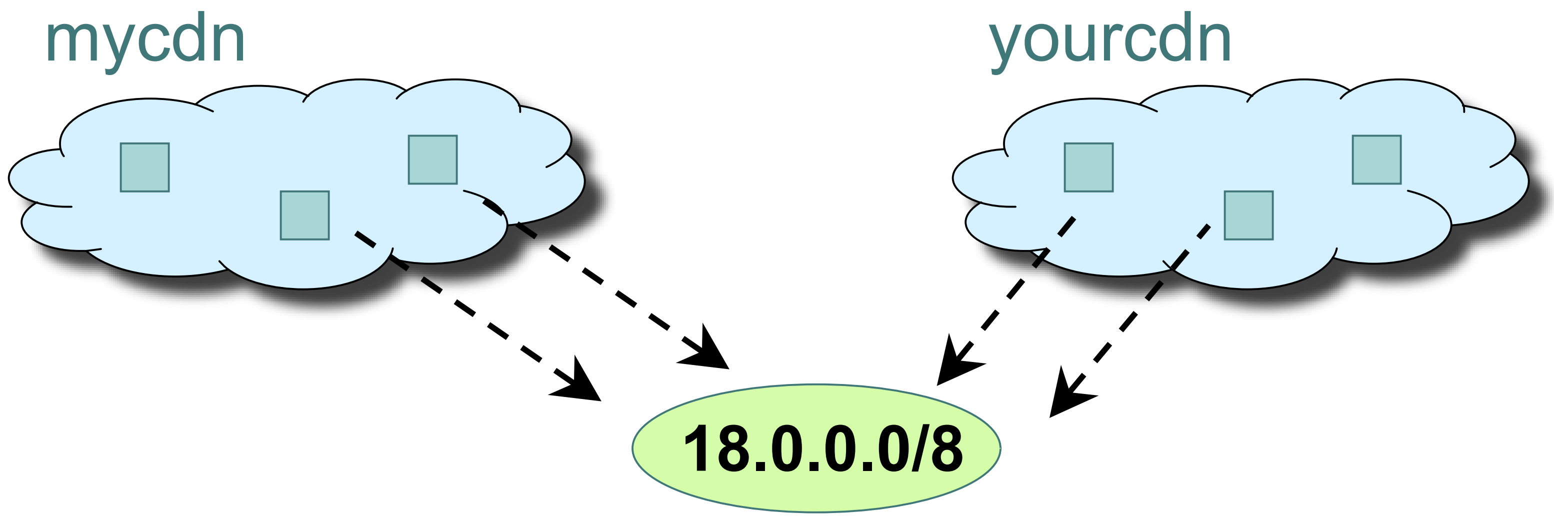
1. Client issues DNS request for ***mycdn**.nyuld.net*
2. OASIS redirects client to nearby application replica

What would this require?

- Measure the entire Internet in advance
  - Reduce the state space
  - Intermediate representation for locality
  - Detect and filter out measurement errors

- Architecture to organize nodes and manage data

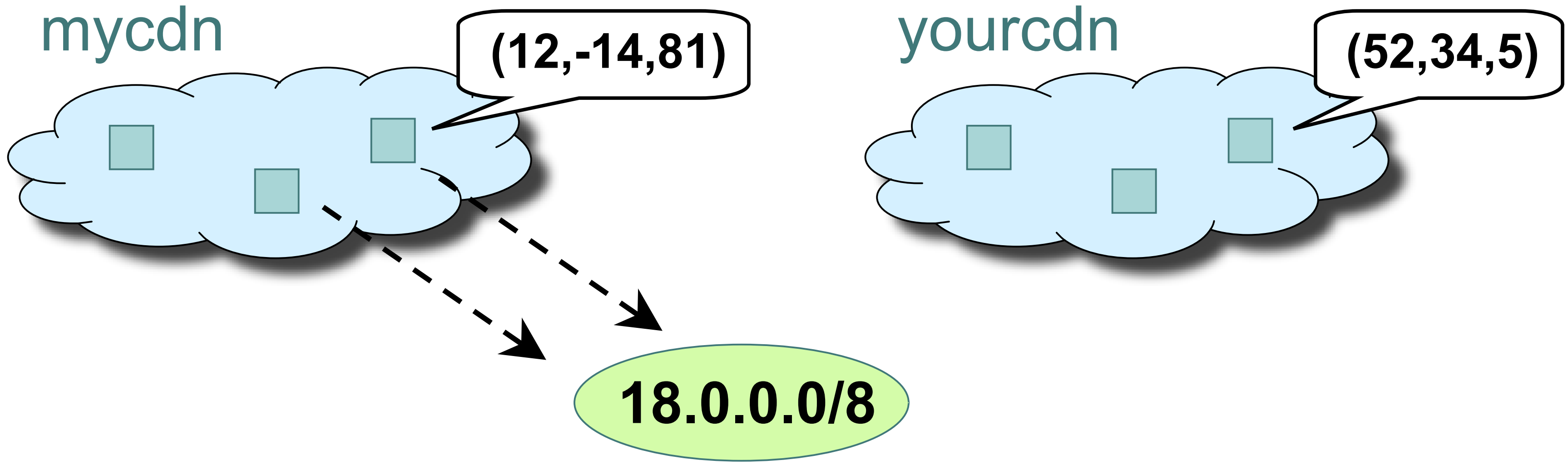
Reduce the state space



- 3-4 orders of magnitude by aggregating IP addresses
- [IMC '05]: nodes in same IP prefix are often close
  - 99% of prefixes with same first three-octets (x.y.z.*)
- Dynamically split prefixes until at same location

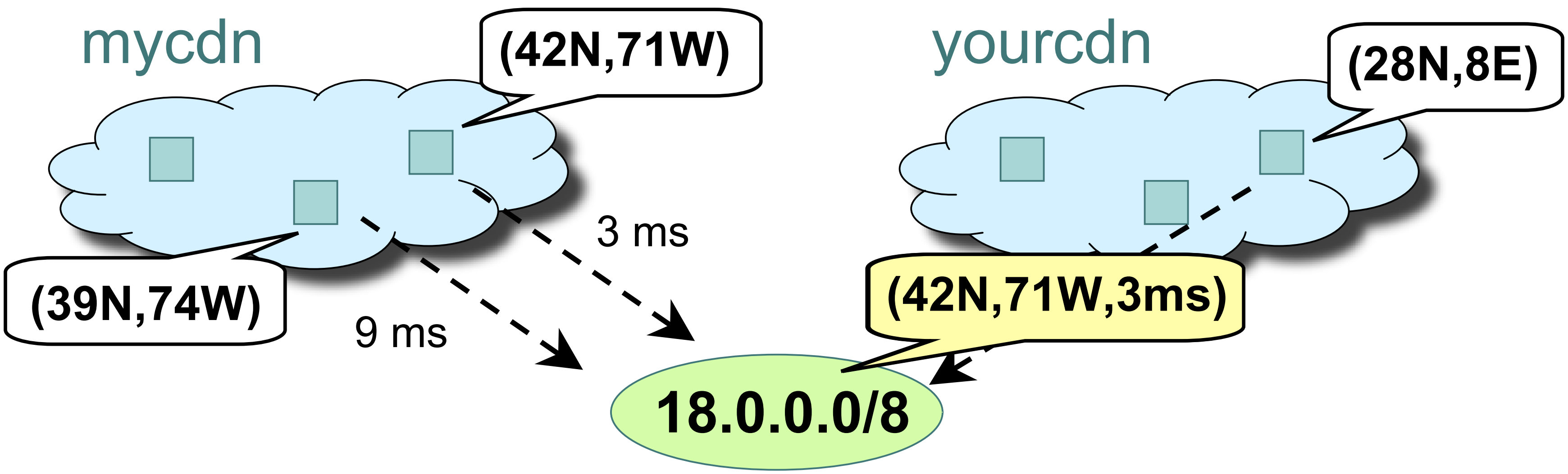
Representing locality

[IPTPS '05]



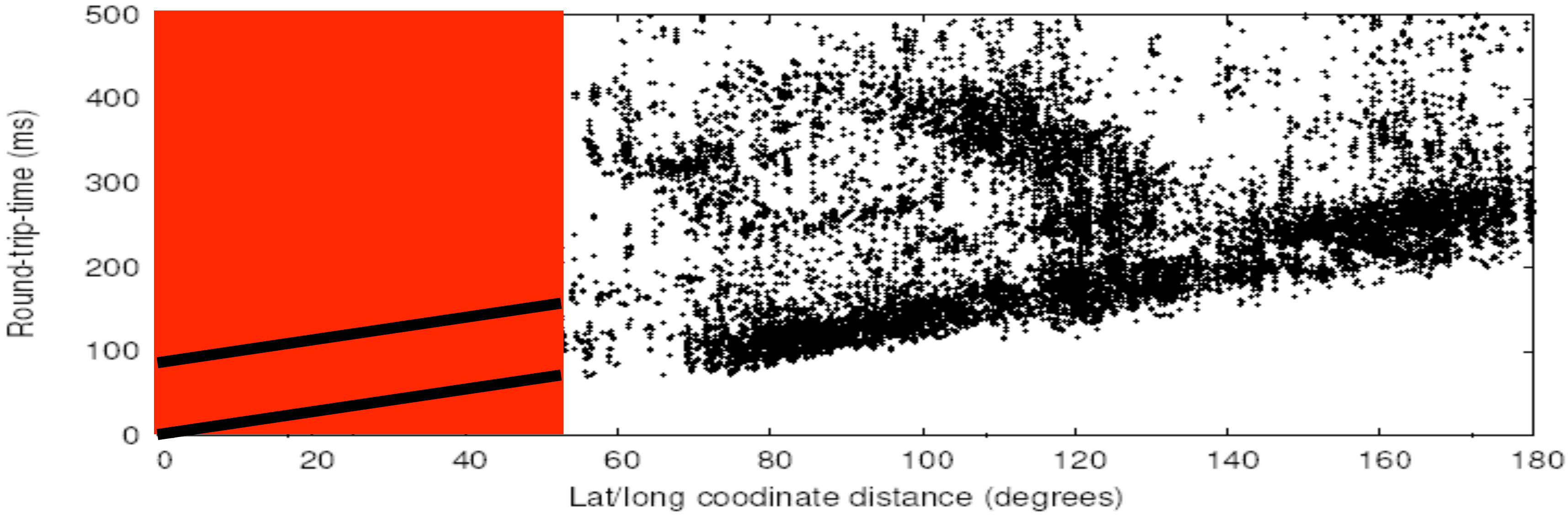
- Use virtual coordinates?
  - Predicts Internet latencies, fully decentralized
  - But designed for clients participating in protocol
  - Cached values useless: Coordinates drift over time

Representing locality



- Combine geographic coordinates with latency
  - Add't assumption: Replicas know own geo-coords
  - RTT accuracy has real-world meaning
    - Check if new coordinates improve accuracy

Representing locality

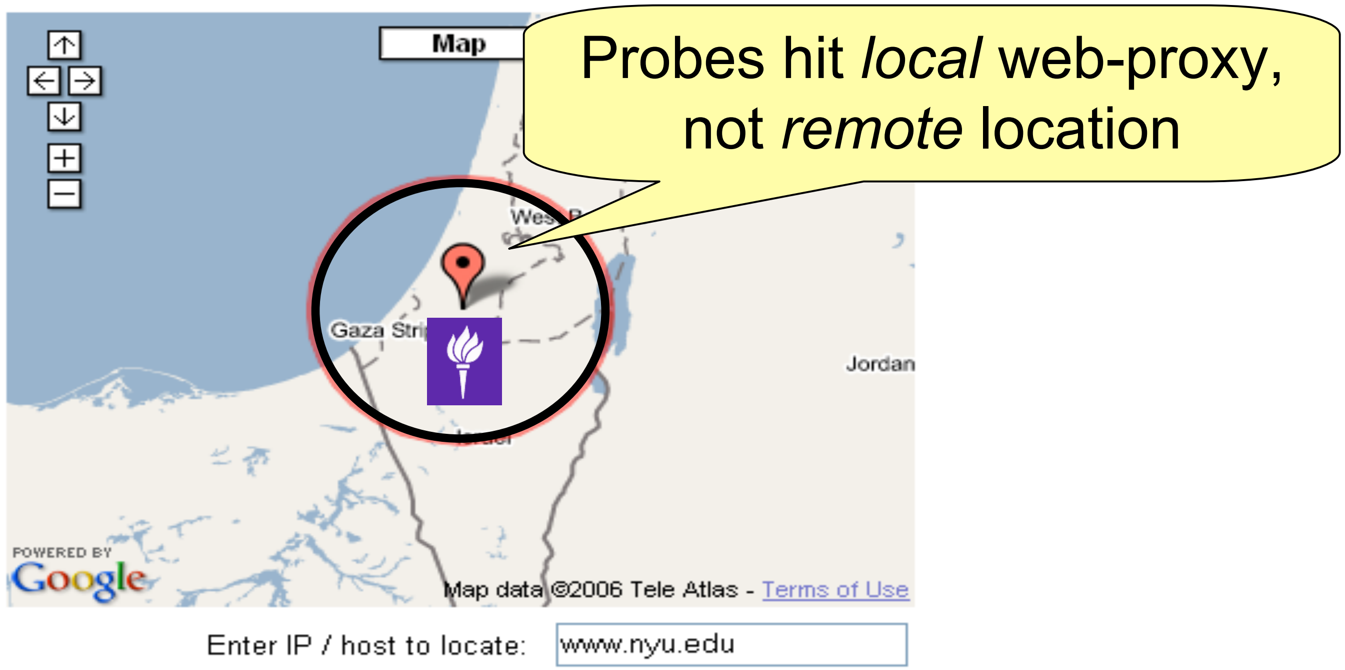


Correlation b/w geo-distance and RTT

Designing for high-density deployments

More nodes participate ⇄ Higher accuracy

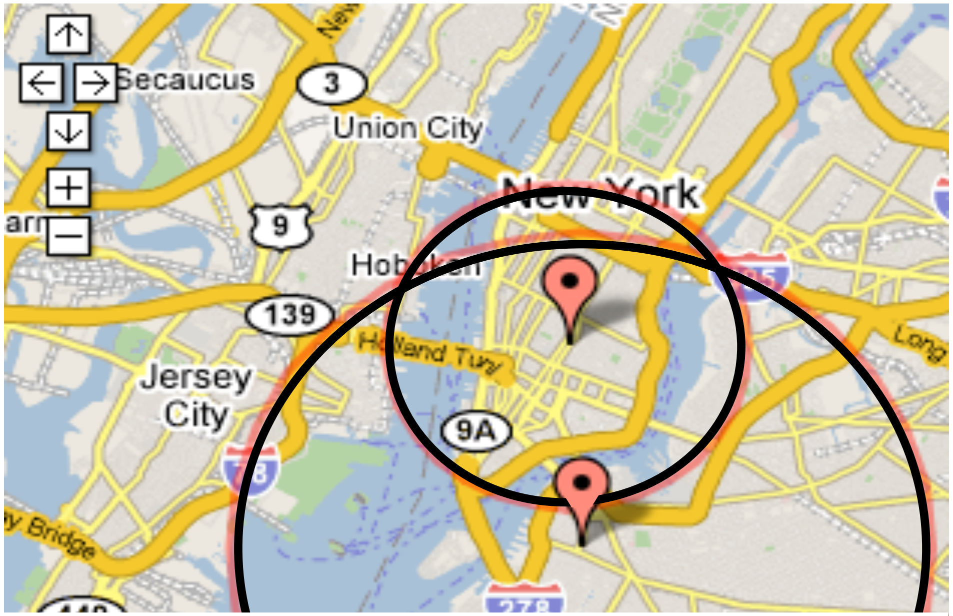
Measurements have errors



Israeli node 3 ms from NYU ?

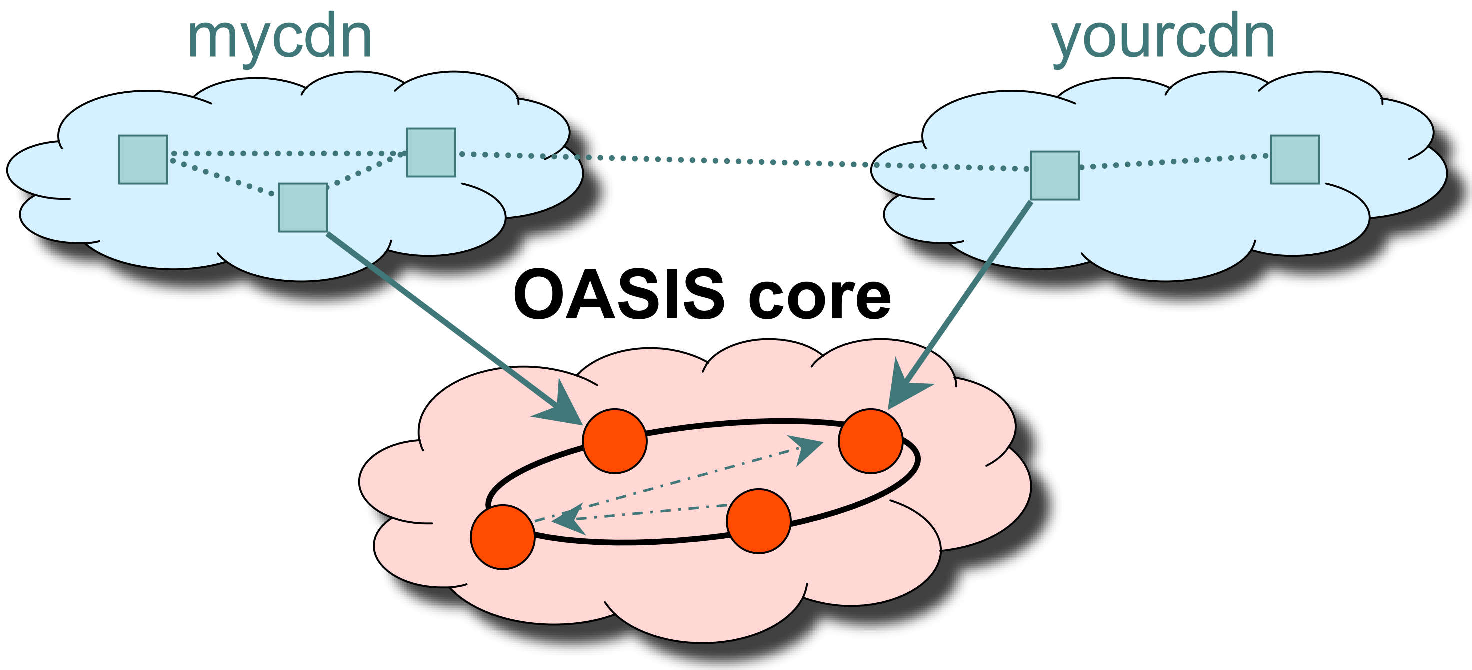
- Many conditions cause wildly wrong results
- Need general solution robust against errors

Finding measurement errors



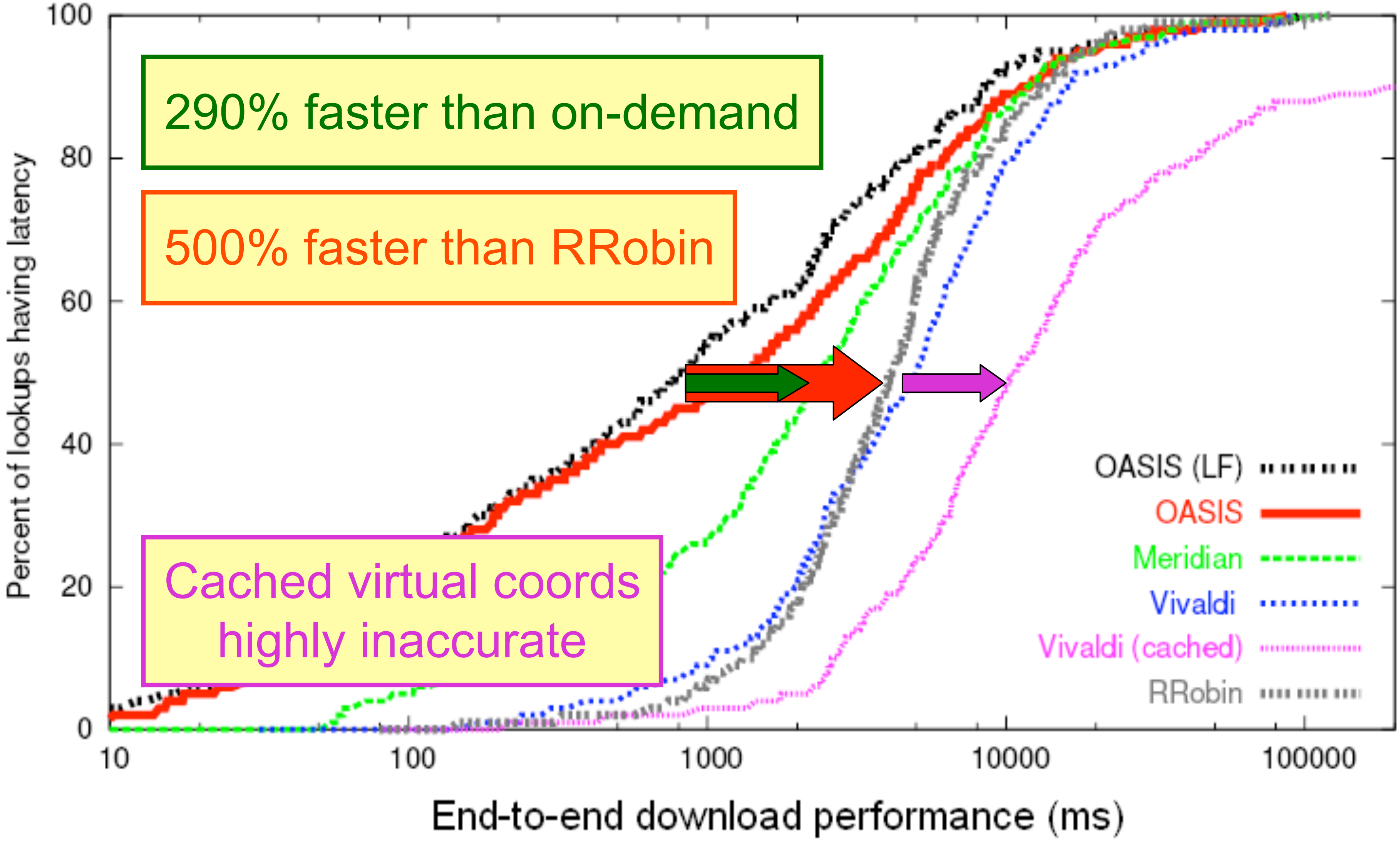
- Require measurement agreement
  - At least two results from different services must satisfy constraints (e.g., speed of light)

Engineering… (Lessons from Coral)



- OASIS core
  - Global membership view
  - Epidemic gossiping
    - Scalable failure detection
    - Replicate network map
  - Consistent hashing
    - Probing assignment, liveness of replicas
- Service replicas
  - Heartbeats to core
  - Meridian overlay for probing
    - $O(\log^2 n)$ probes finds closest

E2E download of web page



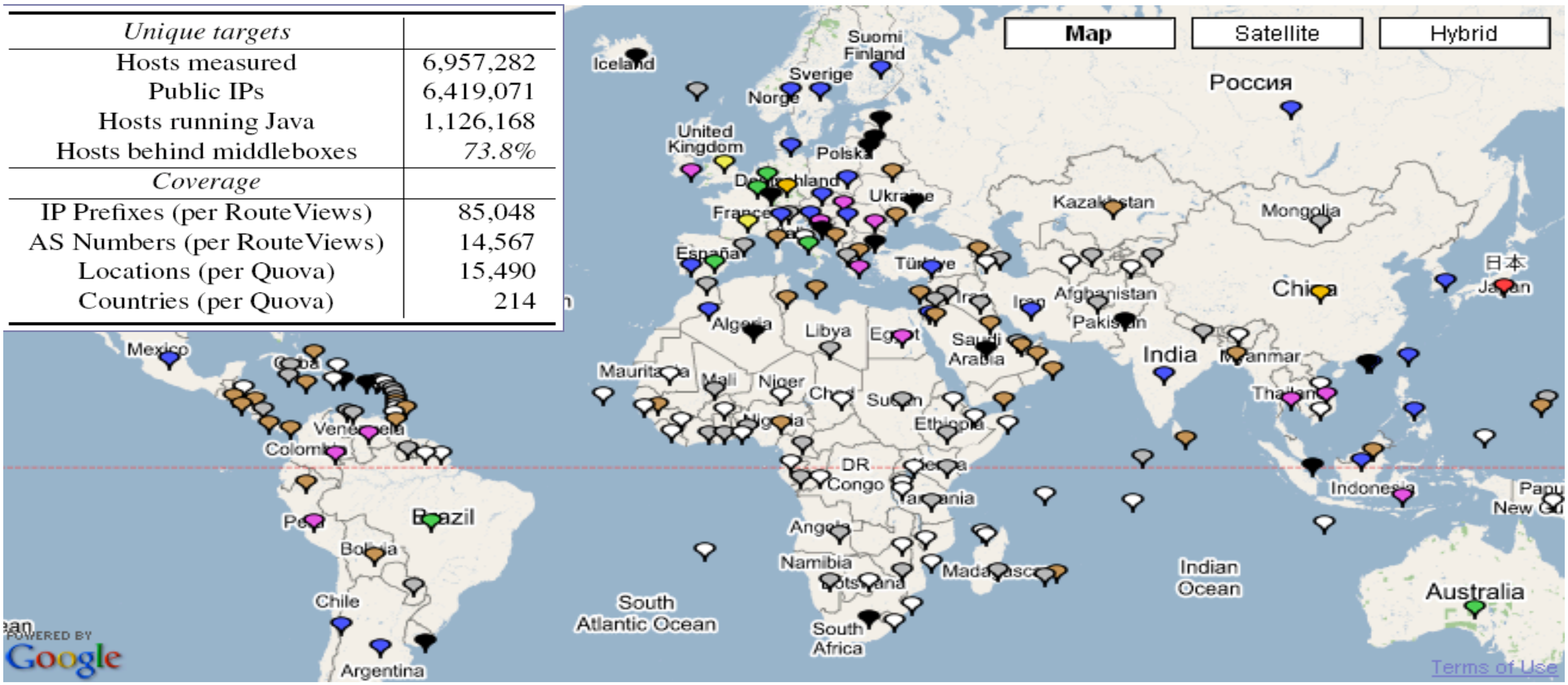
Deployed with thousands of replicas

- **AChord** topology-aware DHT (KAIST)
- **Chunkcast** block anycast (Berkeley)
- **CoralCDN** content distribution (NYU)
- **DONA** data-oriented network anycast (Berkeley)
- **Galaxy** distributed file system (Cincinnati)
- **Na Kika** content distribution (NYU)
- **OASIS:** RPC, DNS, HTTP interfaces
- **OCALA** overlay convergence (Berkeley)
- **OpenDHT** public DHT service (Berkeley)
- **OverCite** distributed library (MIT)
- **SlotNet** overlay routing (Purdue)

Systems as research platforms

- Measurements made possible by CoralCDN
  - Can’t probe clients behind middleboxes
  - CoralCDN clients execute active content

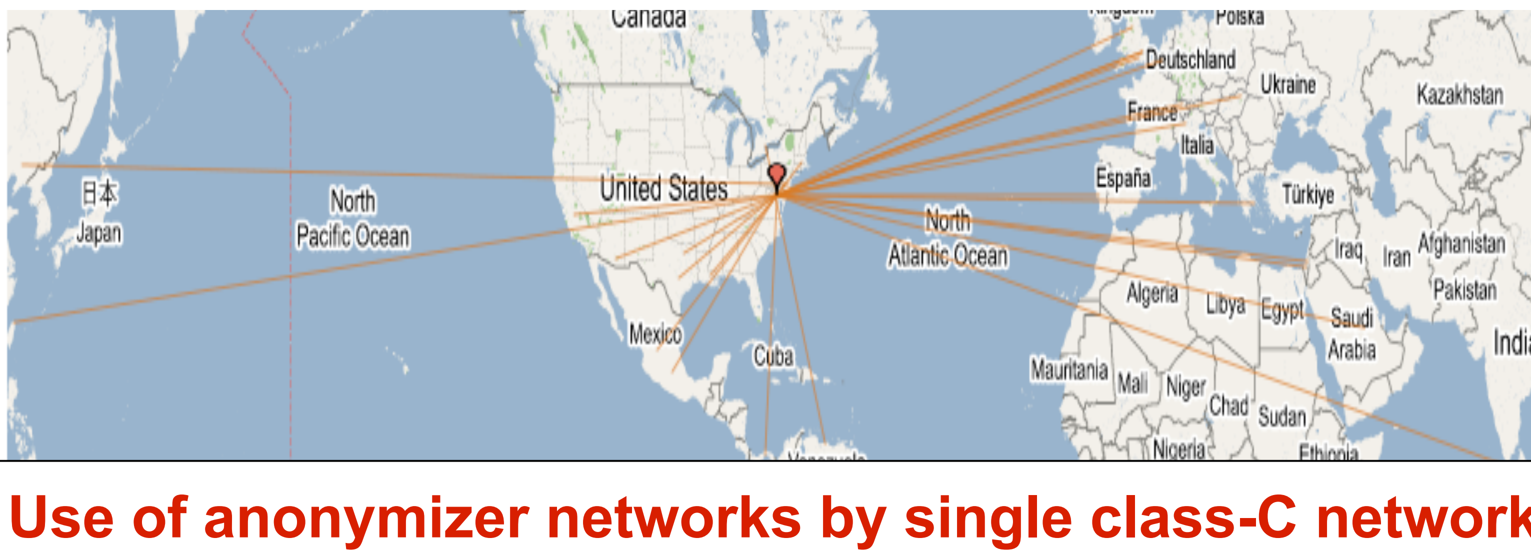
Unique targets	
Hosts measured	6,957,282
Public IPs	6,419,071
Hosts running Java	1,126,168
Hosts behind middleboxes	*73.8%*
Coverage	
IP Prefixes (per RouteViews)	85,048
AS Numbers (per RouteViews)	14,567
Locations (per Quova)	15,490
Countries (per Quova)	214



Measuring the edge: illuminati

[NSDI '07]

- DNS redirection: Clients near their nameservers?
  - Mostly within 20ms; diminishing returns to super-optimize
- Client blacklisting: Safe to blacklist an IP?
  - Quantify collatoral damage: NATs small, DHCP slow
- Client geolocation: Where are clients truly located?
  - Product for real-time proxy detection with Quova



Use of anonymizer networks by single class-C network

Security too…

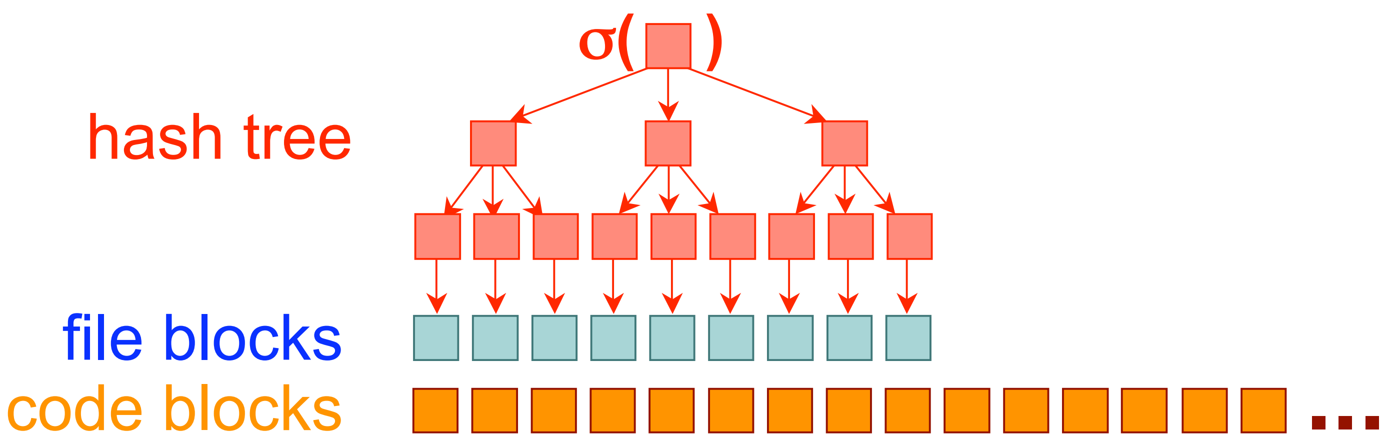
Theme throughout talk: How to leverage previously untapped resources to gain new functionality

- Cooperative content distribution
  - Locate and deliver cached content ⇒ CoralCDN
  - Select good servers ⇒ OASIS
- Adding security enables *untrusted* resources
  - Shark: scaling distributed file systems [NSDI '06]
    - Mutually-distrustful clients use each others' file caches

[S&P '04]

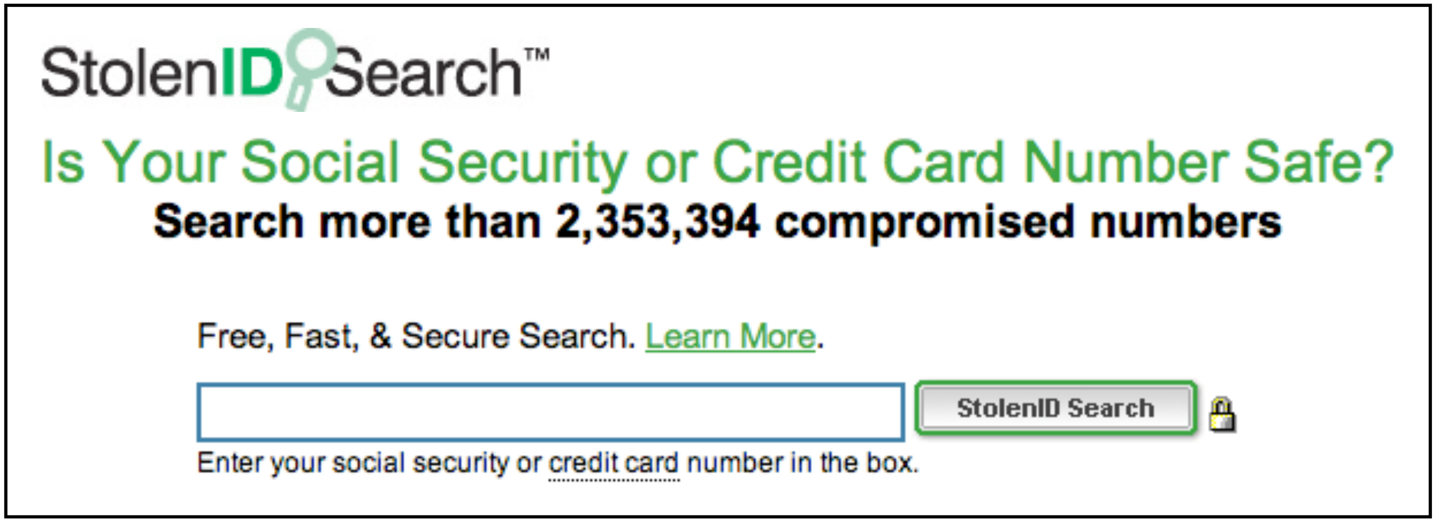
Large-file delivery via rateless erasure codes

- Encode blocks of large file, block negotiation unneeded
  - Exponential number of potential code blocks
- Prevents traditional hash trees for verification



- Instead, hashing based on homomorphic accumulator
  - Given $h(f_1)$, $h(f_2)$, $c_{1+2} = f_1+f_2$, compute $h(c_{1+2}) = h(f_1) \cdot h(f_2)$
- By batching PK operations, can verify at 60 Mbps

Need not be security *or* functionality



- Private matching (PM) [EUROCRYPT '04]
  - Parties compute set intersection (oblivious polynomials)

    P encodes x_i's $\rightleftarrows$ $\forall y_i,\ E(r_i P(y_i) + y_i)$ $\Rightarrow O(n \lg \lg n)$
  - e.g., Passenger manifests ∩ govt. no-fly lists [NSDI '06]
  - e.g., Social path in email correspondence for whitelisting
- Private keyword search (KS) [TCC '05]

Future: Securing and managing distributed systems

- Building and running large-scale systems difficult
  - Security, managability, reliability, scalability, …
  - Especially when decentralized, untrusted, …
  - Hard to reason about, hard to audit, hard to ensure QoS, …
- New architectures
  - Ethane: auditable, secure enterprise networks [Sec ‘06]
- New algorithms
  - Smaller groups with well-defined properties [IPTPS ‘06]
- New tools
  - Tracing transactions across hosts

Research approach

- Today:
  - Techniques for cooperative content distribution
  - Production use for 3 years, millions of users daily

- Generally:
  - New functionality through principled design
    - Distributed algorithms, cryptography, game theory, …
  - Build and deploy real systems
    - Evaluates design and leads to new problems
    - Hugely satisfying to have people use it

Thanks…

www.coralcdn.org

source code (GPL), data, papers available online