



# OASIS

Anycast for Any Service

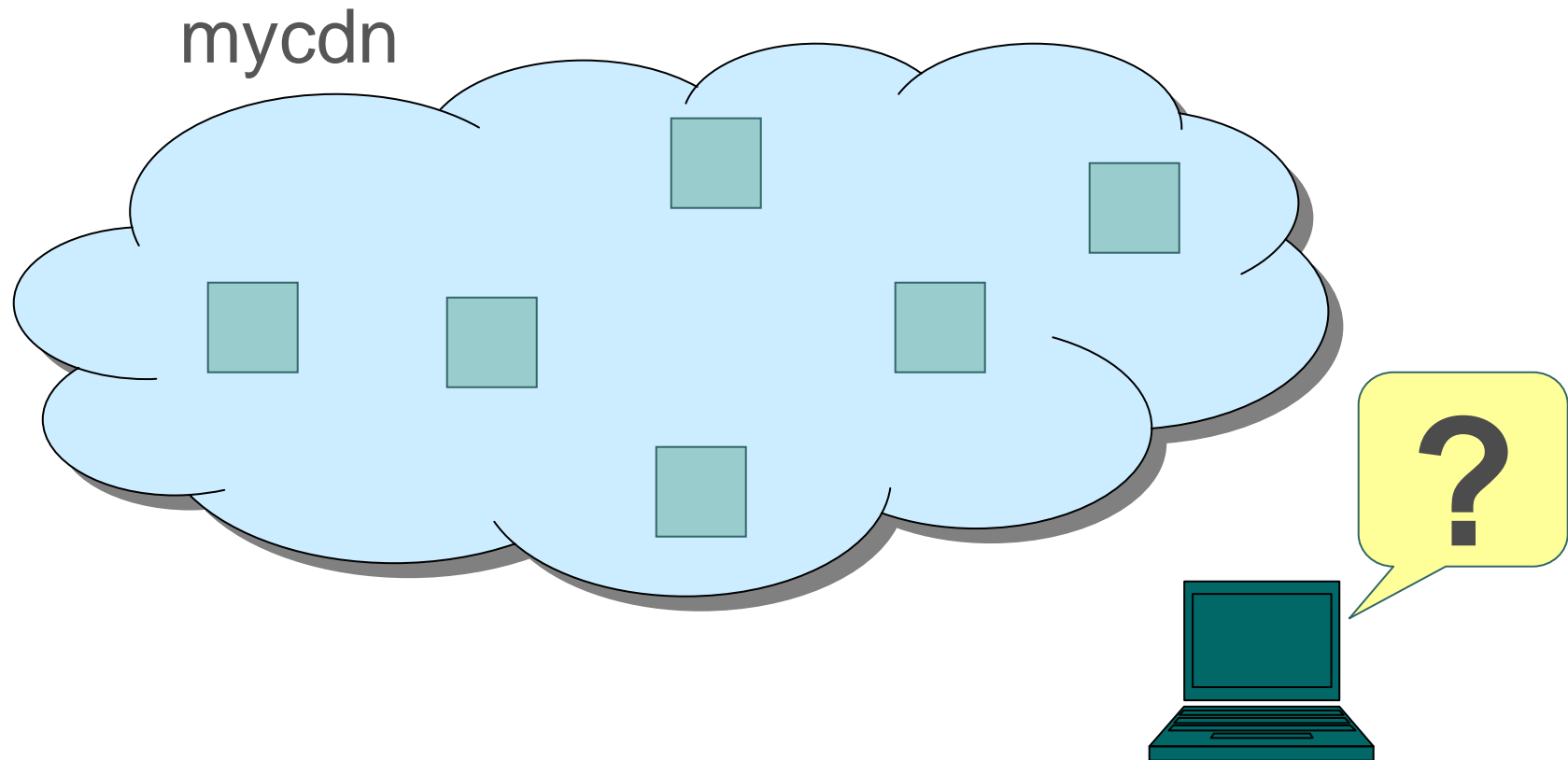
Michael J. Freedman

Karthik Lakshminarayanan

David Mazières

<http://oasis.coralcdn.org/>

- • • | What's the replica-selection problem?

















- Client needs to choose a “good” replica server
- Performance and cost dependent on replica selection

# What do we currently do?

Work for what you are worth \$

SOURCEFORGE.NET  
**DOWNLOAD  
SERVER**

You are requesting file: /gaim/gaim-1.0.2.exe  
Please select a mirror

Host	Location	Continent	Download
	Brussels, Belgium	Europe	 5932 kb
	Duesseldorf, Germany	Europe	 5932 kb
	Paris, France	Europe	 5932 kb
	Bern, Switzerland	Europe	 5932 kb
	Sydney, Australia	Australia	 5932 kb
	Dublin, Ireland	Europe	 5932 kb
	Minneapolis, MN	North America	 5932 kb

Done

Start Select a Mirror for Fil... 93% 9:59 AM

# How bad can it get?

gmail mirror selection - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.gmail.org/

Getting Started Latest Headlines

NYU Secure computer systems group Select a Mirror for File: /gaim/gaim-1.0.2.exe gmail mirror selection

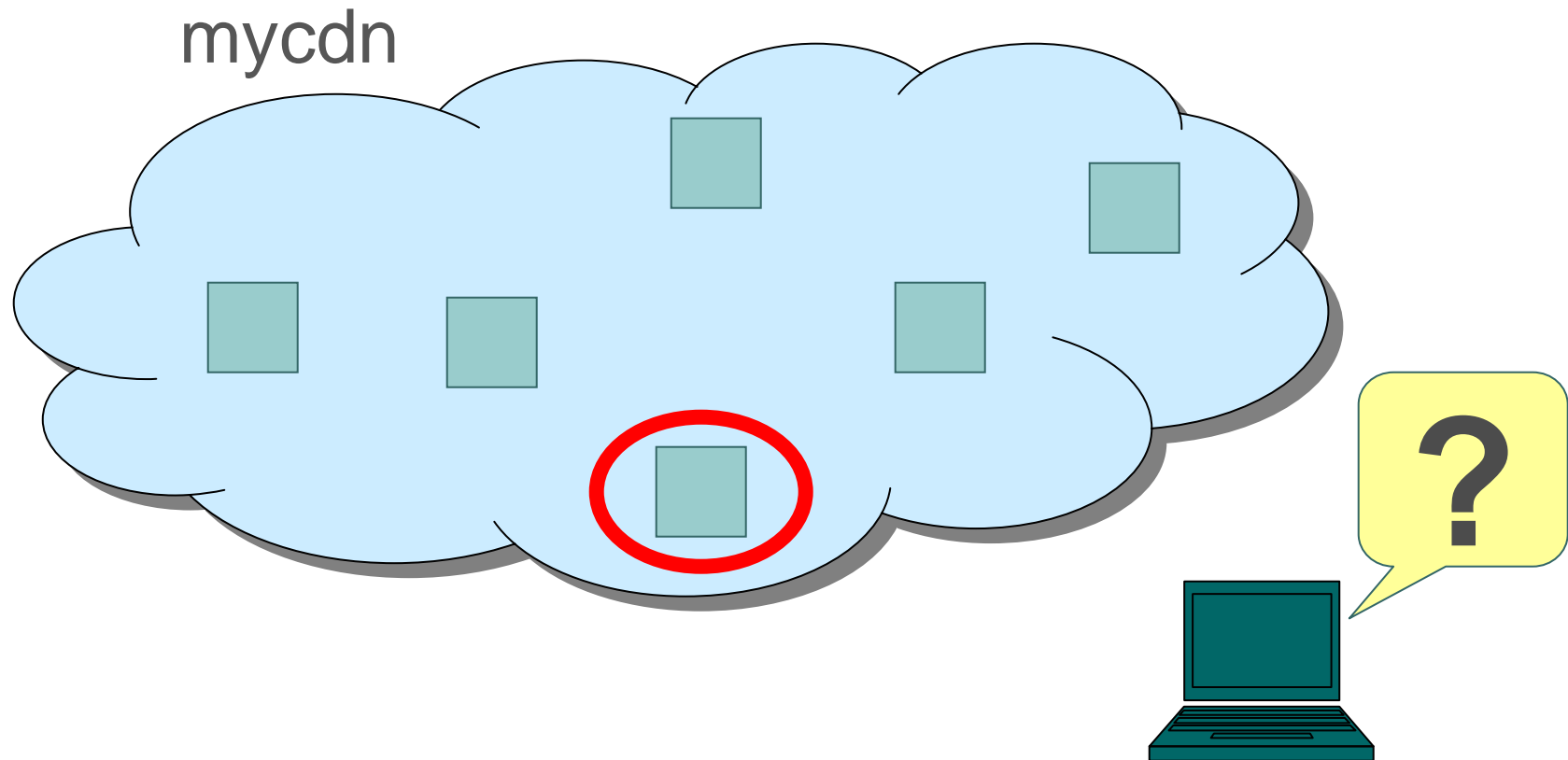
## gmail mirror selection : choose one close to you !

USA	ASIA/OCEANIA	EUROPE	WORLD	
<a href="#">This site</a>	<a href="#">Australia 1</a>	<a href="#">Austria 1</a>	<a href="#">Norway</a>	<a href="#">Argentina 1</a>
<a href="#">Cerberus (Miami, Orlando, San Antonio)</a>	<a href="#">Australia 2</a>	<a href="#">Austria 2</a>	<a href="#">Poland 1 Warszawa</a>	<a href="#">Argentina 2</a>
<a href="#">Alaska</a>	<a href="#">Australia 3</a>	<a href="#">Belgium 1 (last updated Oct 23)</a>	<a href="#">Poland 2 Wroclaw</a>	<a href="#">Argentina 3</a>
<a href="#">Arizona</a>	<a href="#">Australia 4</a>	<a href="#">Belgium 2</a>	<a href="#">Poland 3 (timestamp.html bad)</a>	<a href="#">Argentina 4</a>
<a href="#">California 1; CRL, Above</a>	<a href="#">Australia 5</a>	<a href="#">Bosnia and Herzegovina 1</a>	<a href="#">Poland 4</a>	<a href="#">Brazil 1</a>
<a href="#">California 2; UUNET &amp; BBN &amp; AT&amp;T</a>	<a href="#">Australia 6</a>	<a href="#">Bosnia and Herzegovina 2</a>	<a href="#">Poland 5</a>	<a href="#">Brazil 2</a>
<a href="#">California 3</a>	<a href="#">China 1</a>	<a href="#">Bulgaria 1</a>	<a href="#">Poland 6</a>	<a href="#">Canada 1</a>
<a href="#">California 4; Level3 &amp; XO</a>	<a href="#">Hong Kong 1</a>	<a href="#">Croatia</a>	<a href="#">Poland 7</a>	<a href="#">Canada 2</a>
<a href="#">California 5; San Jose</a>	<a href="#">Hong Kong 2</a>	<a href="#">Czech Republic 1</a>	<a href="#">Portugal 1</a>	<a href="#">Canada 3</a>
<a href="#">Florida 1; UUNet</a>	<a href="#">Hong Kong 3</a>	<a href="#">Czech Republic 2</a>	<a href="#">Portugal 2.4; IPv4</a>	<a href="#">Canada 4</a>
<a href="#">Florida 2; Level3</a>	<a href="#">Hong Kong 4</a>	<a href="#">Denmark 1</a>	<a href="#">Portugal 2.6; IPv6</a>	<a href="#">Canada 5</a>
<a href="#">Georgia; Nivis</a>	<a href="#">Hong Kong 5</a>	<a href="#">Denmark 2</a>	<a href="#">Portugal 3</a>	<a href="#">Canada 6</a>
<a href="#">Illinois</a>	<a href="#">Hong Kong 6</a>	<a href="#">Denmark 3</a>	<a href="#">Portugal 4</a>	<a href="#">Canada 7</a>
<a href="#">Indiana; Sprint &amp; AT&amp;T</a>	<a href="#">Indonesia 1</a>	<a href="#">Estonia</a>	<a href="#">Portugal 5</a>	<a href="#">Chile</a>
<a href="#">Kansas</a>	<a href="#">Indonesia 2</a>	<a href="#">Finland 1</a>	<a href="#">Romania 1</a>	<a href="#">Egypt</a>
<a href="#">Kentucky</a>	<a href="#">Indonesia 3 (last updated Nov 2)</a>	<a href="#">France 1</a>	<a href="#">Romania 2</a>	<a href="#">(timestamp.html bad)</a>
<a href="#">Massachusetts</a>	<a href="#">Indonesia 4</a>	<a href="#">France 2 (timestamp.html bad)</a>	<a href="#">Romania 3 (last updated Nov 1)</a>	<a href="#">Israel</a>
<a href="#">Michigan; UUNET</a>	<a href="#">Indonesia 5</a>	<a href="#">France 3</a>	<a href="#">Russia 1</a>	<a href="#">Mexico</a>
<a href="#">Missouri 1; SBC &amp; Sprint</a>	<a href="#">Iran</a>	<a href="#">France 5</a>	<a href="#">Russia 2</a>	<a href="#">South Africa</a>
<a href="#">Missouri 2; Sprint &amp; C&amp;W</a>	<a href="#">Japan 1</a>	<a href="#">France 6</a>	<a href="#">Slovak Republic 1</a>	<a href="#">Venezuela</a>
<a href="#">New Jersey; Algx &amp; XO</a>	<a href="#">Japan 2</a>		<a href="#">Slovak Republic 2</a>	
<a href="#">North Carolina</a>	<a href="#">Japan 3</a>		<a href="#">Slovak Republic 2</a>	
<a href="#">New York; Sprint</a>	<a href="#">Japan 4</a>		<a href="#">Spain 1</a>	
<a href="#">New Jersey</a>				

Done

Start gmail mirror selectio... sourceforge.bmp - Paint 94% 10:01 AM

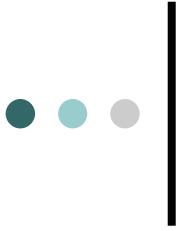
# ● ● ● | Anycast is the solution



- Anycast = automated “good” replica selection
- OASIS is a flexible anycast system for multiple services

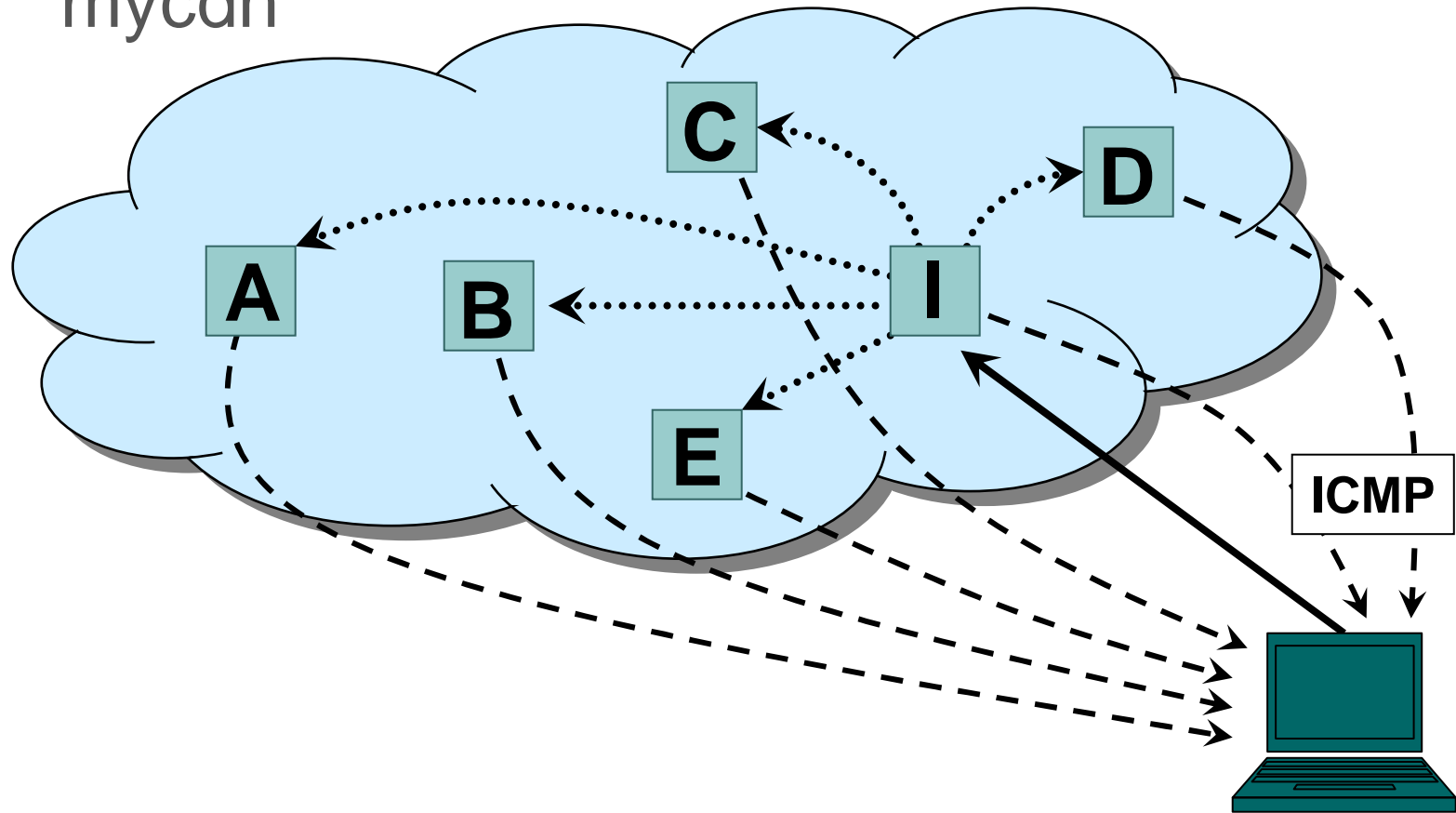
# ● ● ● | The need for anycast

- Internet systems rely on replicated content and services
  - Distributed mirrors: Web servers, FTP servers, ...
  - Content Distribution Networks: Akamai, CoralCDN, ...
  - Internet Naming Systems: DNS, SFR, DOA, ...
  - Distributed File Systems: CFS, Shark, ....
  - Routing Overlays: RON, Detour, i3, ...
  - Distributed Hash Storage Systems: OpenDHT, ...
- All could benefit from anycast service



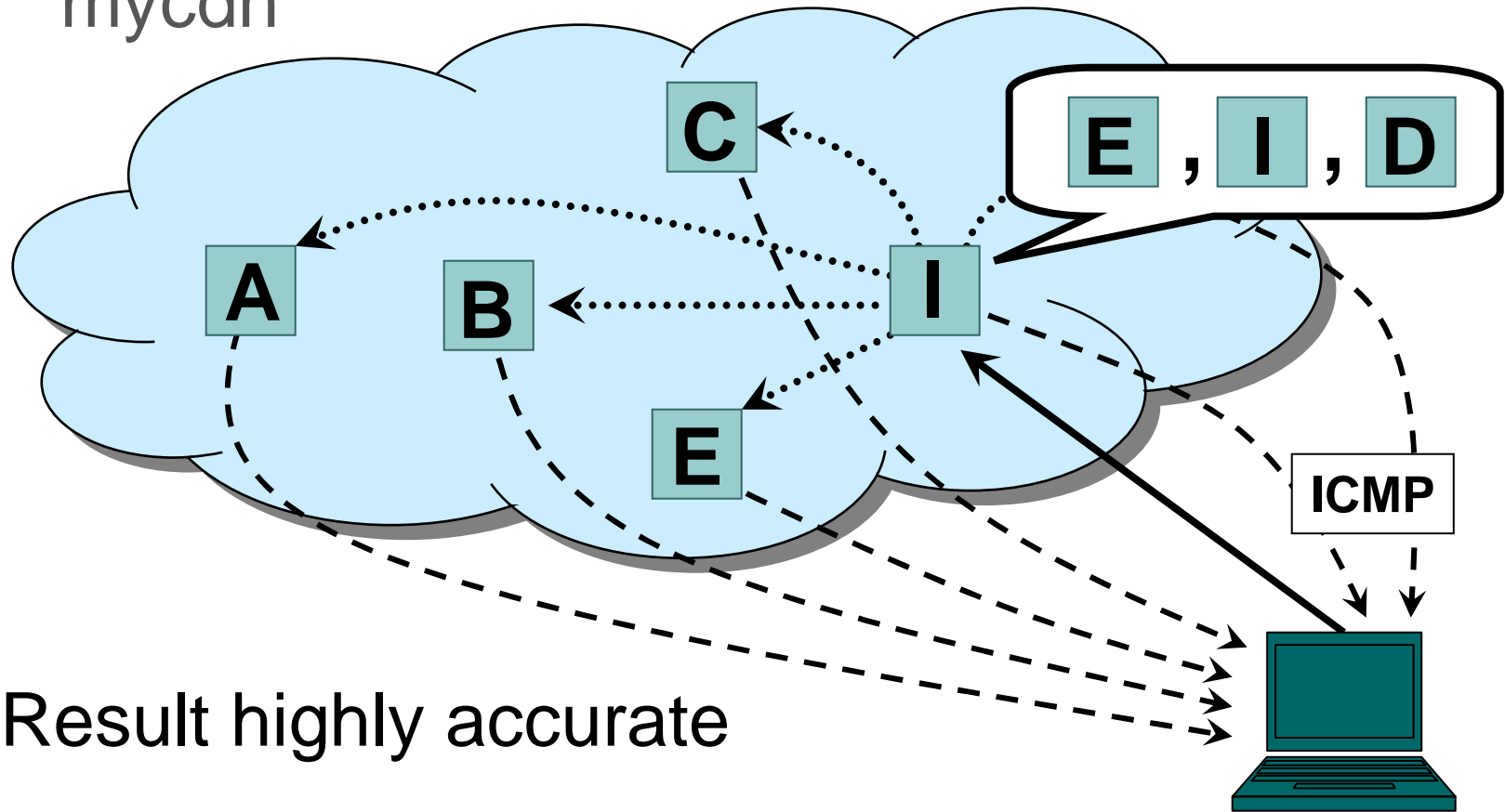
How should one  
implement anycast?

● ● ● | Strawman: probe & find nearest  
mycdn



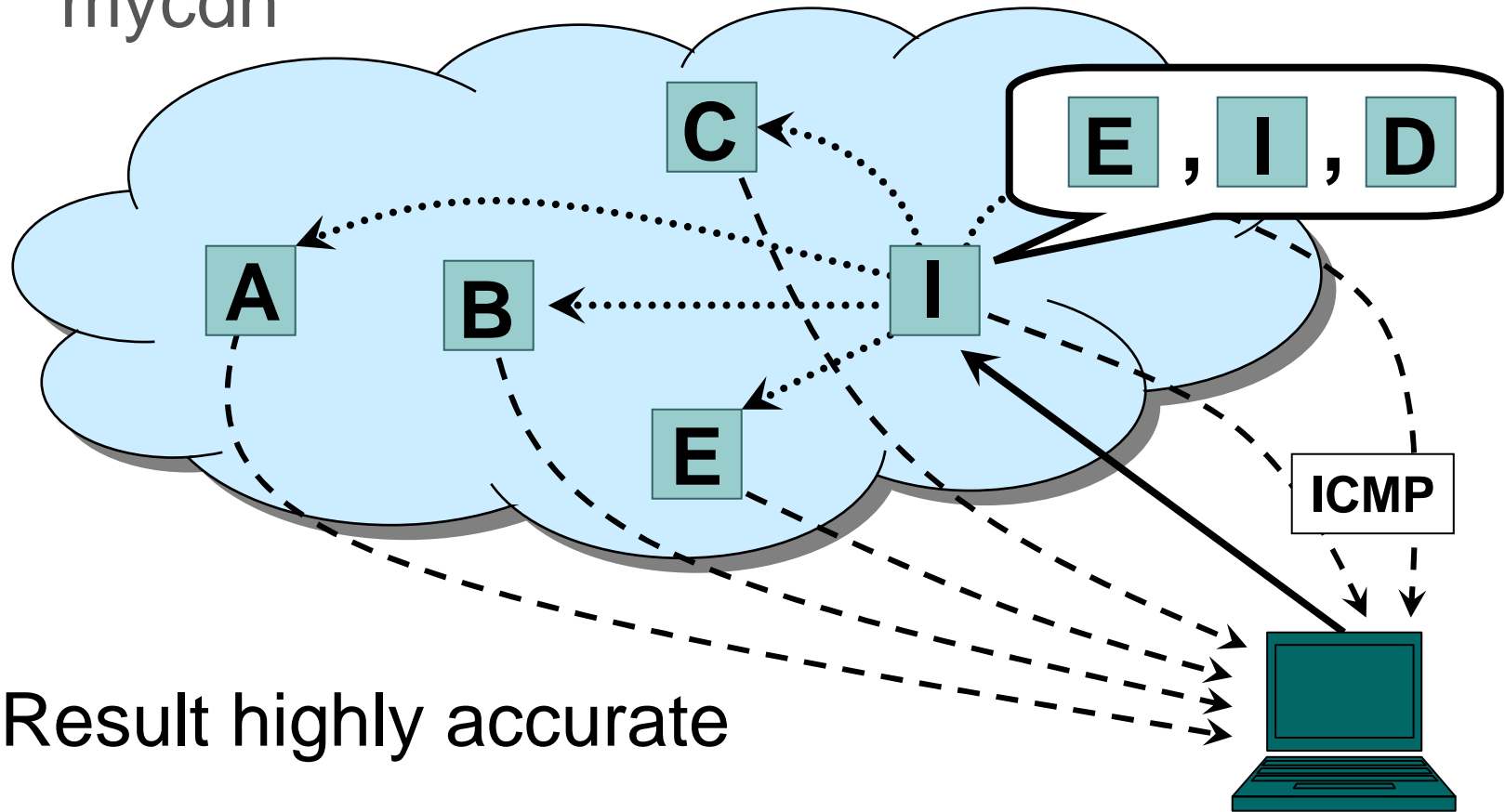


● ● ● | Strawman: probe & find nearest  
mycdn



- ✓ Result highly accurate
- ✗ Lots of probing
- ✗ Slow to compute

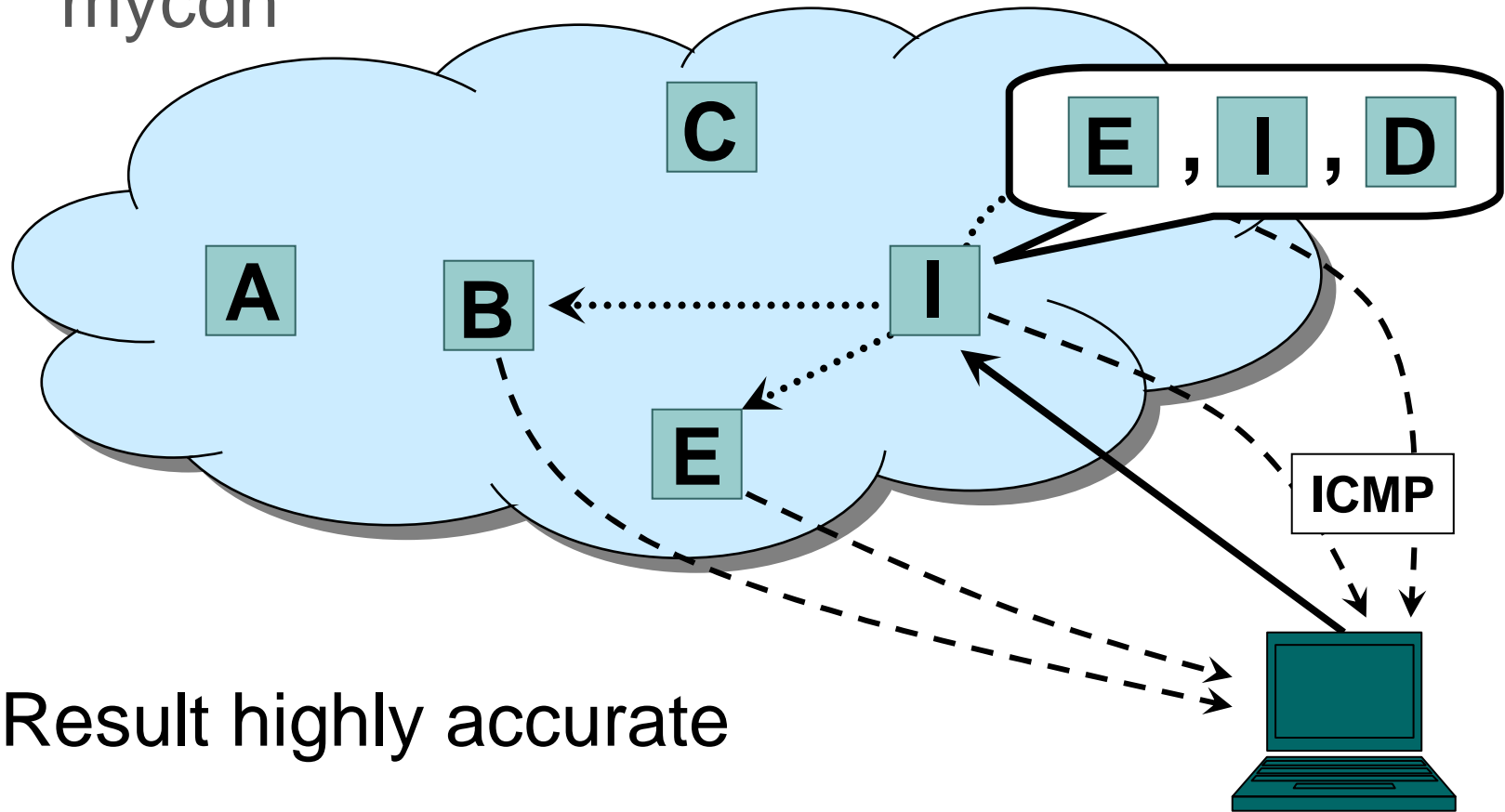
● ● ● | Strawman: probe & find nearest  
mycdn



- ✓ Result highly accurate
- ✗ ~~Lots of probing~~
- ✗ Slow to compute

# ●●● Avoid probing on-demand

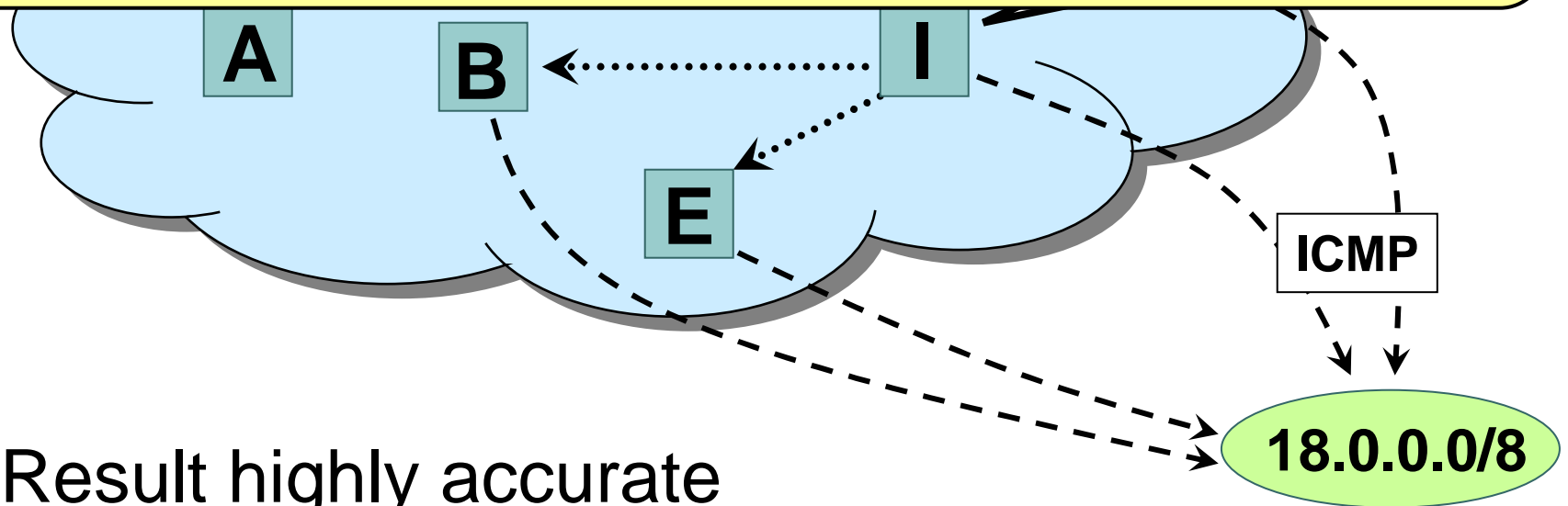
mycdn



- ✓ Result highly accurate
- ✗ ~~Lots of probing~~
- ✗ ~~Slow to compute~~

- Avoid probing on-demand

[IMC05] shows IP prefixes often preserve locality  
( 99% of /24s by stub AS at the same location )



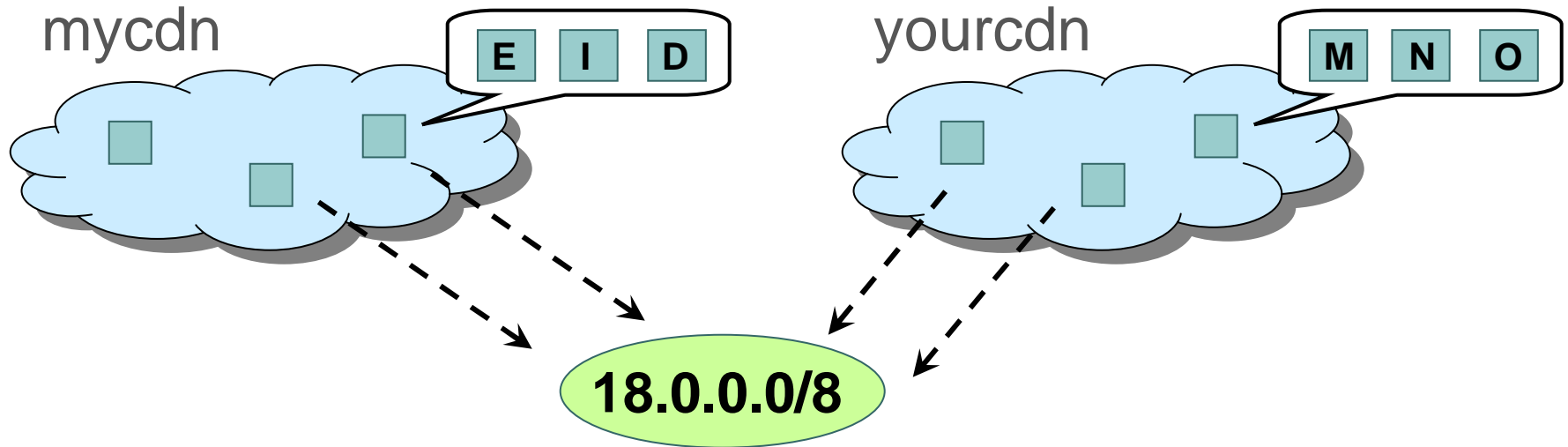
✓ Result highly accurate

✗ ~~Lots of probing~~

✗ ~~Slow to compute~~

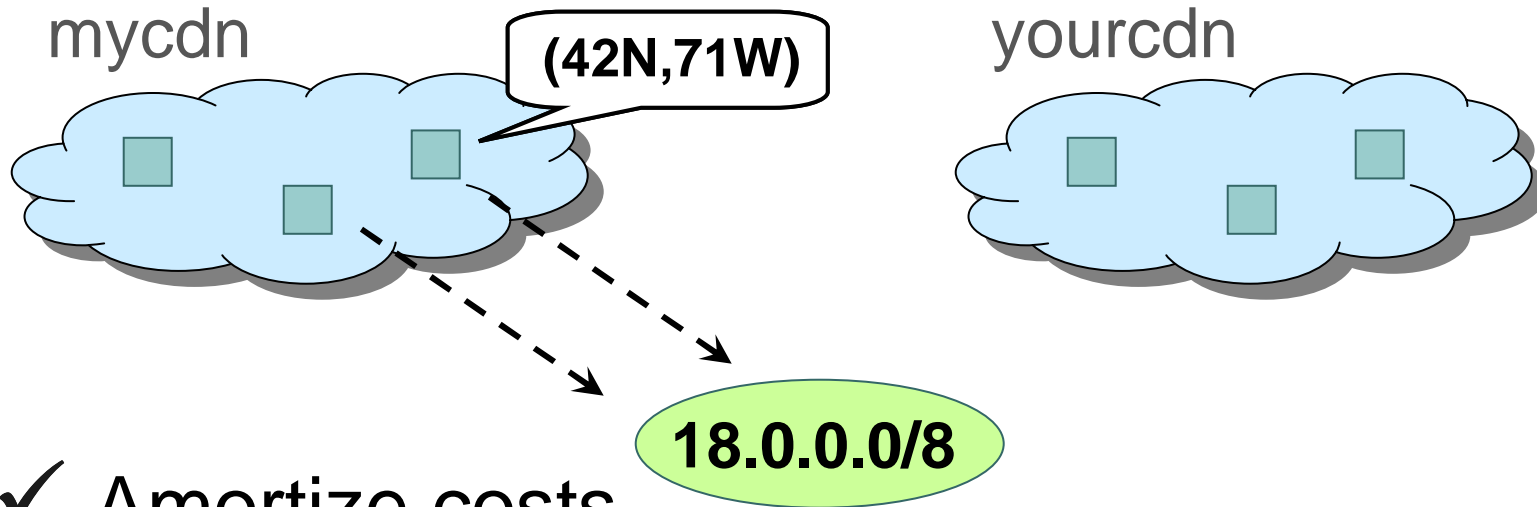
This is the problem  
Akamai must solve

# ••• | What about *yourcdn*?



- ✓ Result highly accurate
- ✗ ~~Lots of probing~~
- ✗ ~~Slow to compute~~

# ••• Idea: Use geographic coordinates



- ✓ Amortize costs
- ✓ Stable across services, time, and failures
- ✓ Result highly accurate
- ✗ ~~Lots of probing~~
- ✗ ~~Slow to compute~~

Assume all replicas  
know geo-coords

## ••• | OASIS provides...

- ✓ Amortize costs
- ✓ Stable across time, services, and failures
- ✓ Result highly accurate
- ✓ Fast response time
- ✓ Supports flexible anycast policies
  - Balances tension between:
    - Performance: finding nearest replica
    - Cost: minimizing 95% bandwidth usage

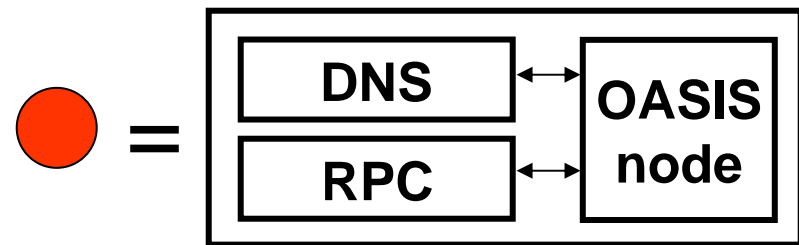
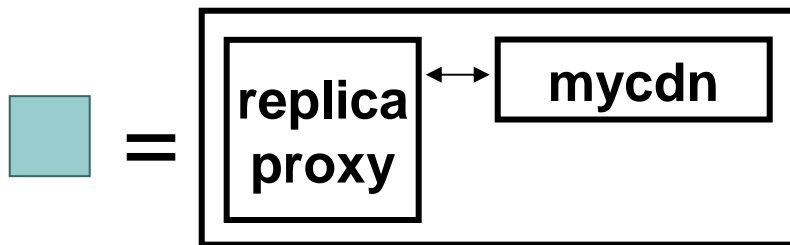
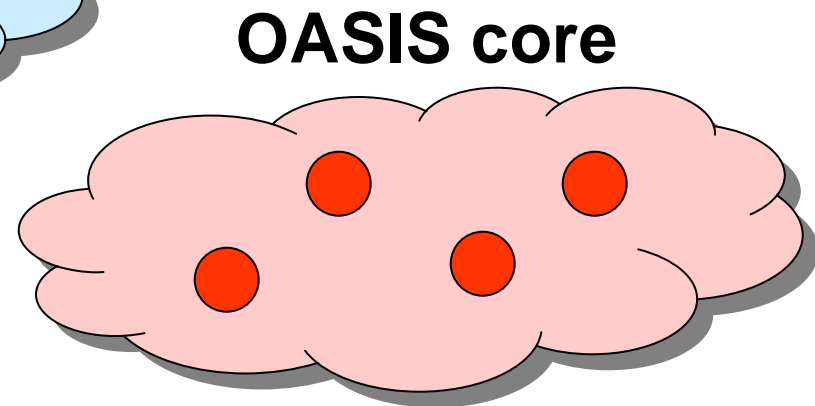
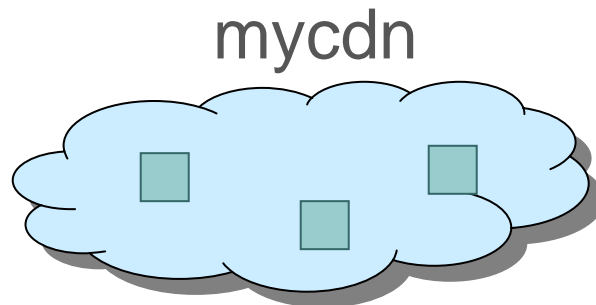


# Outline

- Architecture and design decisions
- Detailed design
- Evaluation
- Deployment and integration lessons
  - OASIS deployed since November 2005
  - Currently in use by 10 services

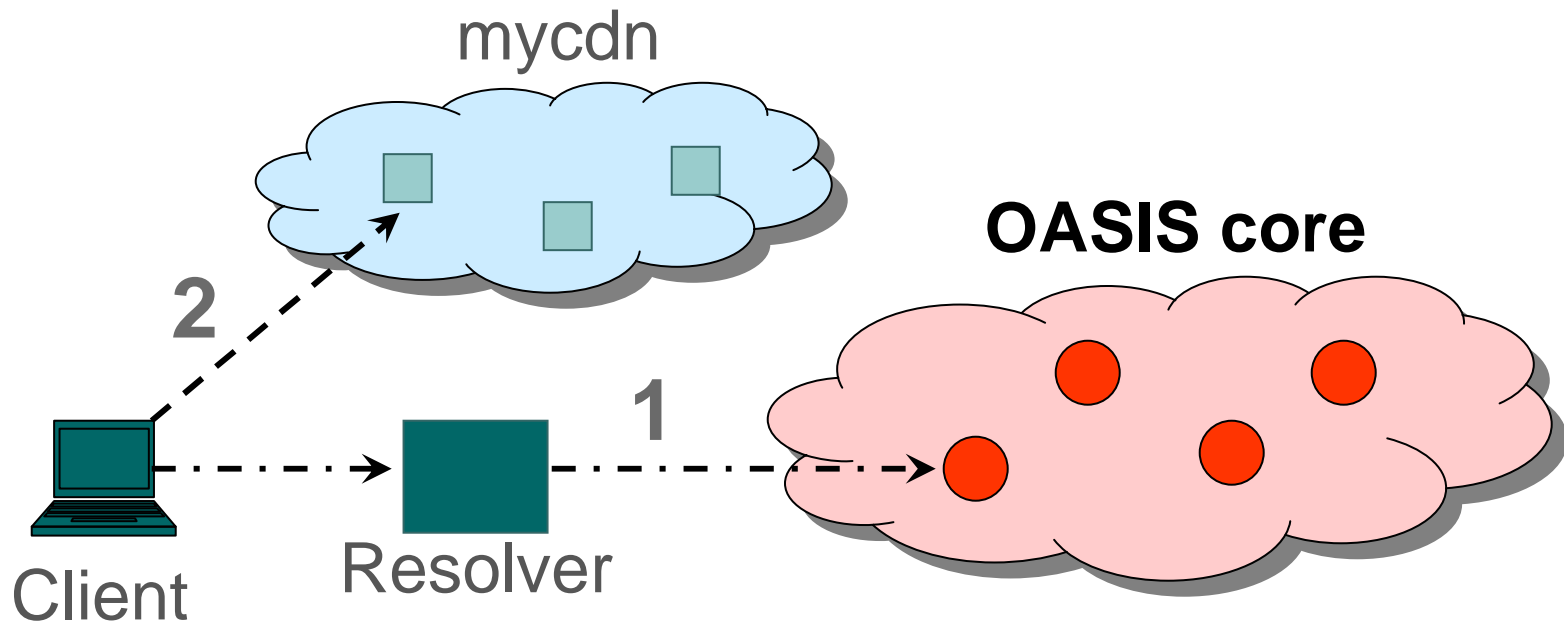


# Two-tier architecture



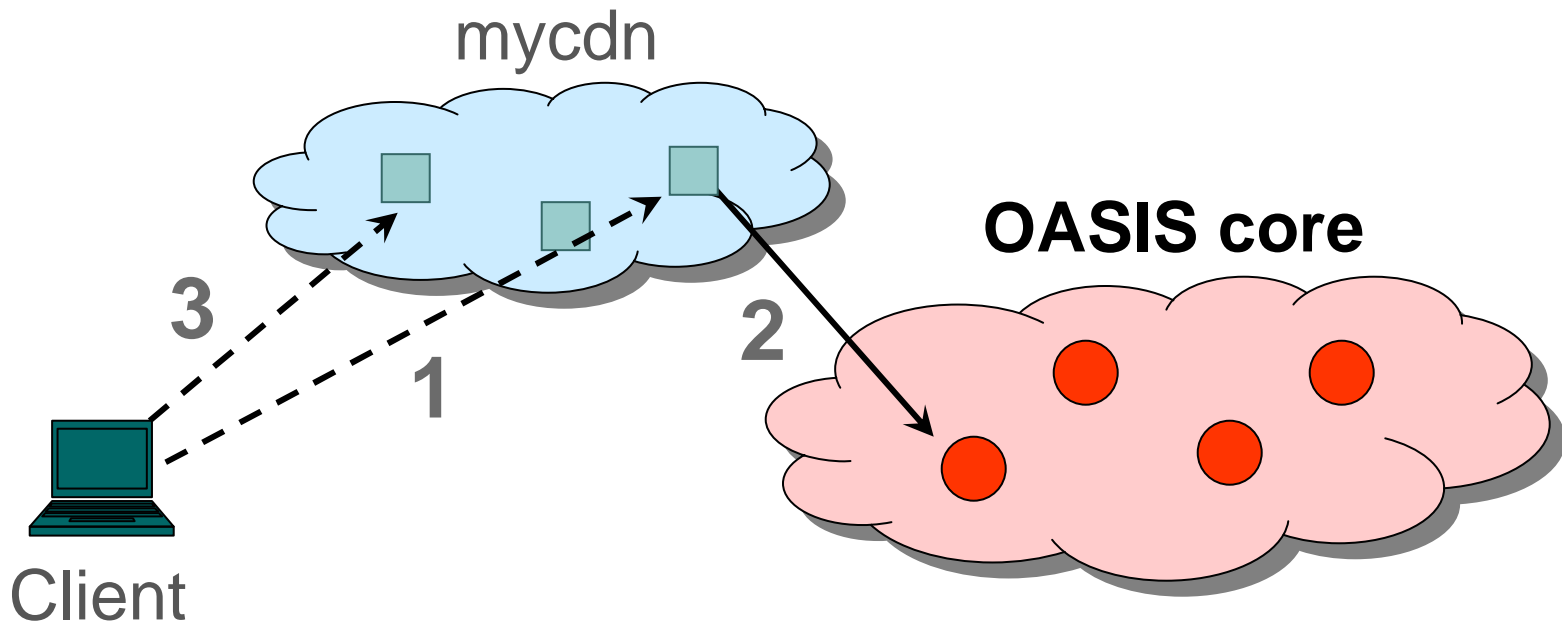
Large set of *replicas* that assist in measurement  
Reliable *core* of hosts that implement anycast

# Using OASIS via DNS



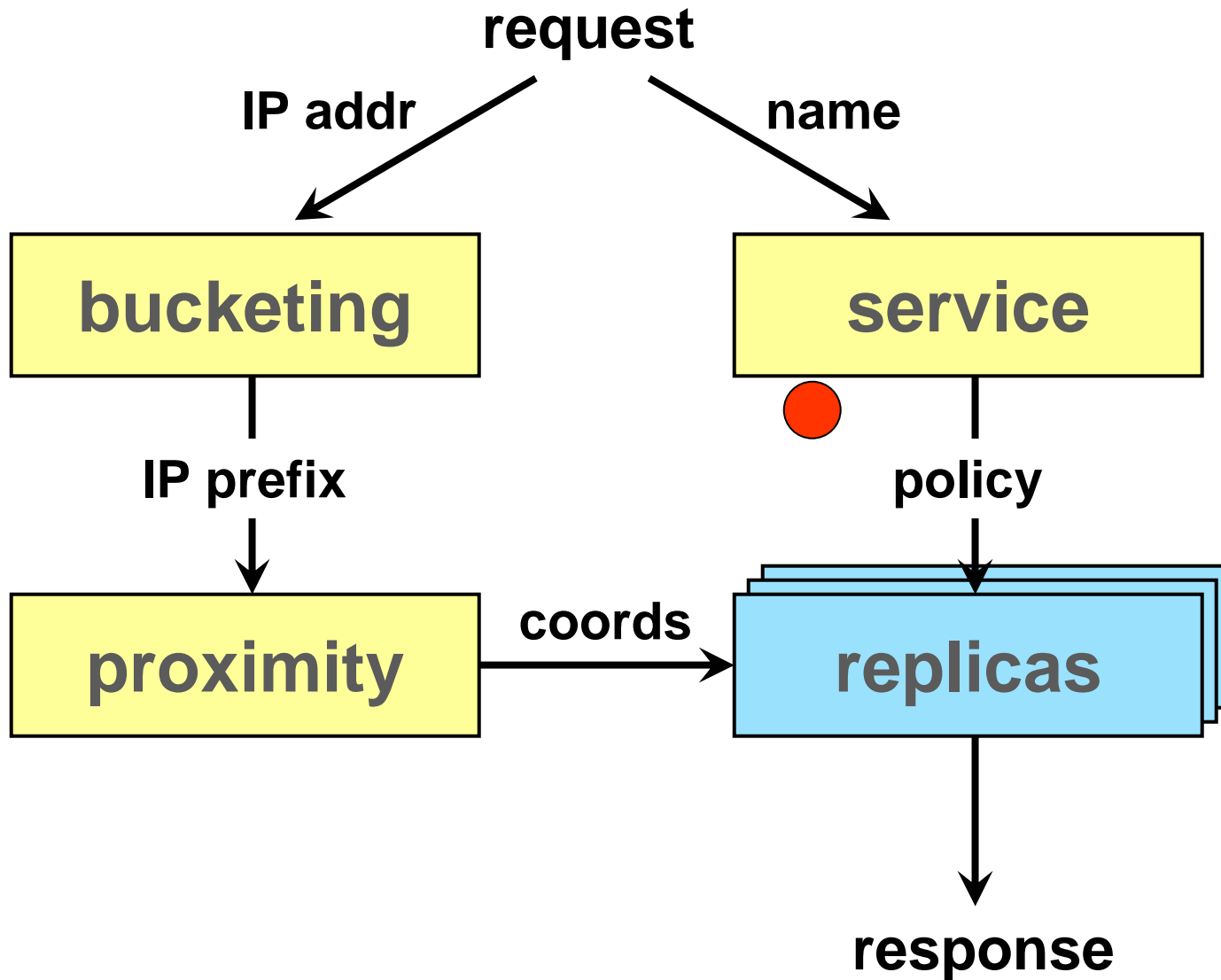
1. Client issues DNS request for *mycdn.nyuld.net*
2. OASIS redirects client to nearby application replica

# Using OASIS via HTTP

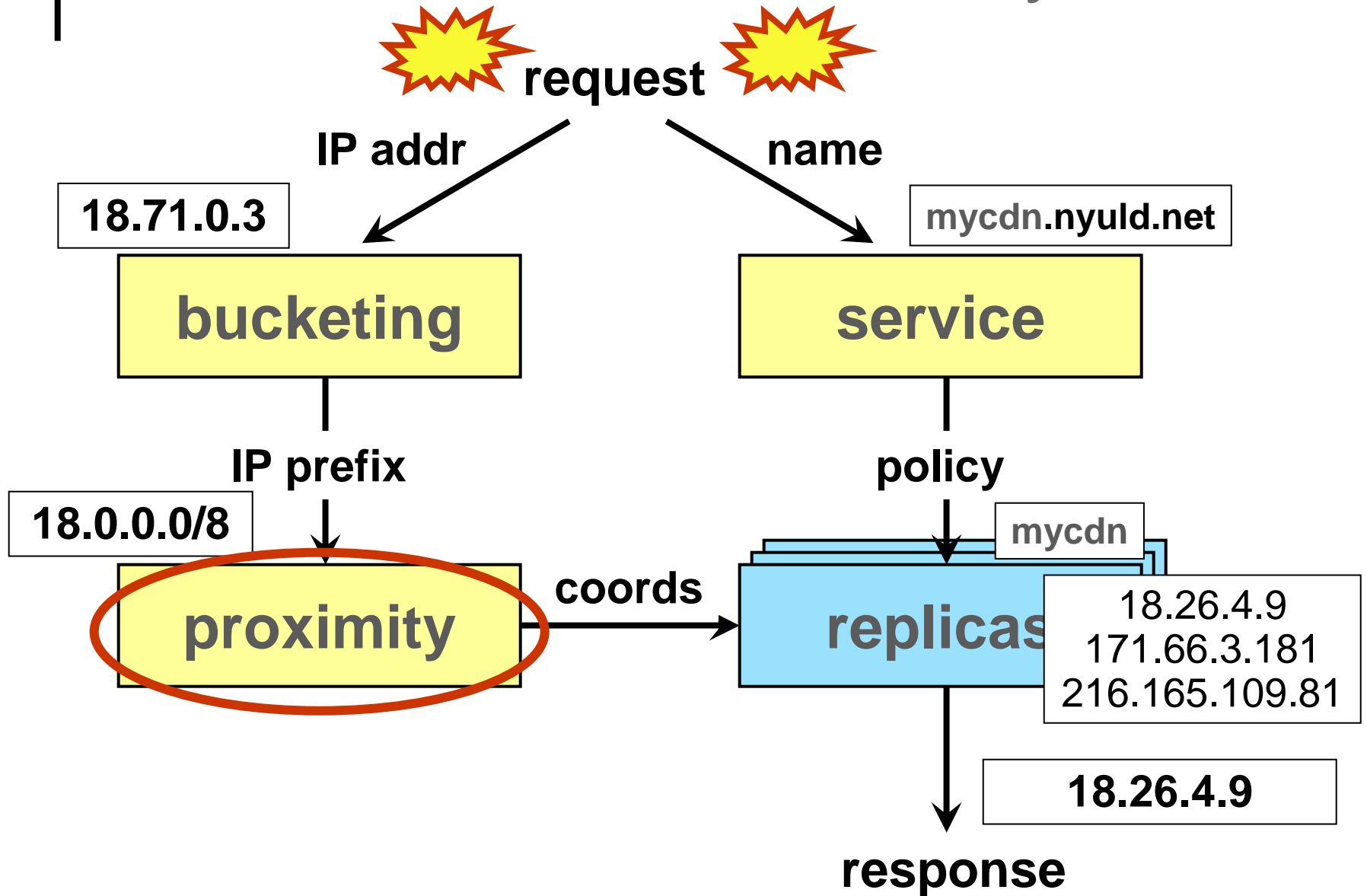


1. Client issues HTTP request
2. Web cgi-bin issues RPC to OASIS core
3. Client redirected to nearby application replica

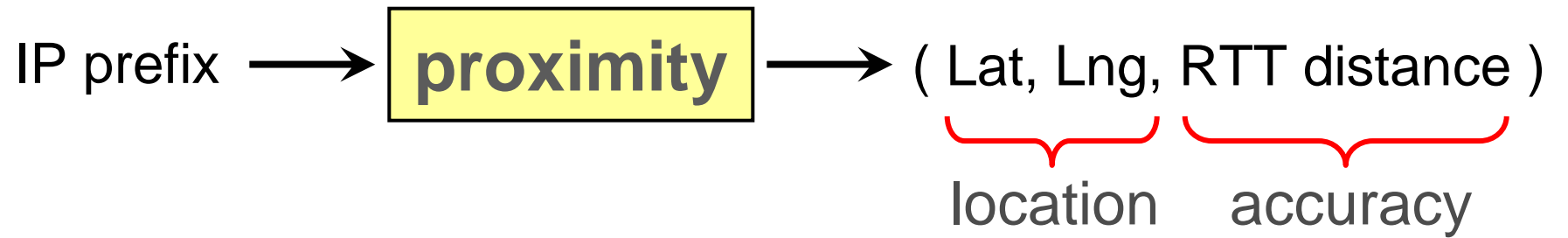
••• | How does core answer anycast?



# How does core answer anycast?



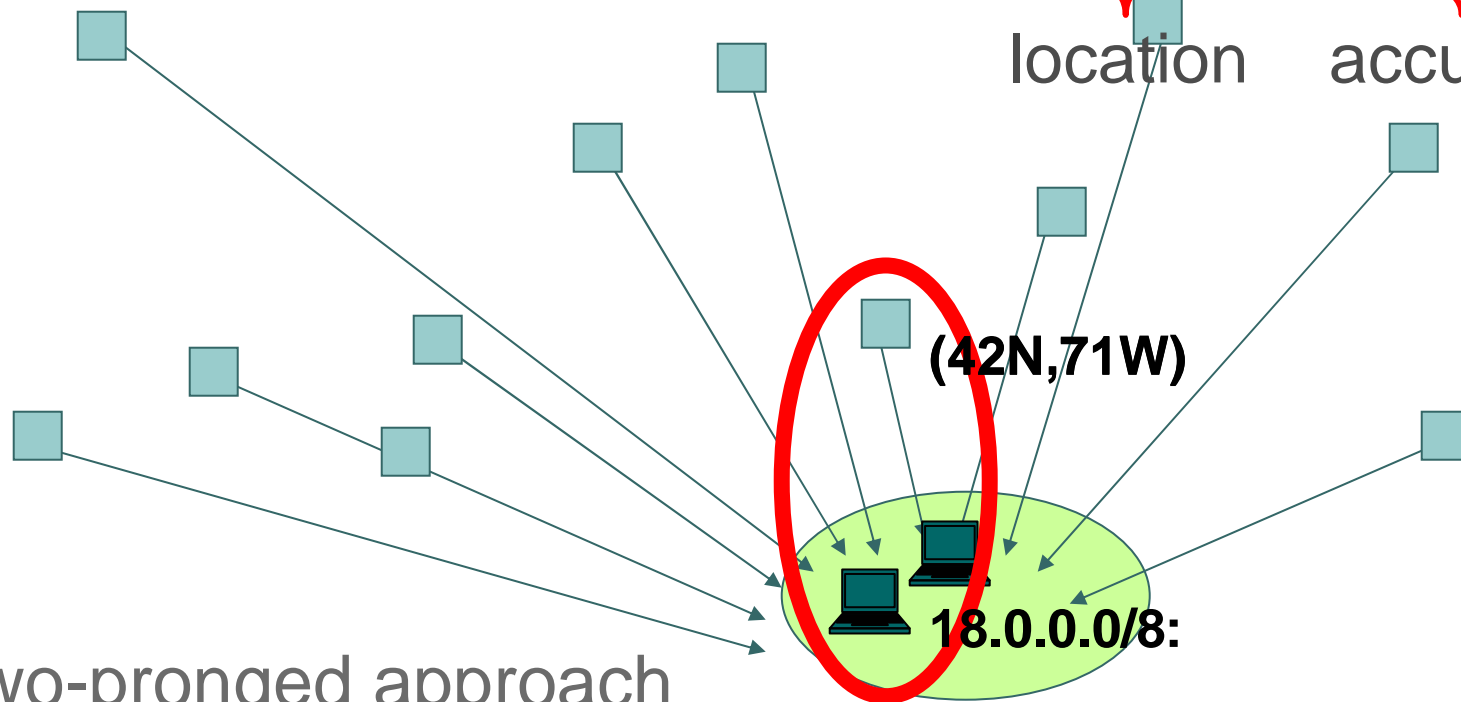
- • • | How to map IP prefix to coords?



# How to map IP prefix to coords?

IP prefix → **proximity** → ( Lat, Lng, RTT distance )

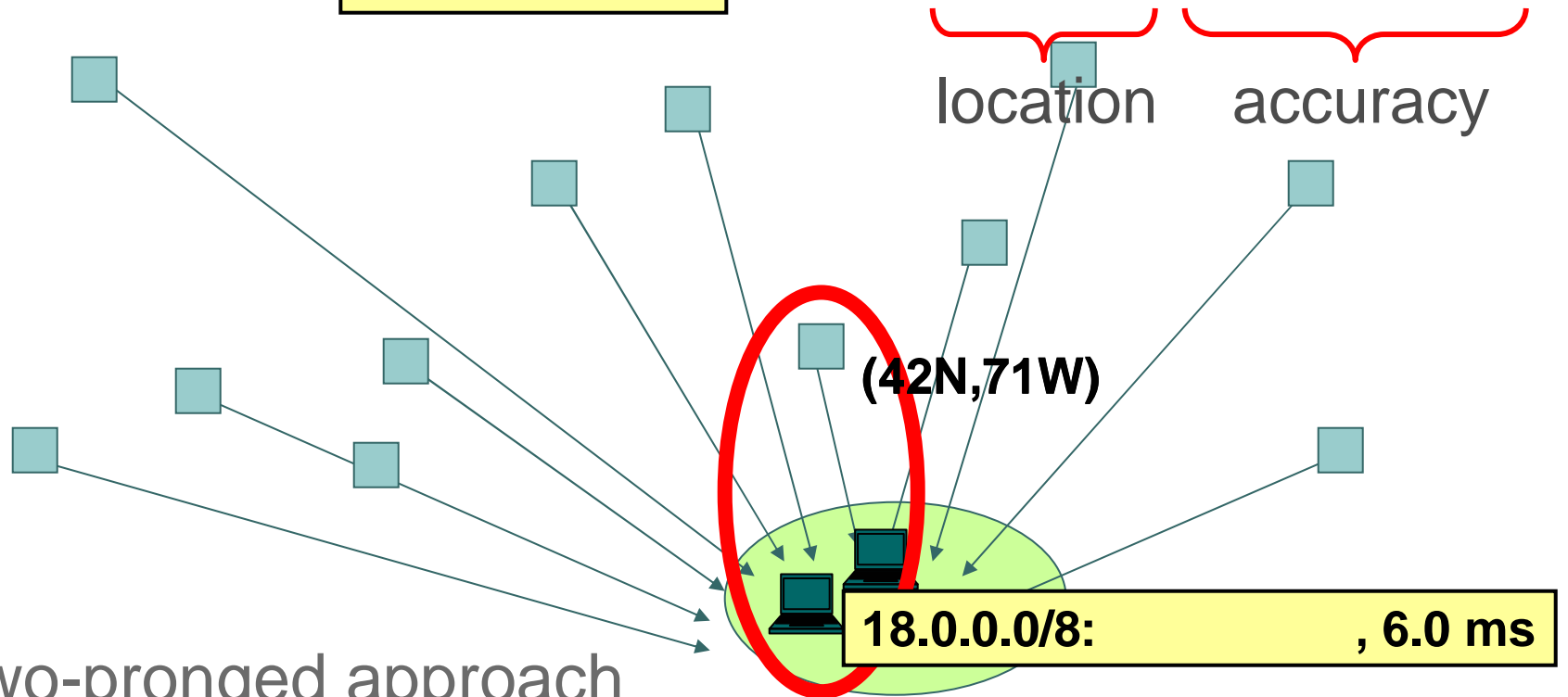
location accuracy



- Two-pronged approach
  - Find *closest* replica proxy

# How to map IP prefix to coords?

IP prefix → **proximity** → ( Lat, Lng, RTT distance )

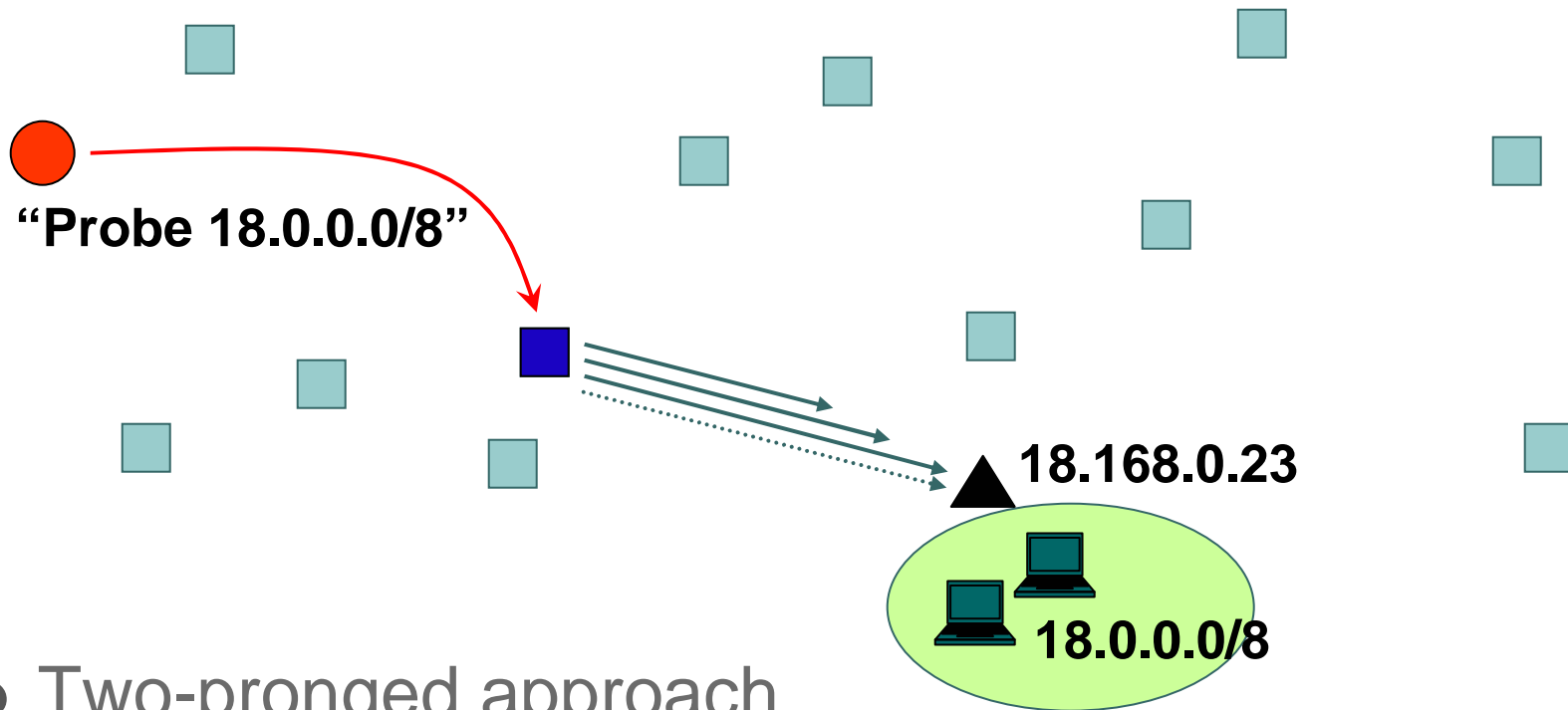


## ○ Two-pronged approach

- Find *closest* replica proxy
- Use closest replica's geo-coords + error RTT as location



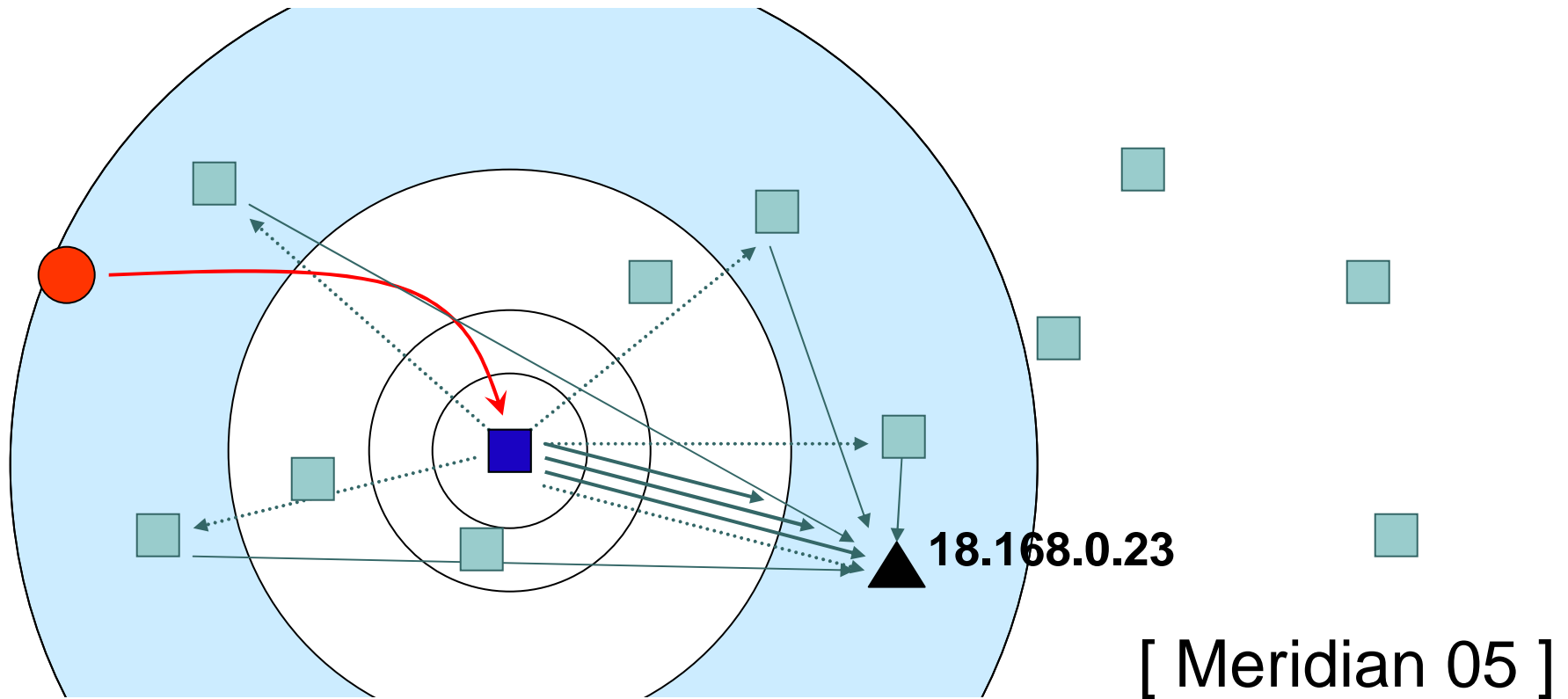
- ● ● | Find replica nearest prefix efficiently



- Two-pronged approach

- Find *closest* replica proxy **with less probing**
- Use closest replica's geo-coords + error RTT as location

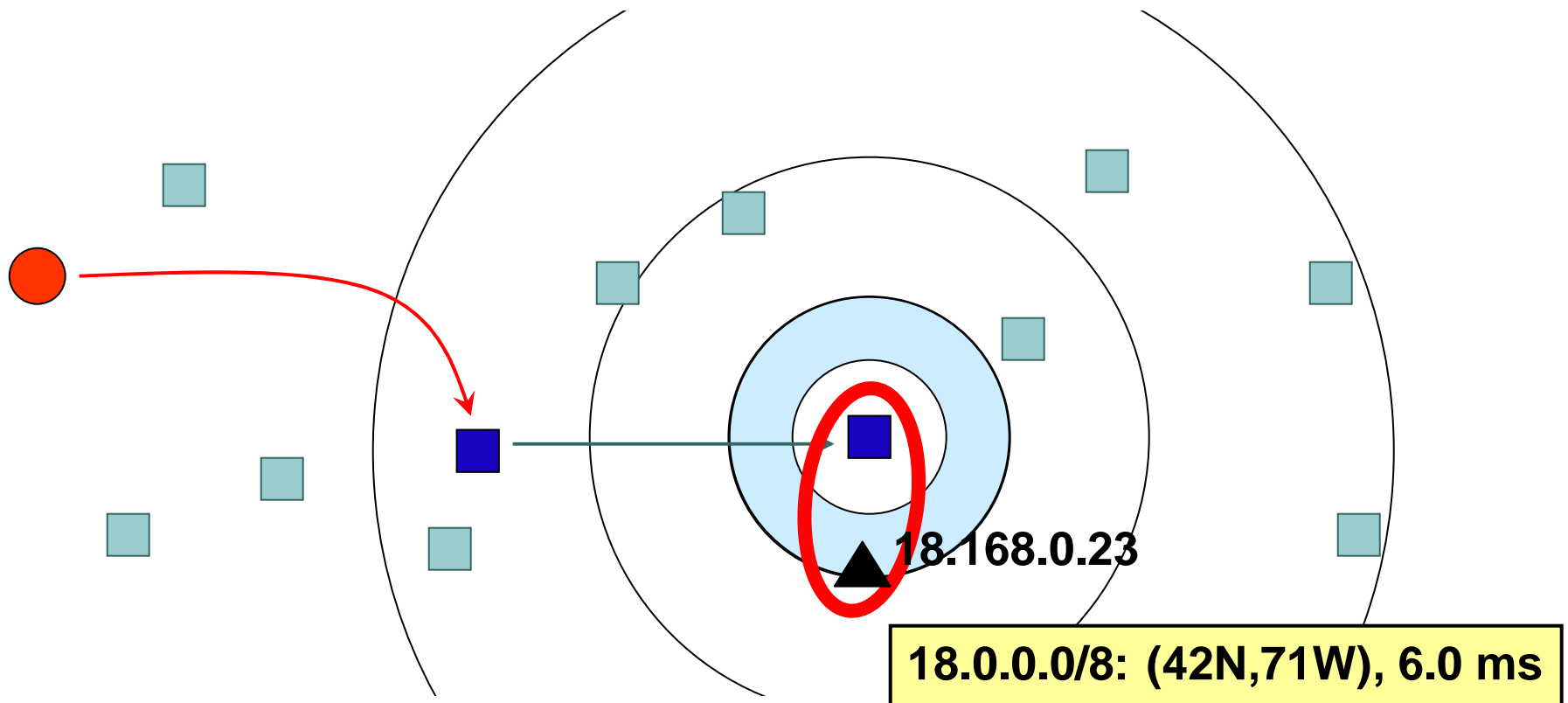
- ● ● | Find replica nearest prefix efficiently



- Two-pronged approach

- Find *closest* replica proxy **with less probing**
- Use closest replica's geo-coords + error RTT as location

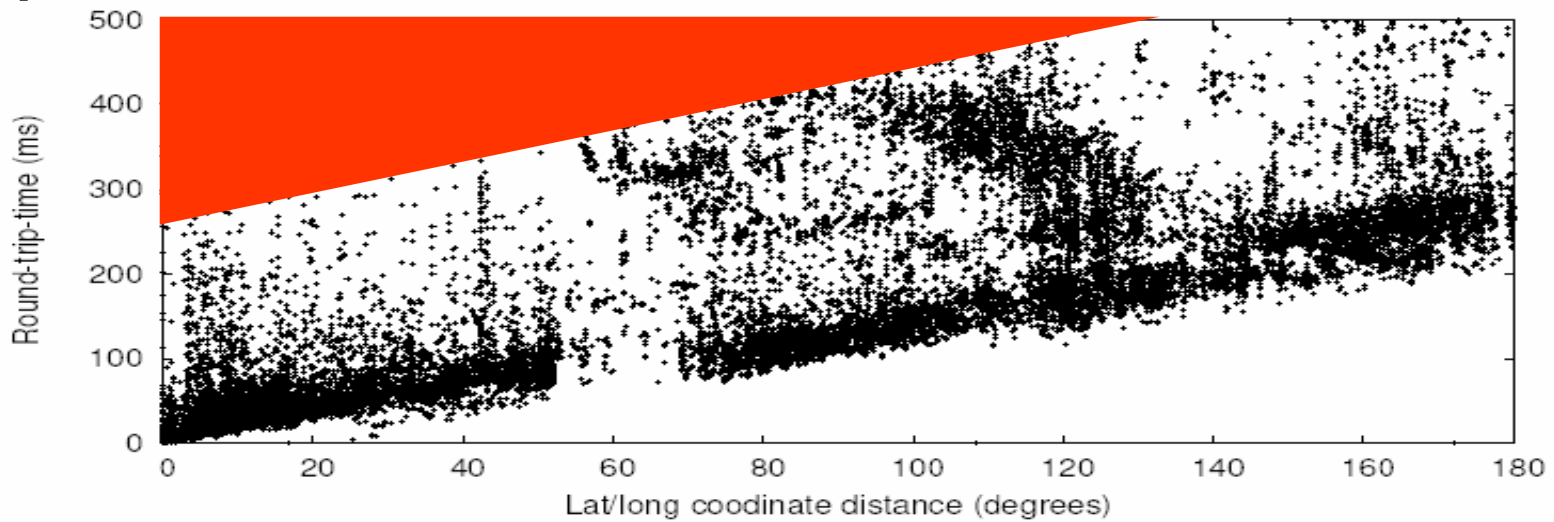
- ● ● | Find replica nearest prefix efficiently



- Two-pronged approach

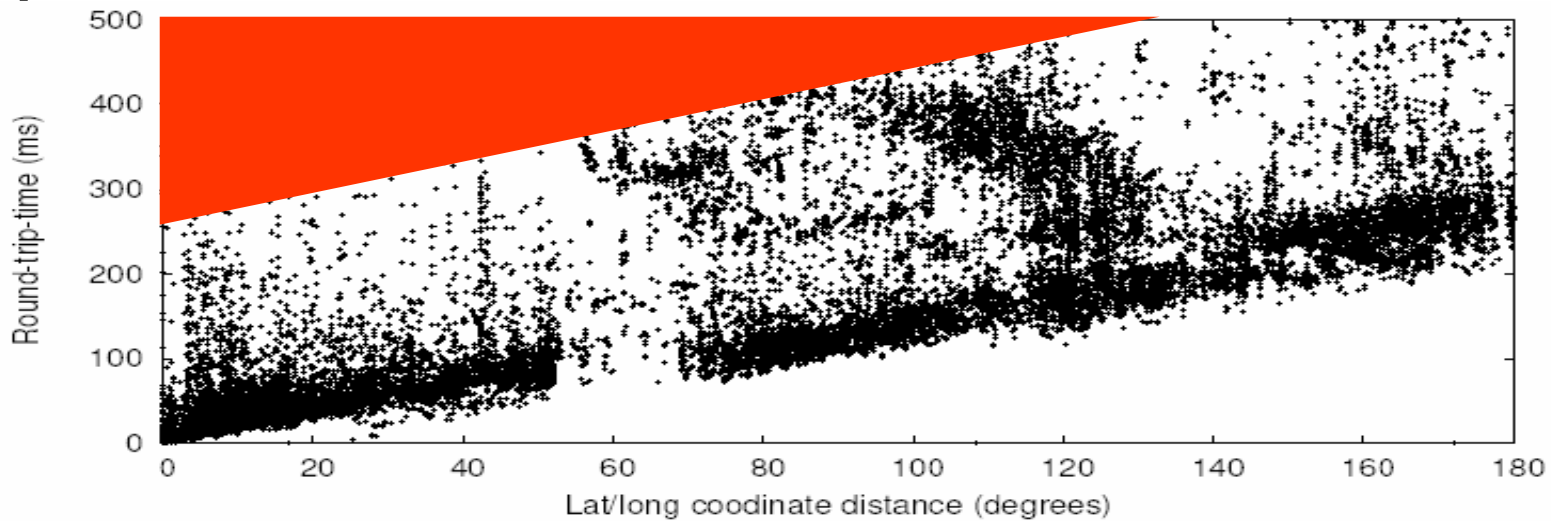
- Find *closest* replica proxy **with less probing**
- Use closest replica's geo-coords + error RTT as location

# Geographic distance vs. RTT



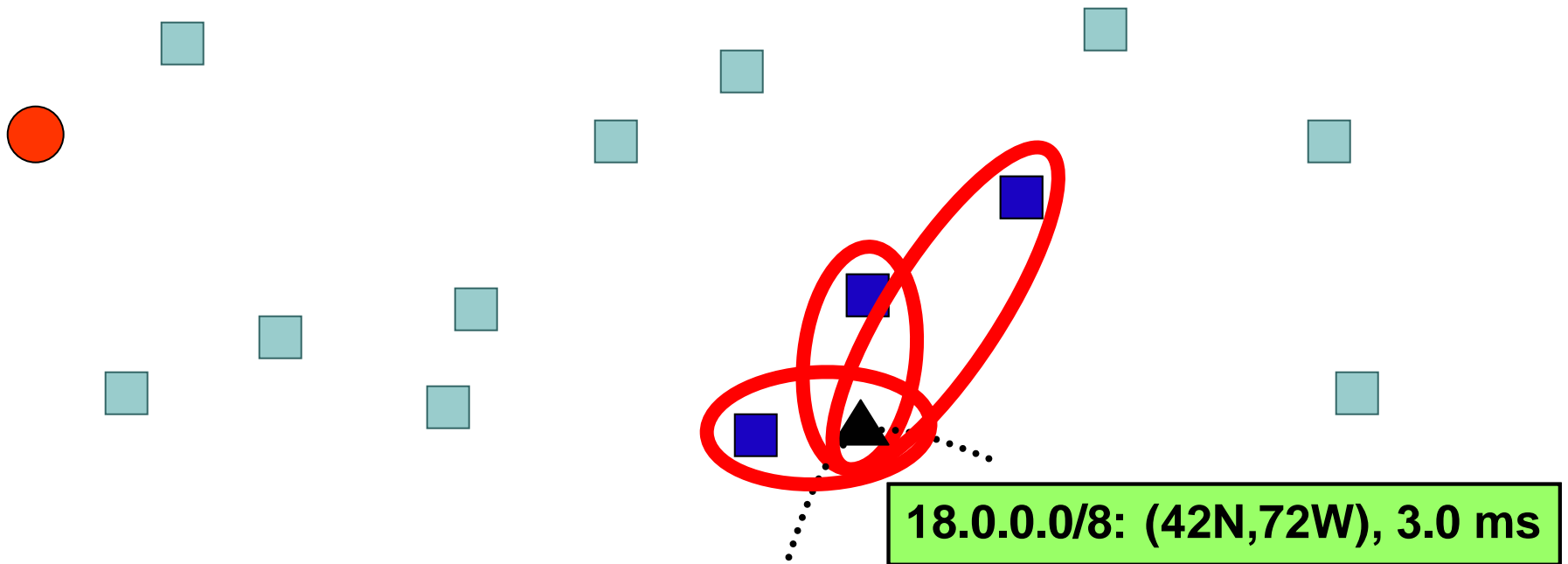
- Strong correlation b/w geographical distance and RTT

# Geographic distance vs. RTT



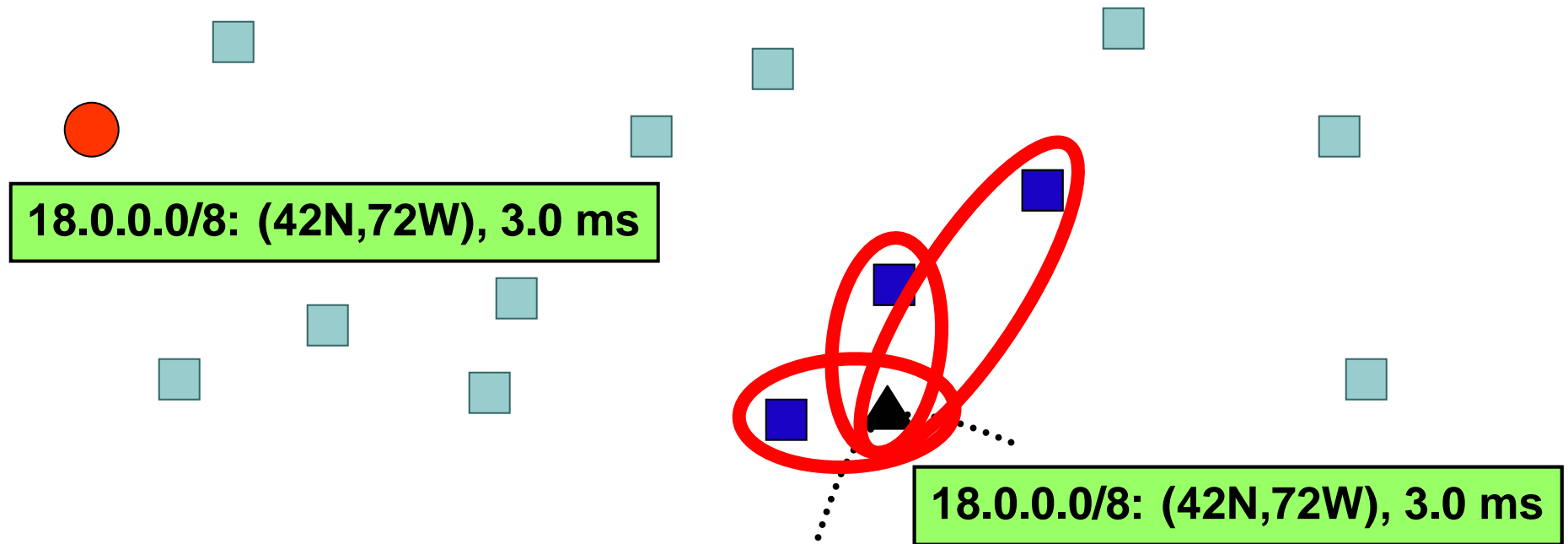
- Strong correlation b/w geographical distance and RTT
- RTT accuracy has real-world meaning
  - Check if new coordinates improve accuracy vs. old coords

# Geographic distance vs. RTT



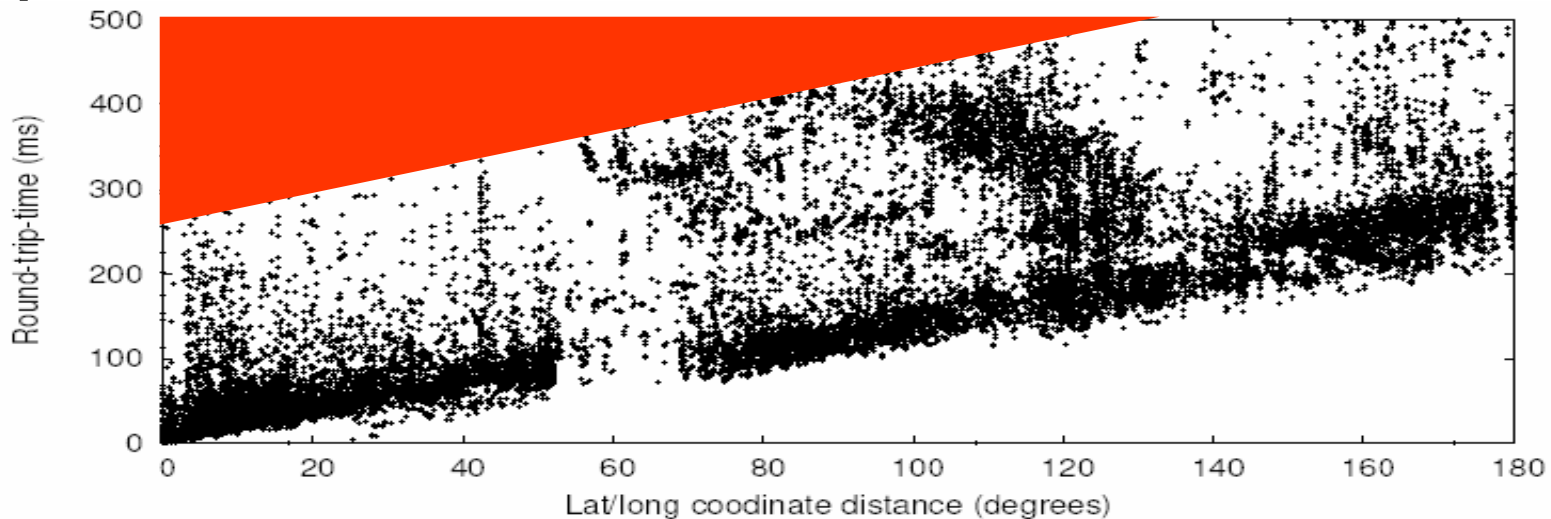
- RTT accuracy has real-world meaning
  - Check if new coordinates improve accuracy vs. old coords

# Geographic distance vs. RTT



- RTT accuracy has real-world meaning
  - Check if new coordinates improve accuracy vs. old coords

# Geographic distance vs. RTT



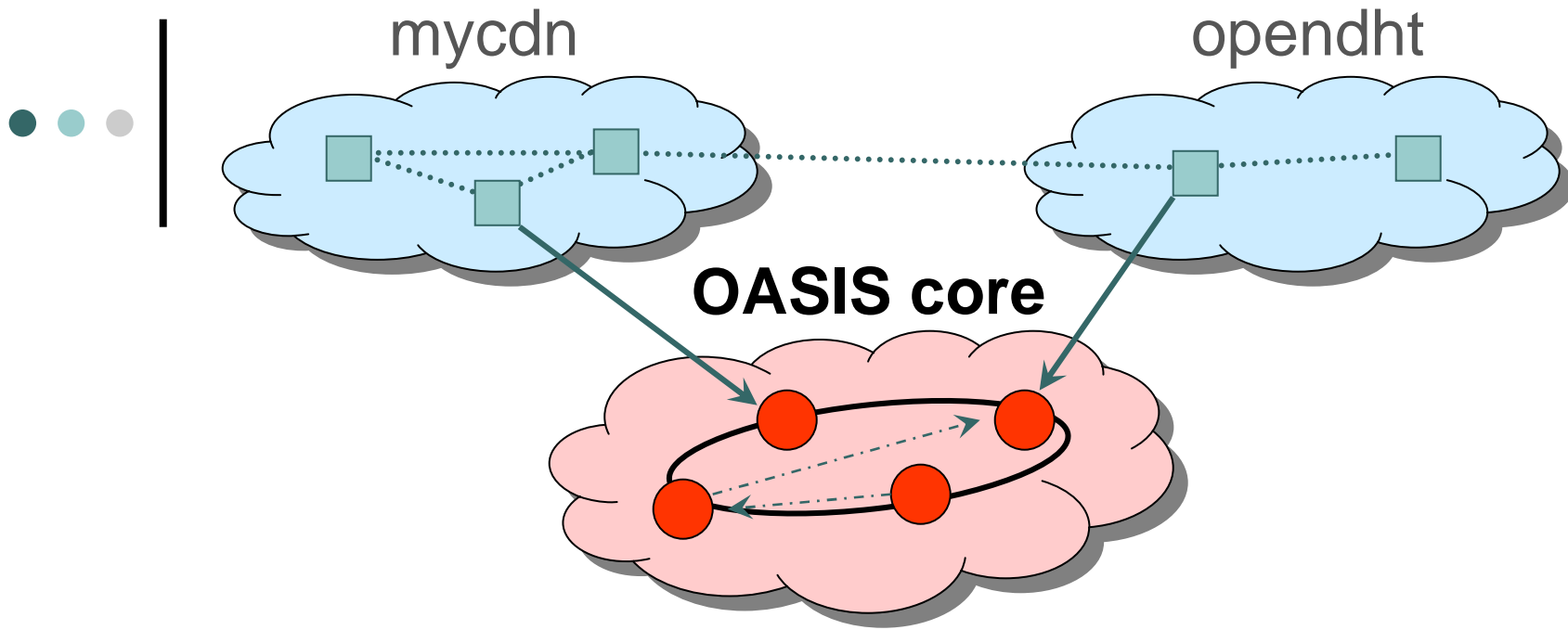
- Strong correlation b/w geographical distance and RTT
- RTT accuracy has real-world meaning
  - Check if new coordinates improve accuracy vs. old coords
- Useful for sanity check for network peculiarities
  - Do multiple results satisfy constraints (e.g., speed of light) ?





# Outline

- Architecture and design decisions
- Detailed design
- Evaluation
- Deployment and integration lessons
  - OASIS deployed since November 2005
  - Currently in use by 10 services



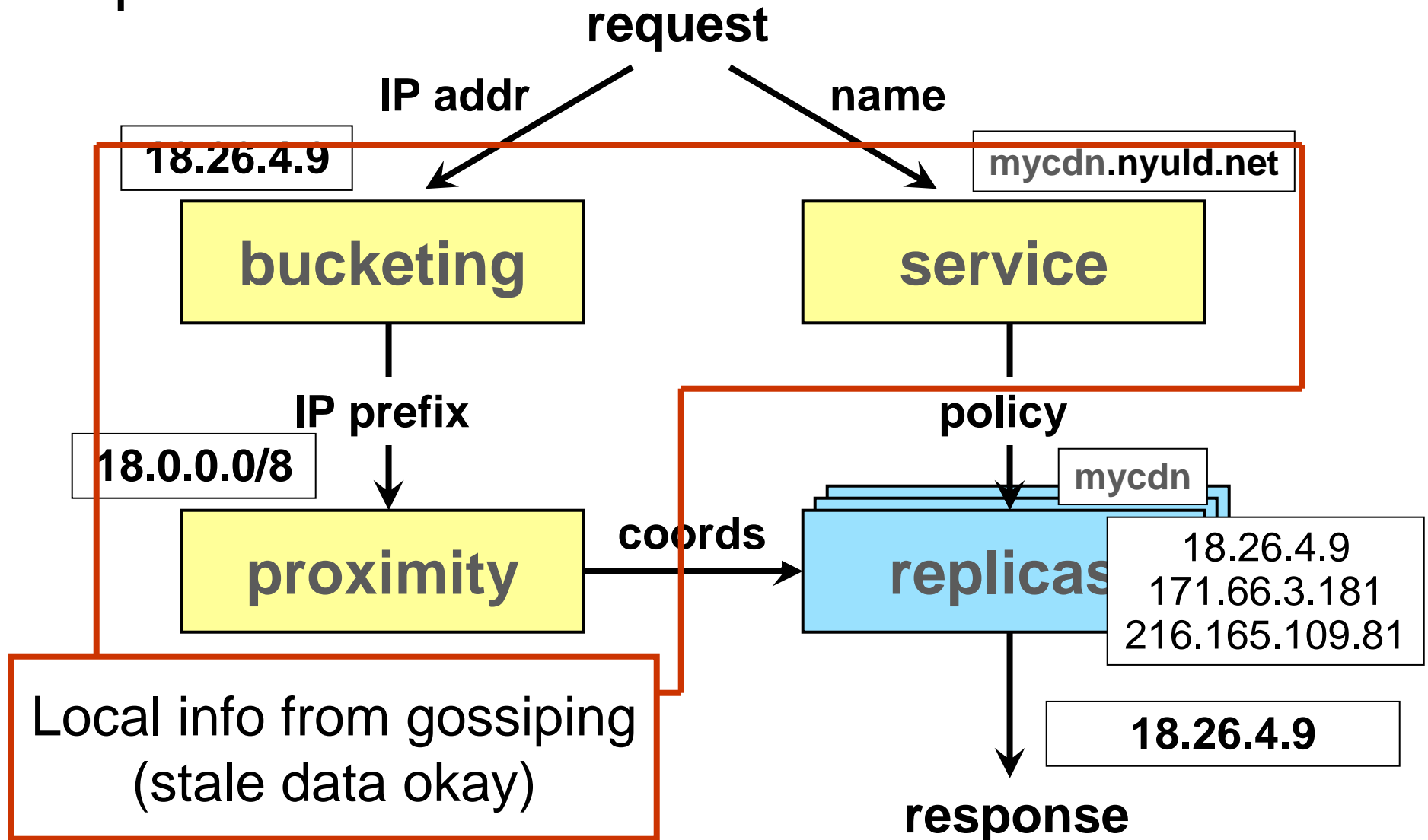
## ○ OASIS core

- Global membership view
- Epidemic gossiping
  - Scalable failure detection
  - Spread policies, prefix→coords
- Consistent hashing
  - Divide up responsibility for prefixes

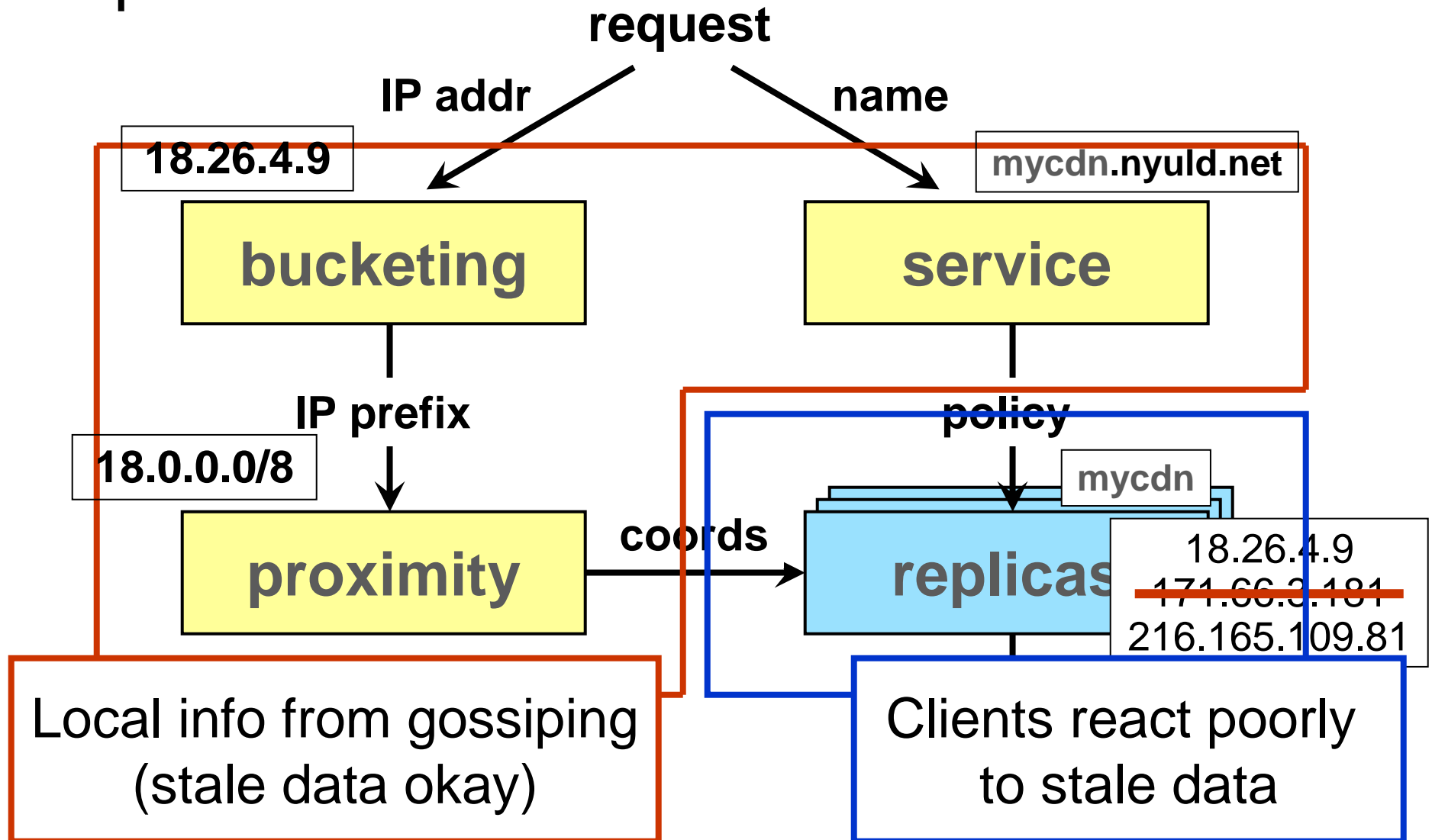
## ○ Service replicas

- Heartbeats to OASIS node
- Form global Meridian overlay for probing

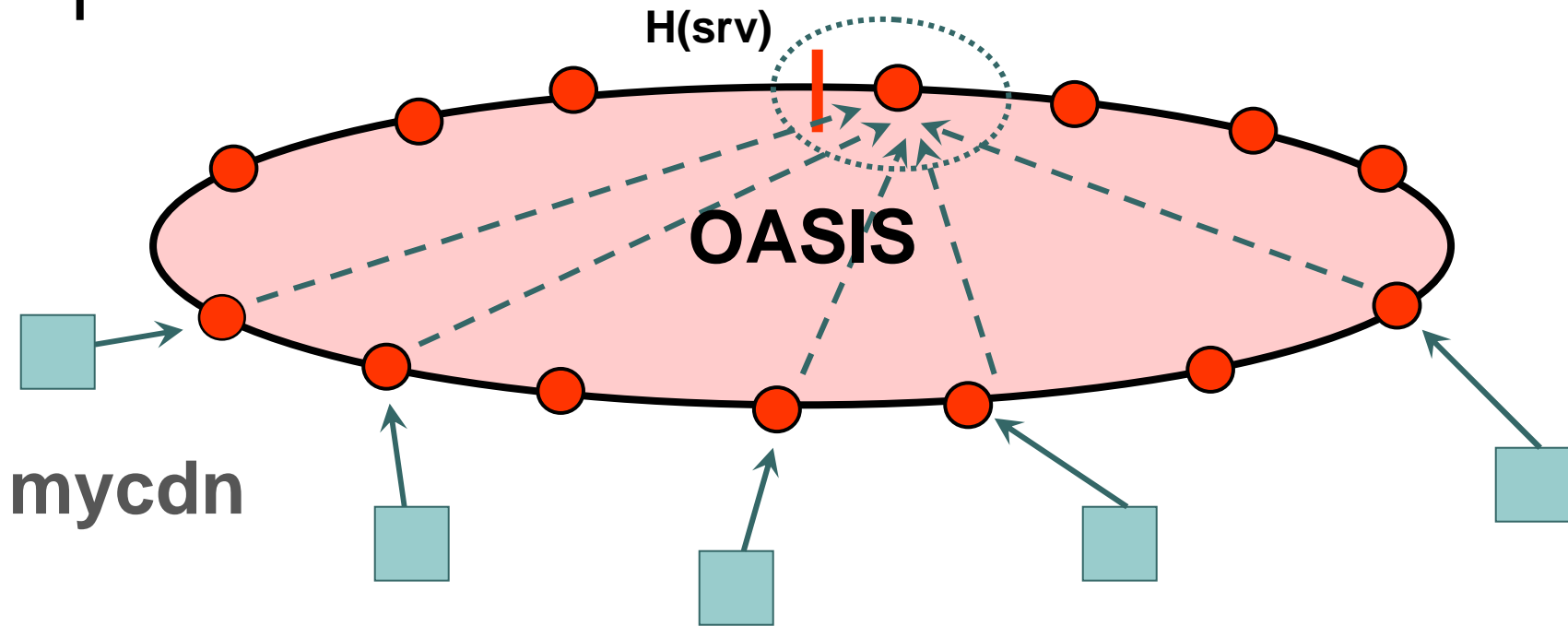
# How to find “nearby” nodes?



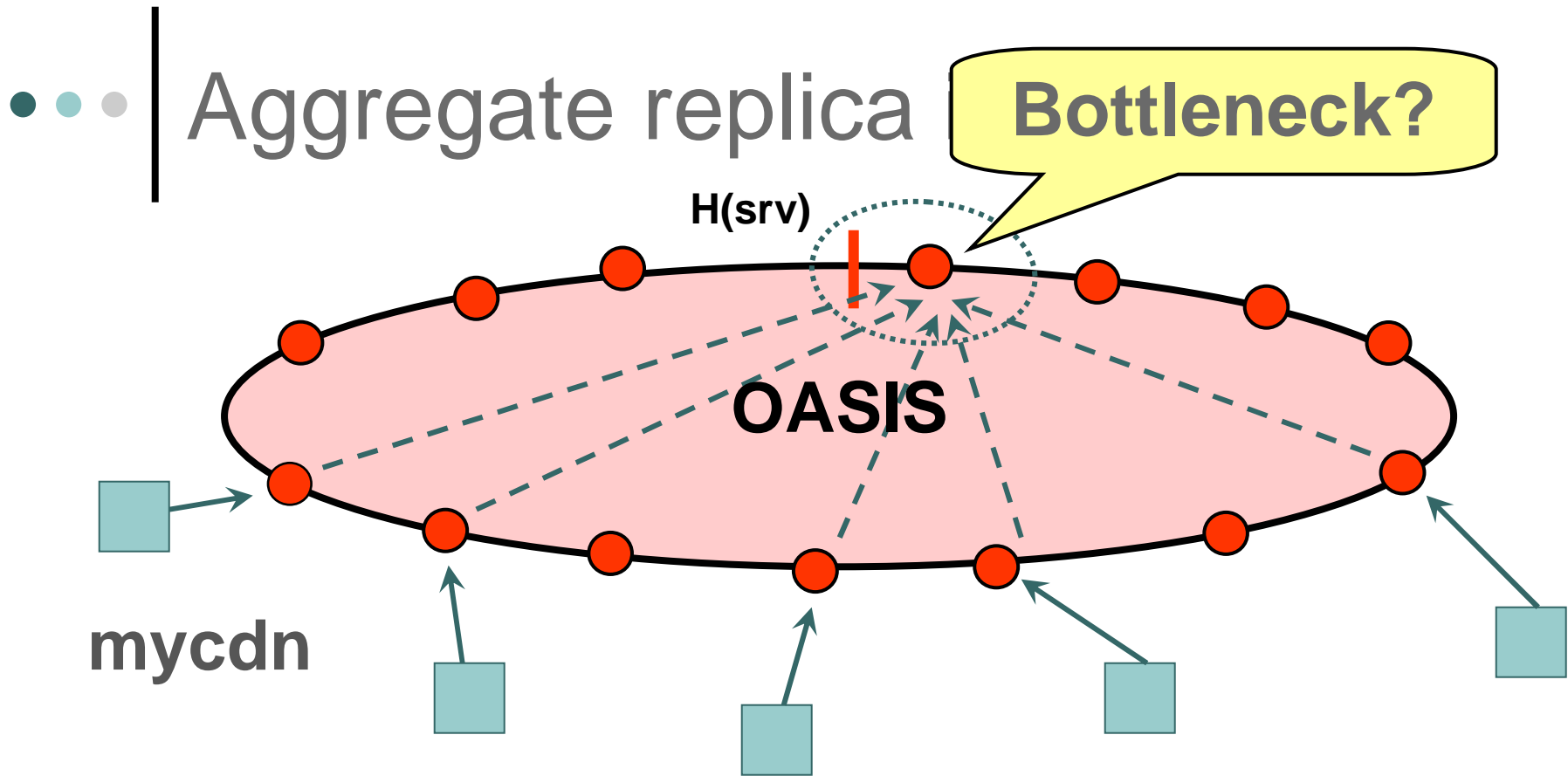
# How to find “nearby” nodes?



# Aggregate replica information

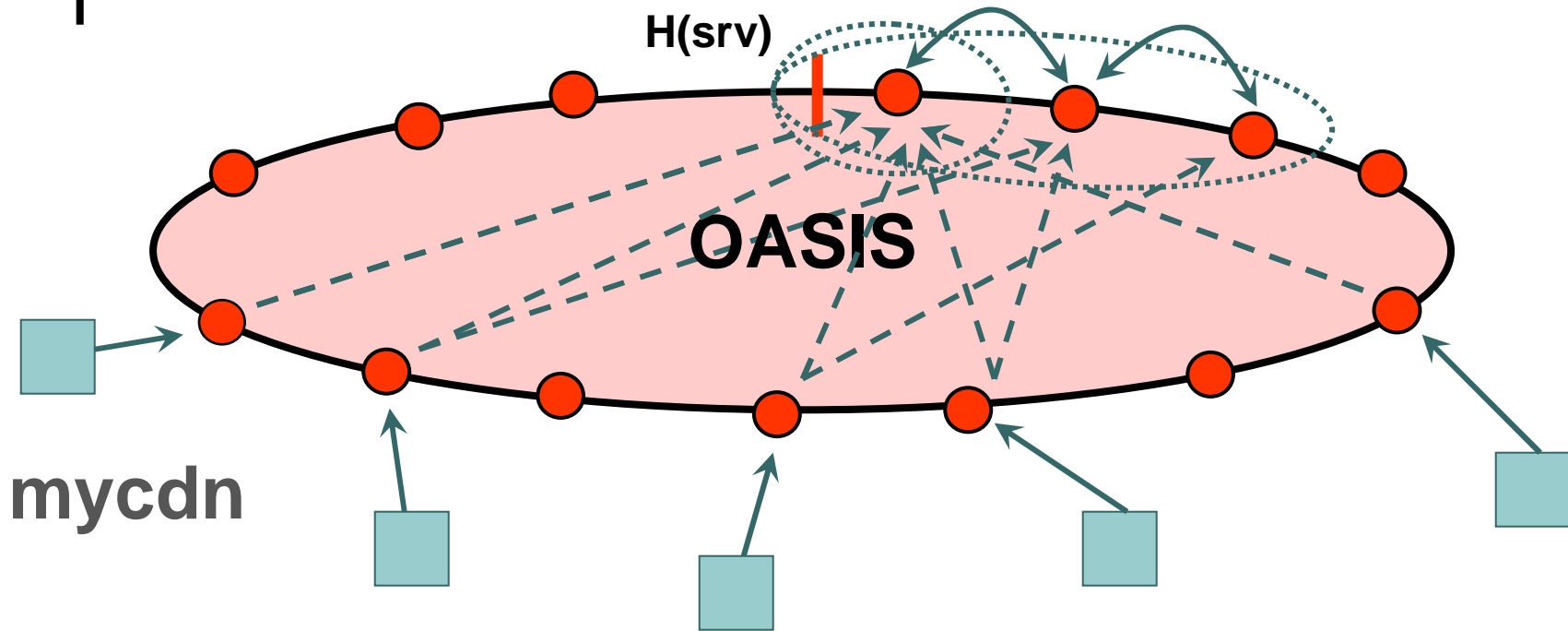


- Define service's rendezvous node via consistent hashing
- Service replicas send keepalives to nearby OASIS nodes
- Update rendezvous when replicas join, leave, large load change



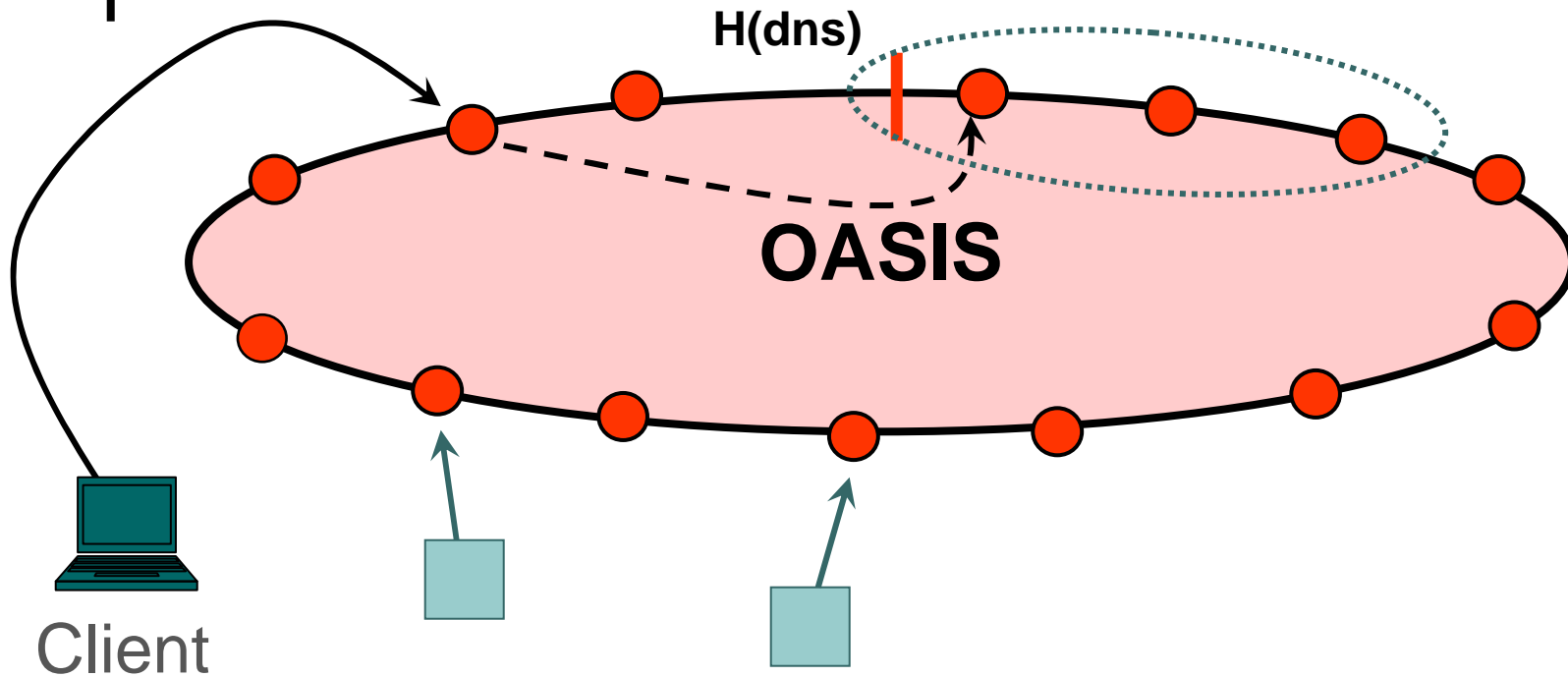
- Define service's rendezvous node via consistent hashing
- Service replicas send keepalives to nearby OASIS nodes
- Update rendezvous when replicas join, leave, large load change

# Aggregate replica information



- Aggregate over  $k$  nodes for scalability
- Rendezvous gossip liveness state for loose consistency
- $k$  can be dynamic for better scalability

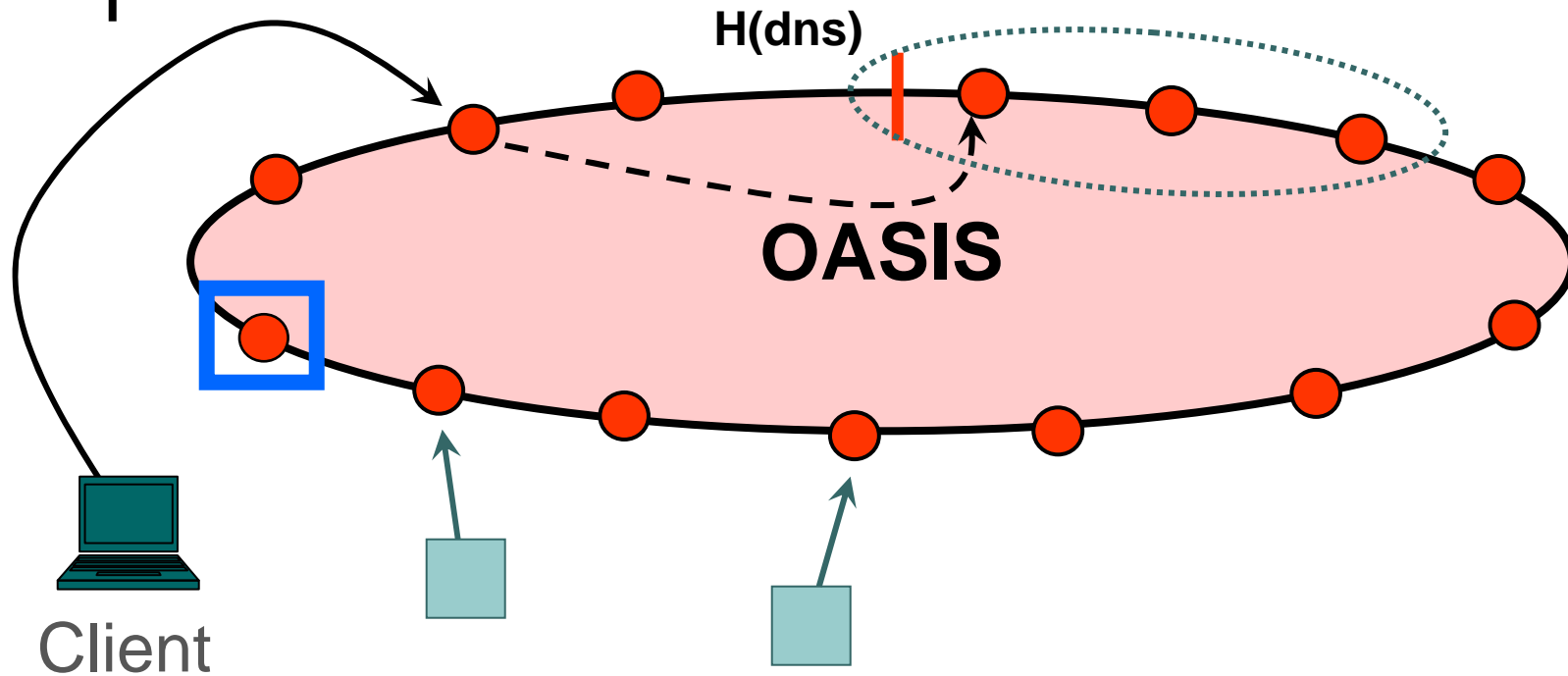
- ● ● | A client's view: Finding a nameserver



- Core lookup: Contacts 1 of 13 nameservers for *.nyuld.net*
  - OASIS “uses itself” to discover replica for service *dns*

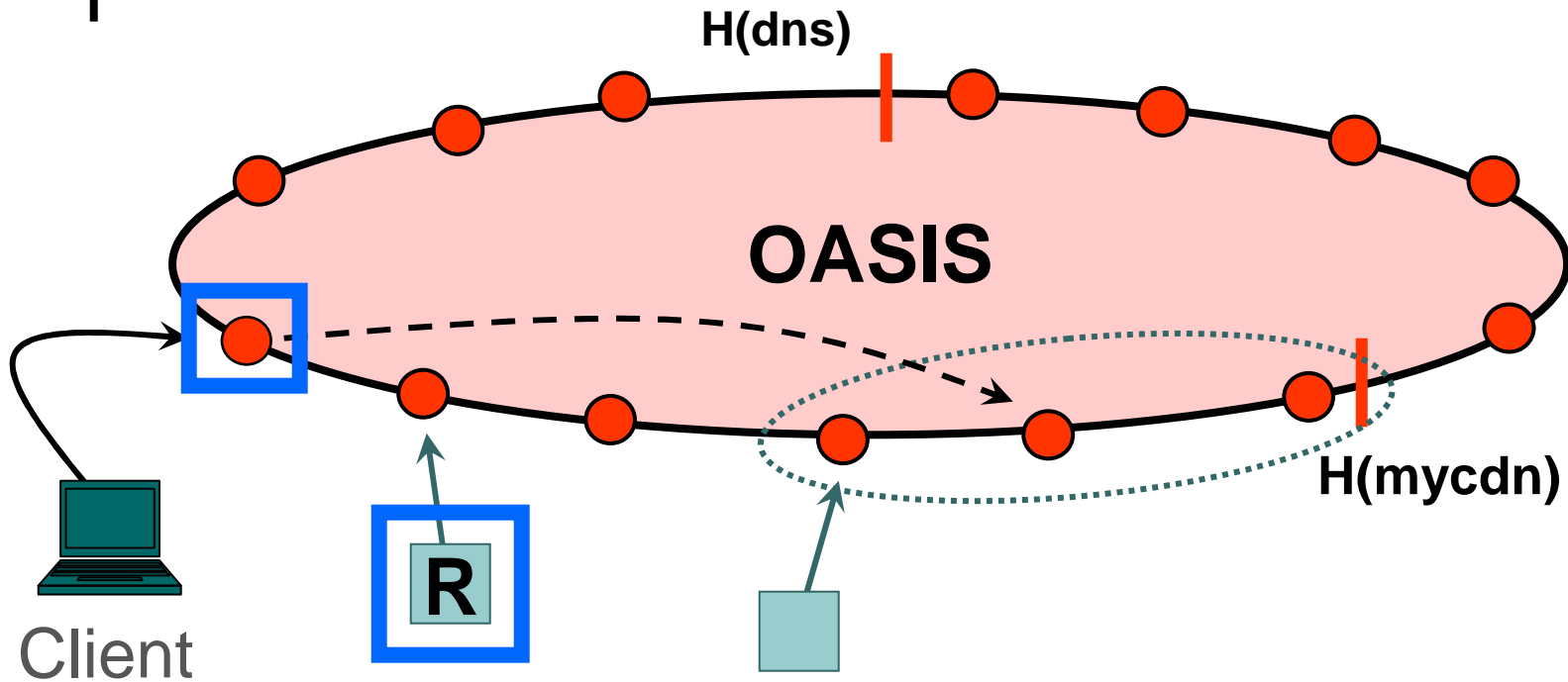


# ••• A client's view: Finding a nameserver



- Core lookup: Contacts 1 of 13 nameservers for *.nyuld.net*
  - OASIS “uses itself” to discover replica for service *dns*
  - Returns nearby nameservers for subsequent requests

● ● ● | A client's view: Finding a replica



- Replica lookup: Client contacts nearby nameserver
  - OASIS discover replica for service *mycdn*
  - Returns nearby replicas for application

# ● ● ● | Evaluation

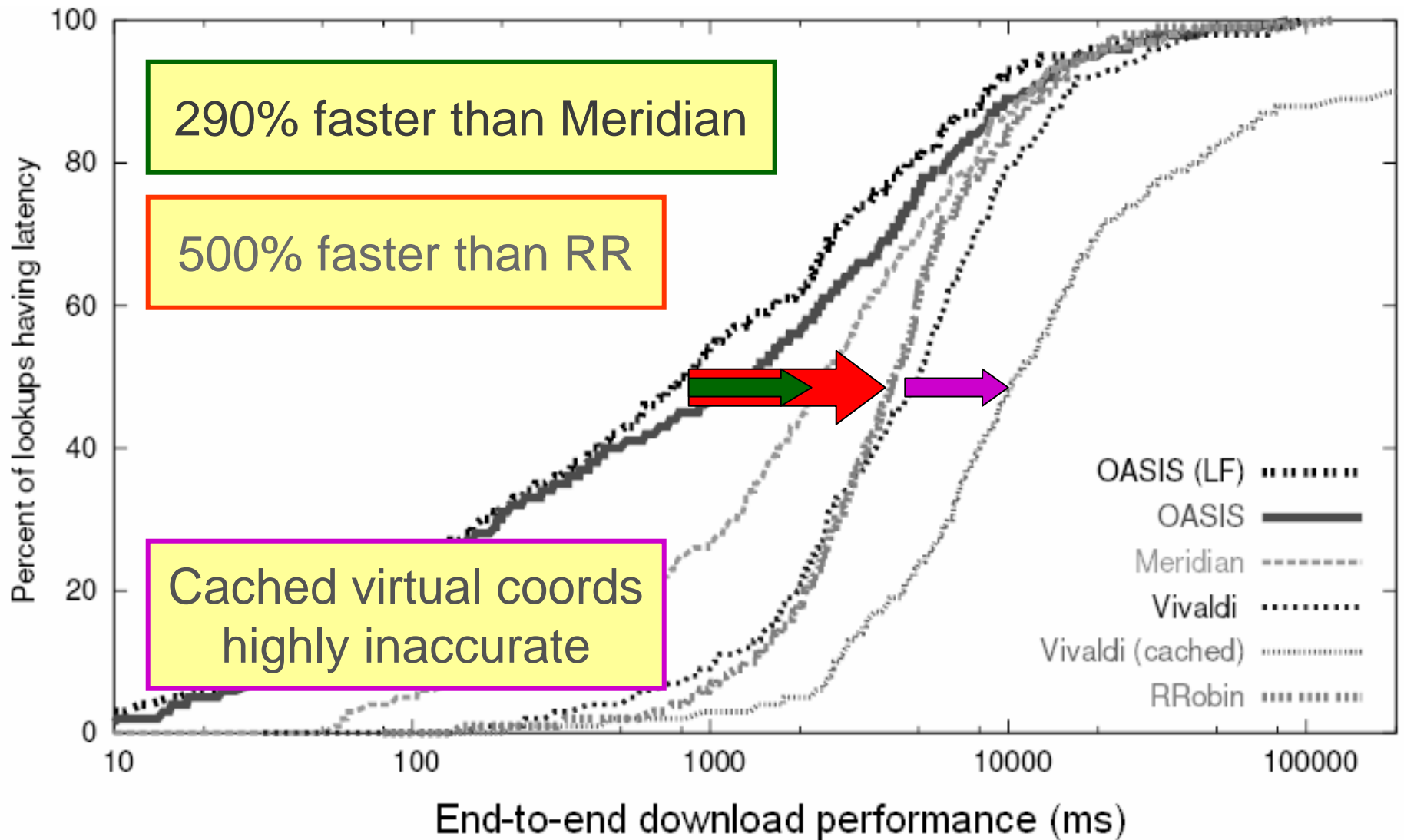
*Powered  
By*



PLANETLAB

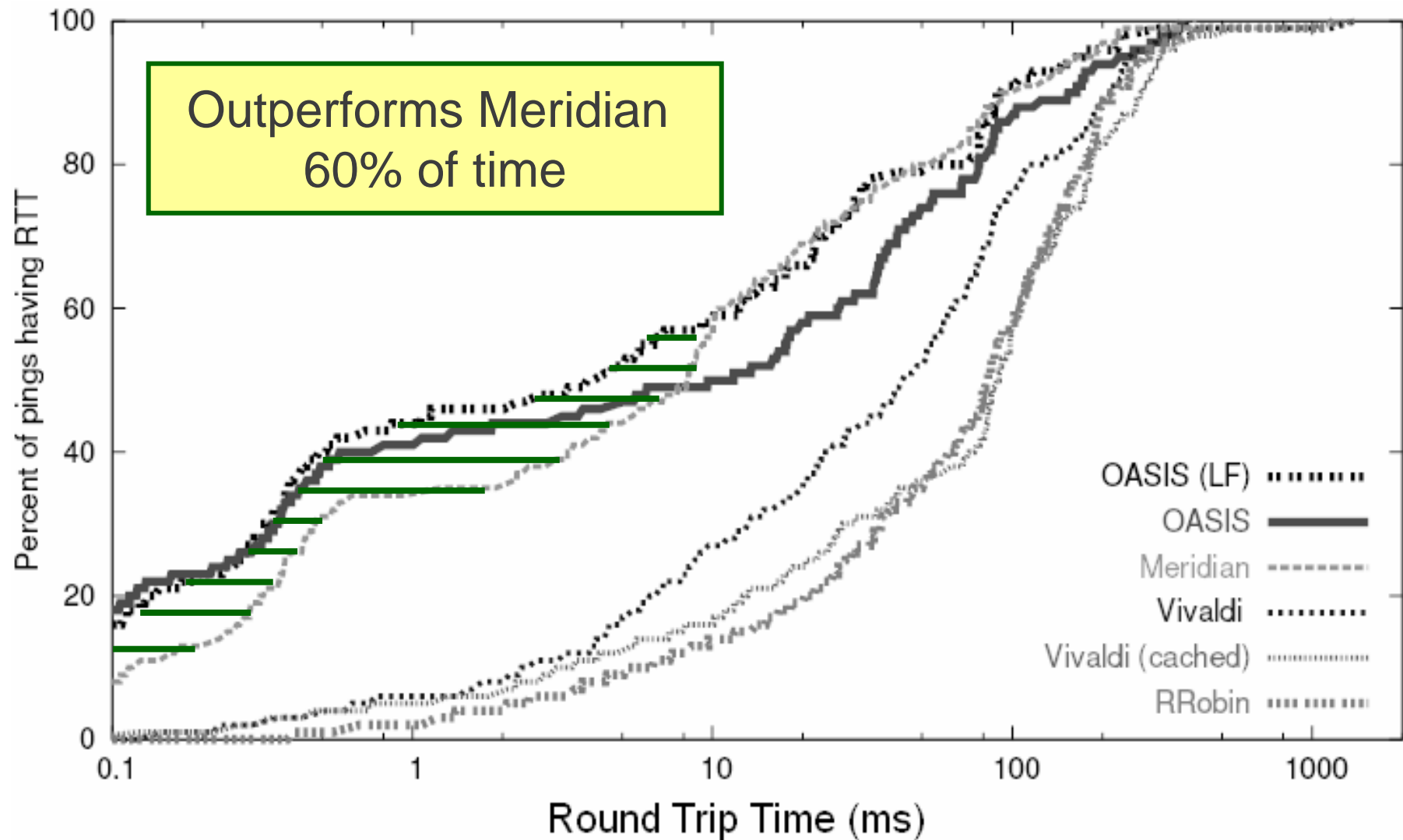
- Deployed on PlanetLab since November 2005
- How much end-to-end benefit from OASIS?
- How accurate is OASIS?
- Effective for load balancing?
- What are OASIS's bandwidth costs?

# E2E download of web page





# Client RTT to chosen replica



# ●●● | OASIS minimizes bandwidth spikes

95% bandwidth usage per replica (MB)

loc \ metric	CA	TX	NY	Germany
Latency Only	<b>23.3</b>	<b>0.0</b>	<b>0.0</b>	<b>0.0</b>
Load + Latency				

- 8 clients in CA repeatedly request 1 MB file
- Replicas report load as *log* (95% bandwidth per 1-min slot)

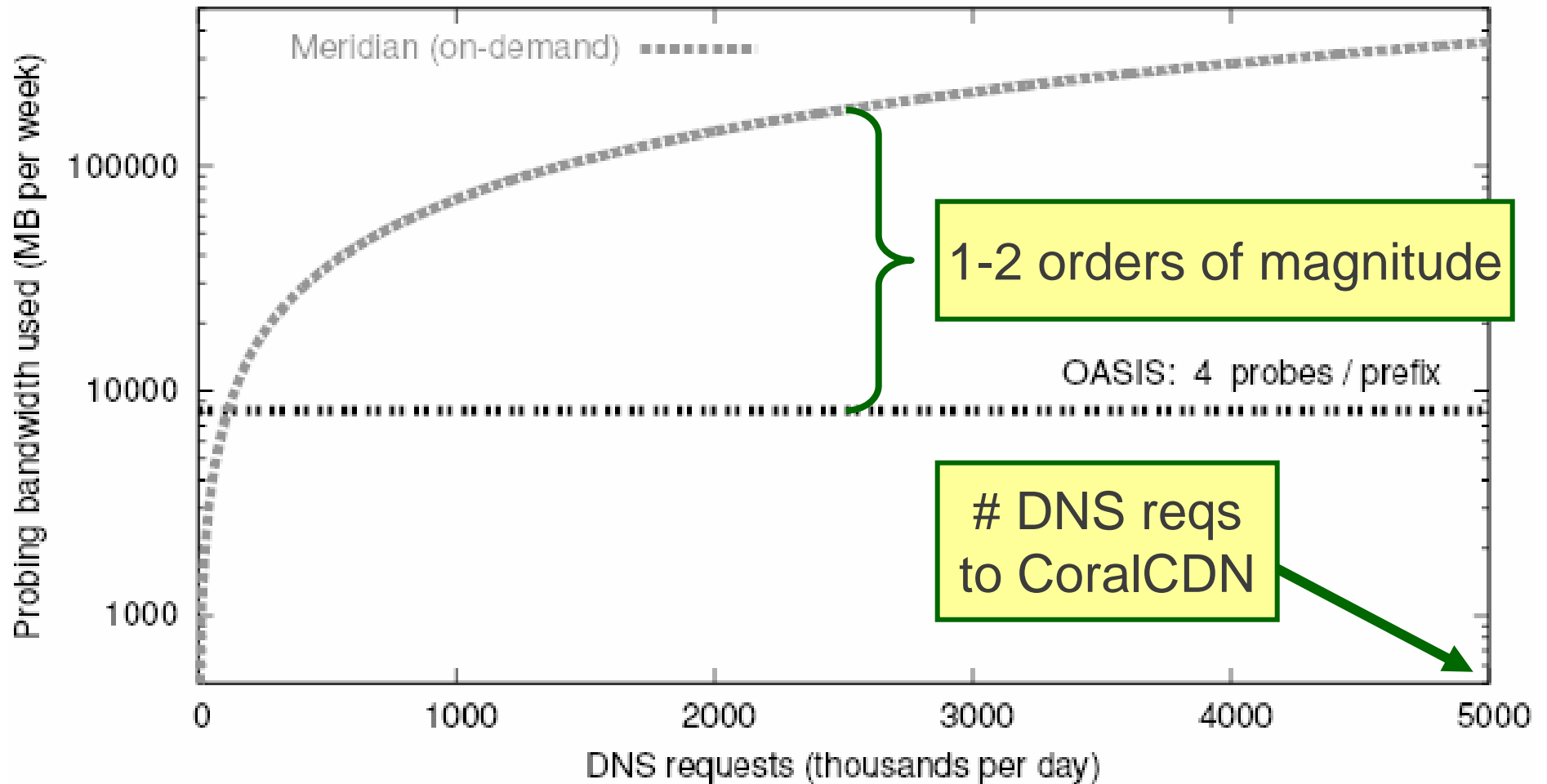
# ● ● ● | OASIS minimizes bandwidth spikes

95% bandwidth usage per replica (MB)

loc \ metric	CA	TX	NY	Germany
Latency Only	<b>23.3</b>	<b>0.0</b>	<b>0.0</b>	<b>0.0</b>
Load + Latency	<b>9.0</b>	<b>11.3</b>	<b>9.6</b>	<b>9.2</b>

- 8 clients in CA repeatedly request 1 MB file
- Replicas report load as *log* (95% bandwidth per 1-min slot)

# Bandwidth costs: OASIS v. on-demand



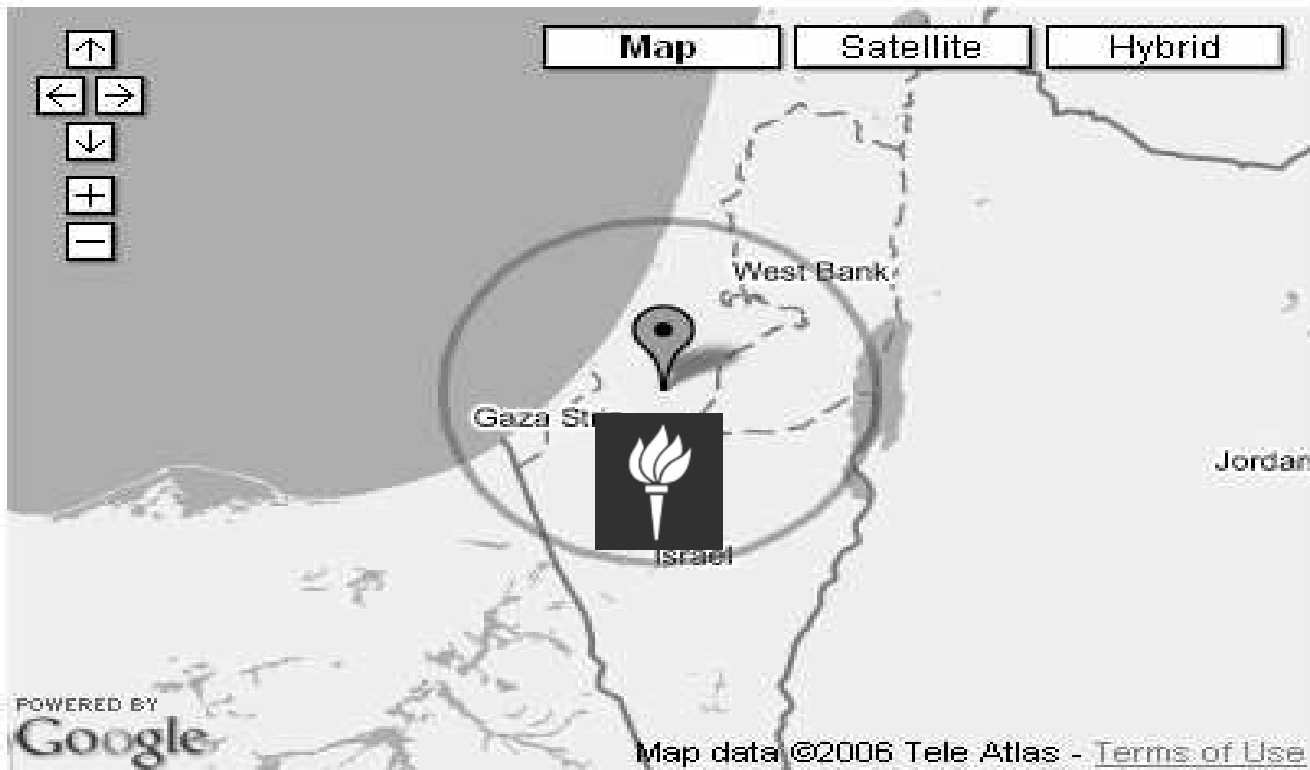




# Outline

- Architecture and design decisions
- Detailed design
- Evaluation
- Deployment and integration lessons
  - OASIS deployed since November 2005
  - Currently in use by 10 services

- ● ● | Sanity check for network peculiarities
  - Employ measurement redundancy
  - Easy visualization significantly helped debugging



Enter IP / host to locate:

# ••• Netops have low tolerance for probing

- Probing generates abuse complaints
- Your service can get blacklisted!

Keyword	Threads	Msgs	Keyword	Threads	Msgs
abuse	198	888	ICMP	64	308
attack	98	462	IDS	60	222
blacklist	32	158	intrusion	14	104
block	168	898	scan	118	474
complaint	216	984	trojan	10	56
flood	4	30	virus	24	82

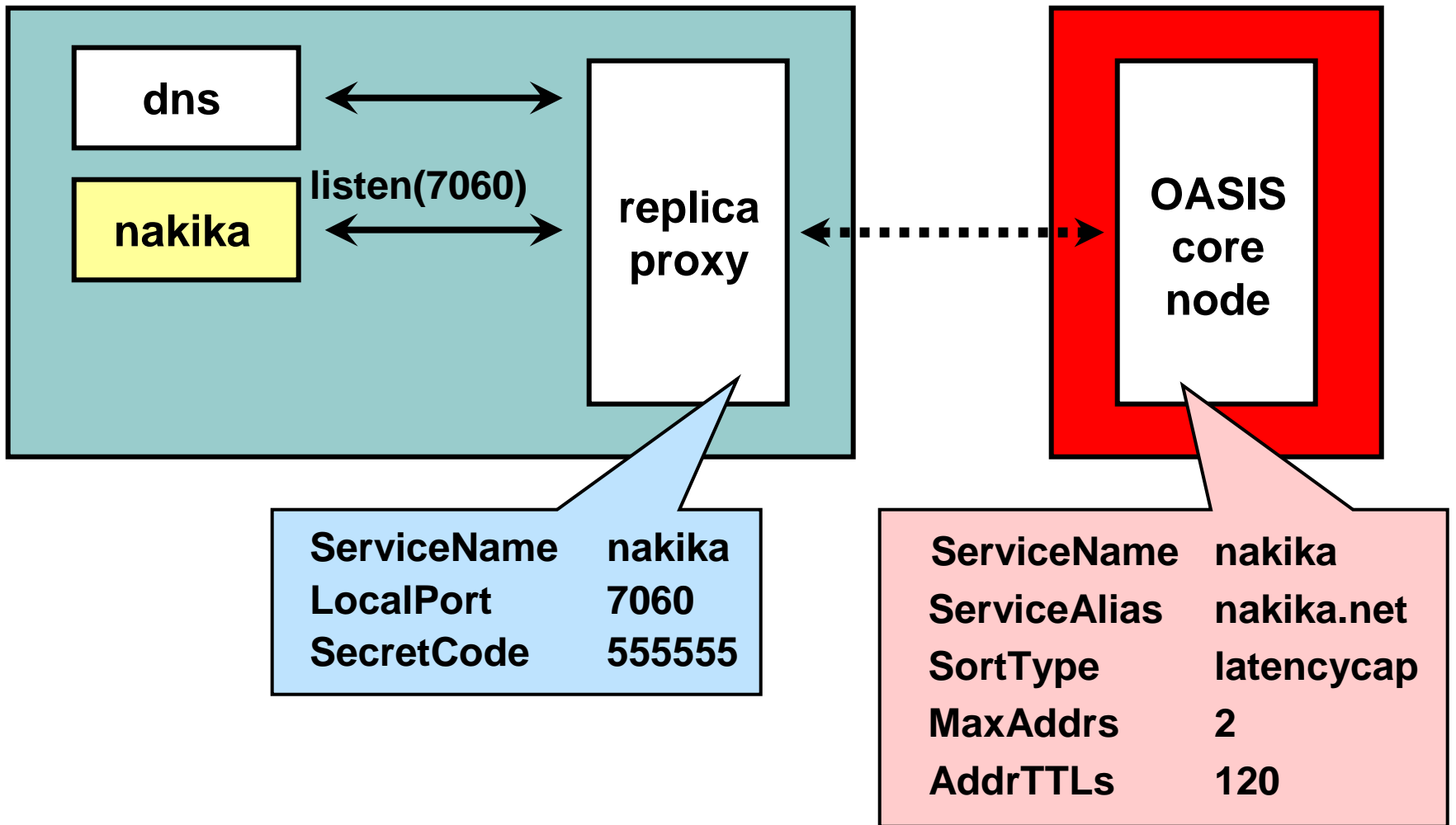
Keyword frequency on PlanetLab support lists

9 months, 1820 threads, 4682 msgs

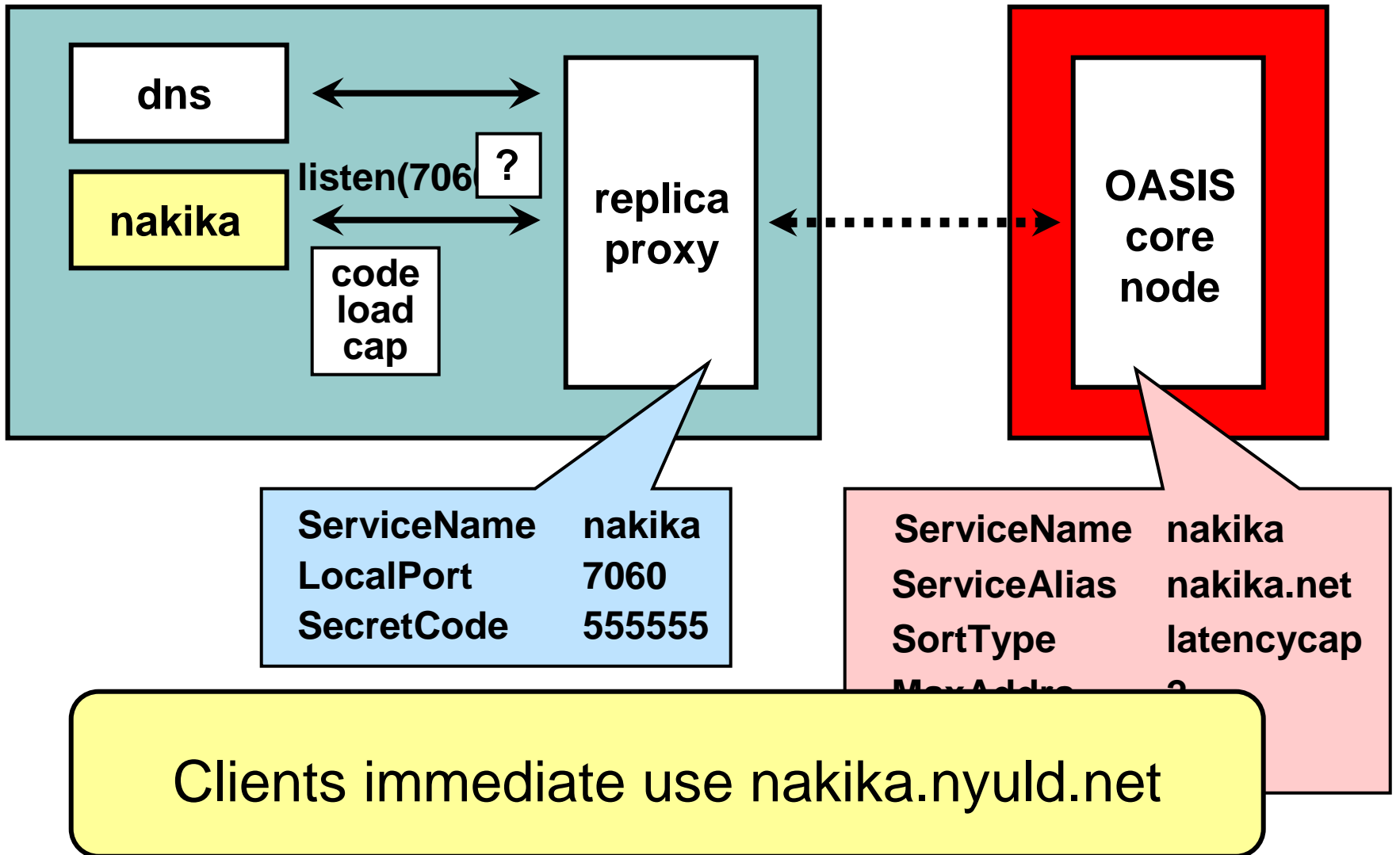
# • • • | Netops have low tolerance for probing

- Be careful *what* you probe
  - Probe slowly and rarely
  - No random ports or obvious attack vectors (TCP port 22/23)
- Be careful *whom* you probe
  - Check blacklist for netblock *and* target IP (after traceroute)

- Make it easy to integrate



- Make it easy to integrate



# ● ● ● | Current services using OASIS...

- **Chunkcast** block anycast (Berkeley)
- **CoralCDN** (NYU)
- **Na Kika** content distribution (NYU)
- **OASIS**
  - RPC, DNS, HTTP interfaces
- **OCALE** overlay convergence (Berkeley)
  - Separate services for client and server IPs gateways
- **OpenDHT** public DHT service (Berkeley)
- **OverCite** distributed library (MIT): Deployed on RON

# ••• | Summary

- OASIS is a general, open anycast service
  - Supports multiple services: more are better
  - Performs accurate server selection
  - Removes all on-demand probing
  - Provides easy integration
- Use OASIS for your distributed system!

[http://oasis.coralcdn.org/](http://<u>oasis.coralcdn.org/</u>)