Experiences with CoralCDN

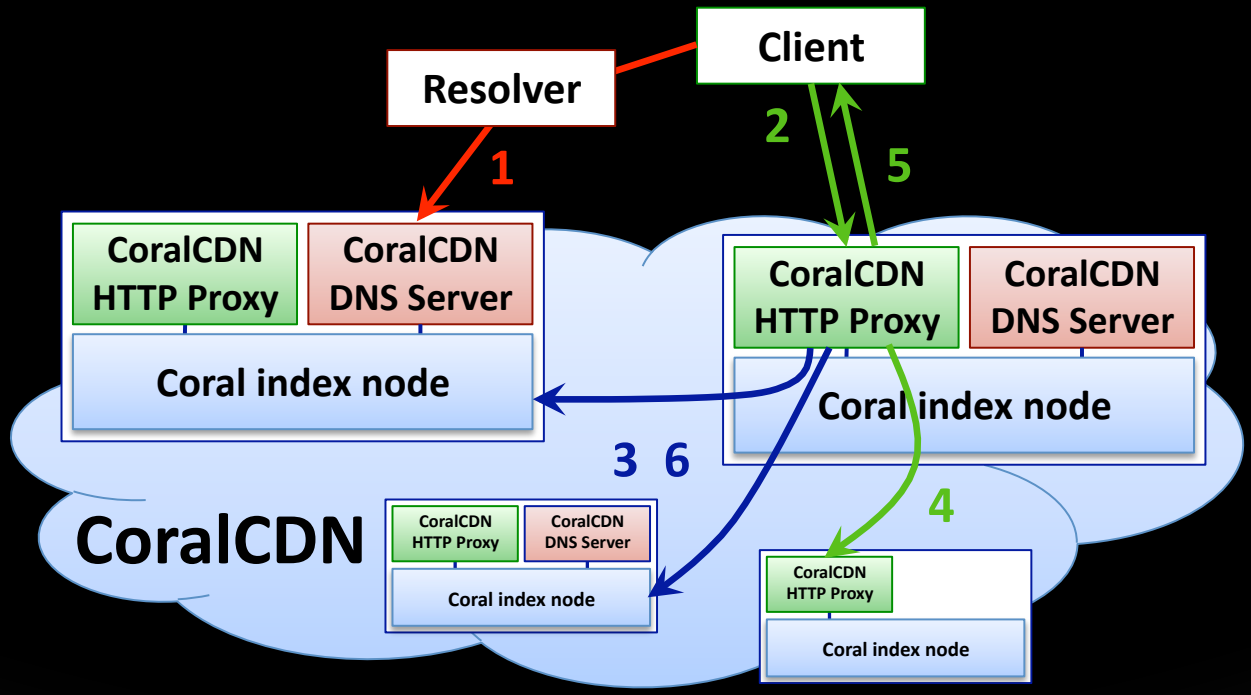
A Five-Year Operational View

Michael J. Freedman

Princeton University

www.coralcdn.org

A Cooperative, Self-Organizing CDN



Goal: To make desired content widely available regardless of publisher's own resources, by organizing and utilizing any cooperative resources

http://example.com/path

http://example.com.nyud.net/path

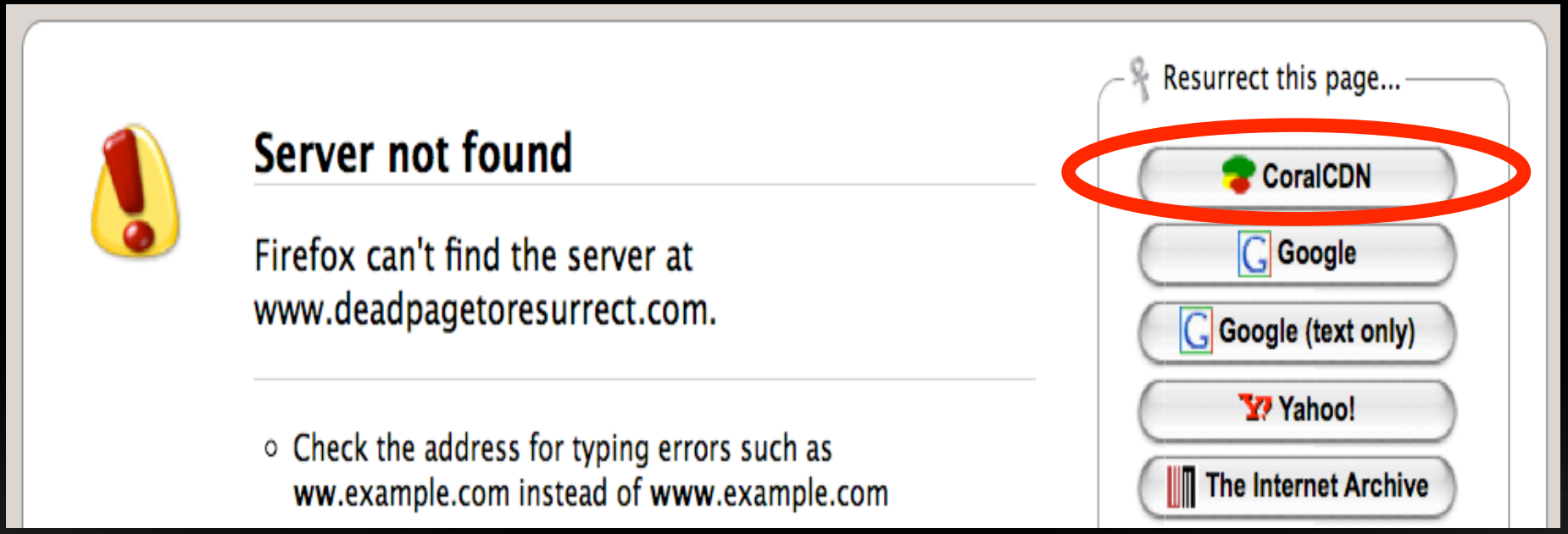
Adopted by:

Clients

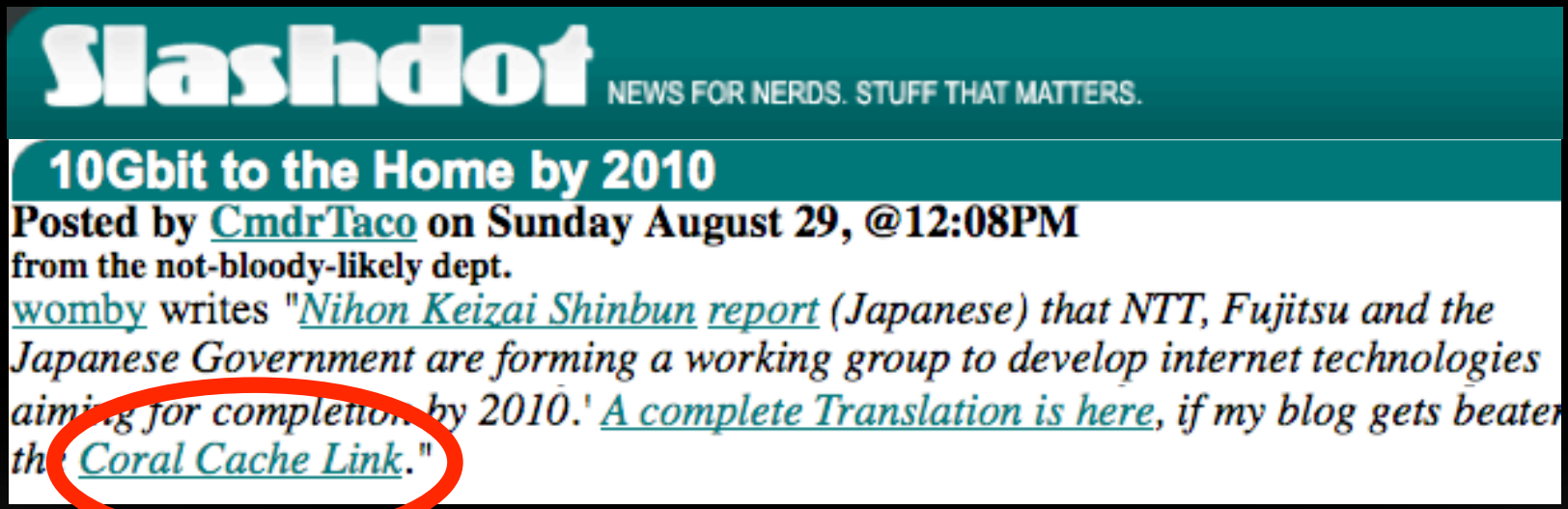
Servers

Third-parties

Many of you have used CoralCDN



Many of you have used CoralCDN

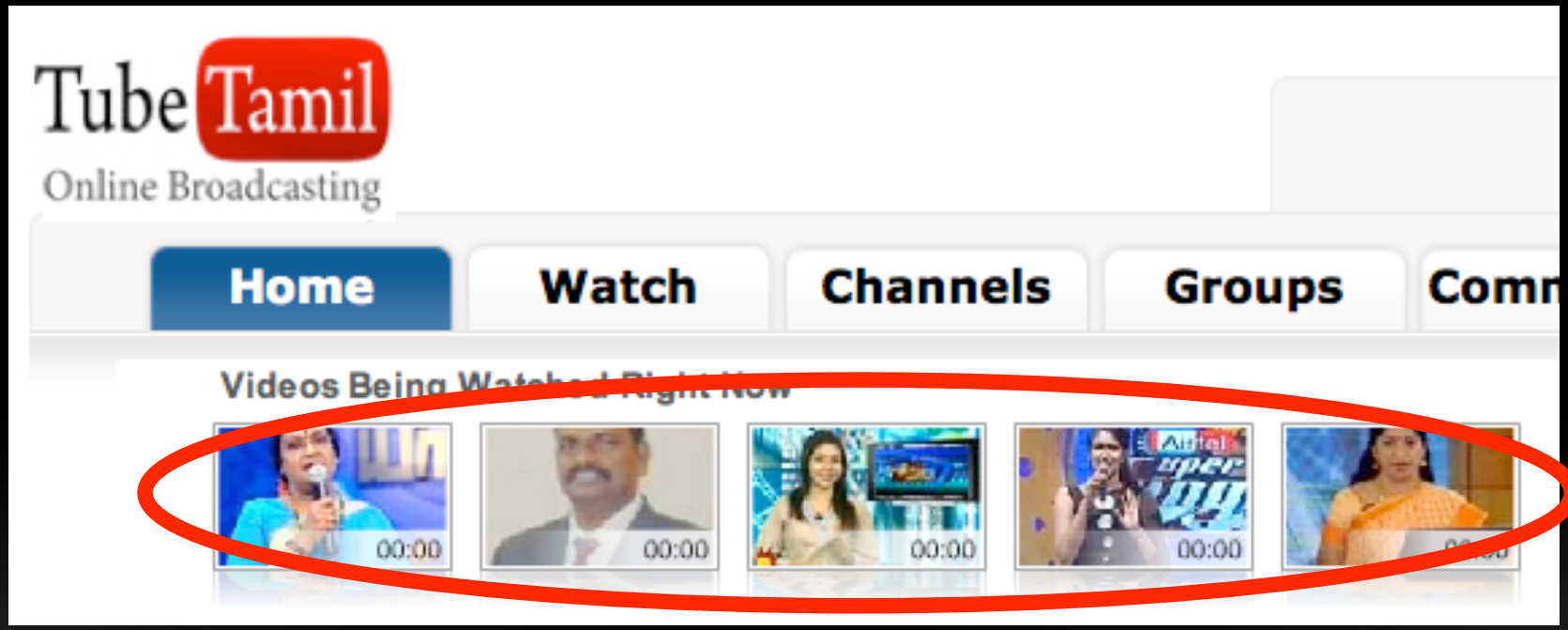

Slashdot NEWS FOR NERDS. STUFF THAT MATTERS.

10Gbit to the Home by 2010

Posted by CmdrTaco on Sunday August 29, @12:08PM
from the not-bloody-likely dept.

womby writes "*Nihon Keizai Shinbun report (Japanese) that NTT, Fujitsu and the Japanese Government are forming a working group to develop internet technologies aiming for completion by 2010.' A complete Translation is here, if my blog gets beaten the Coral Cache Link.*"

Many of you have used CoralCDN

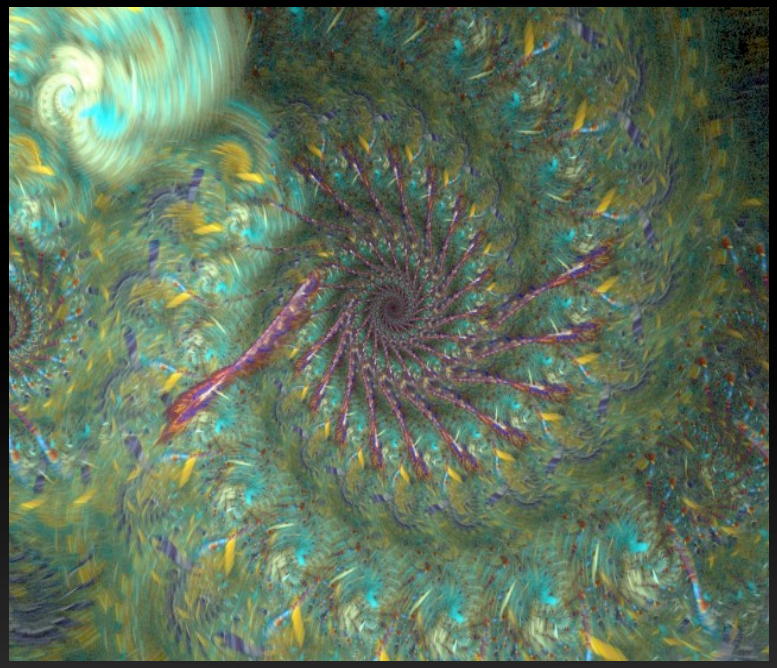


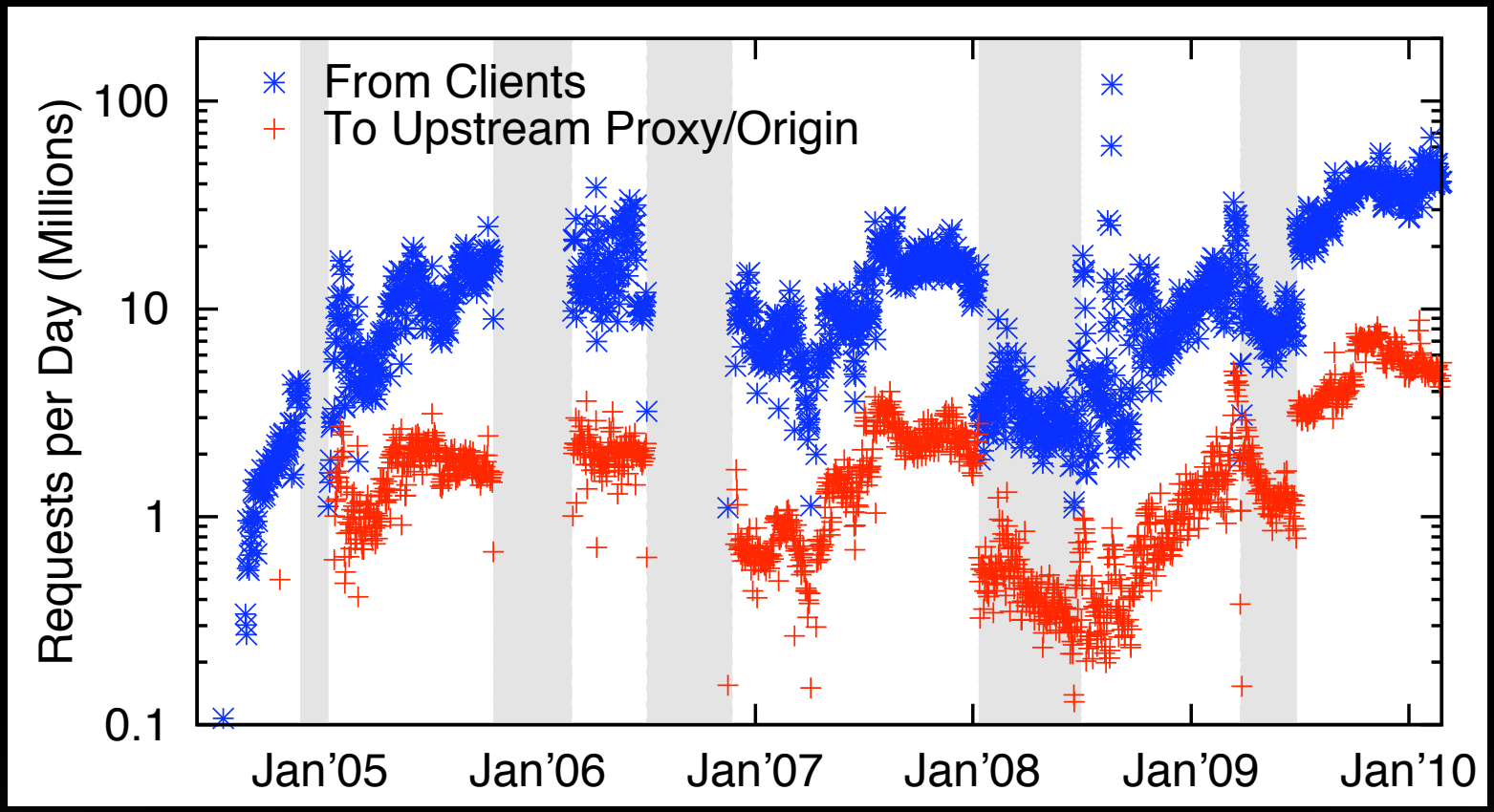
Many of you have used CoralCDN

Many of you have used CoralCDN

Many of you have used CoralCDN

electric sheep

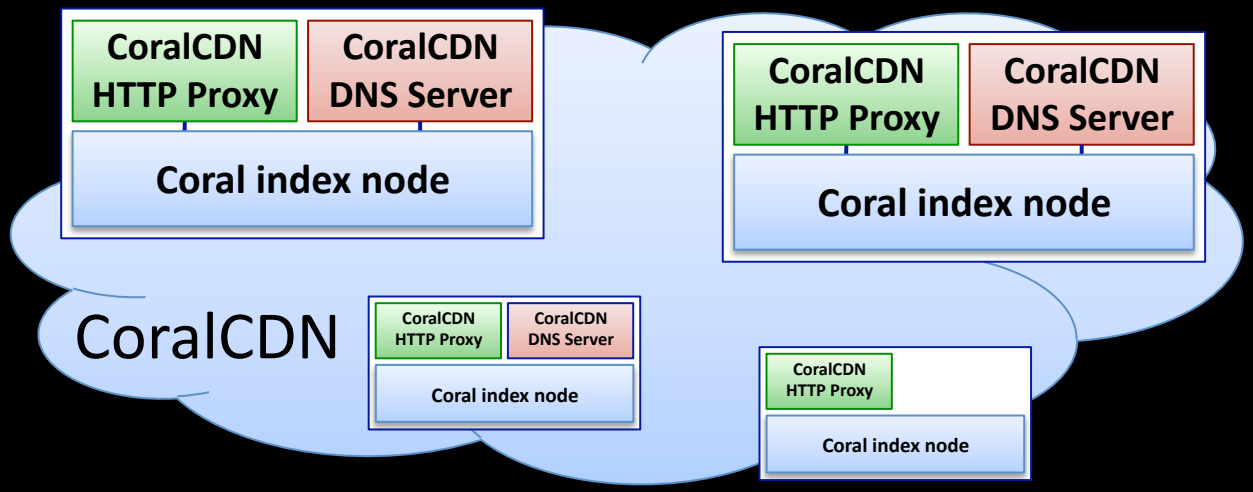




Daily Request Volume

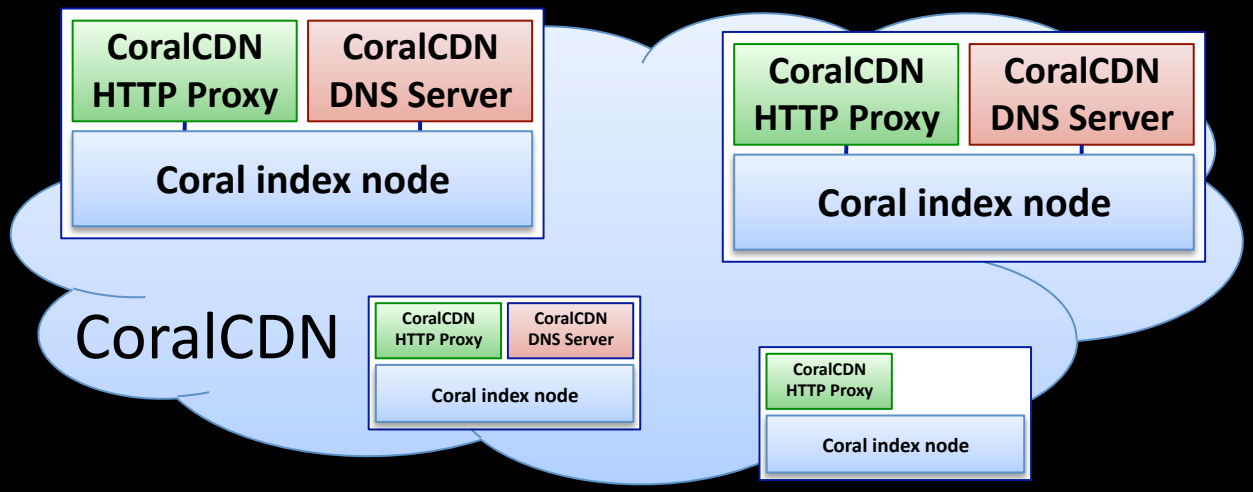
2M clients – 2 TB content – 20K origin domains

From 300-400 PlanetLab servers



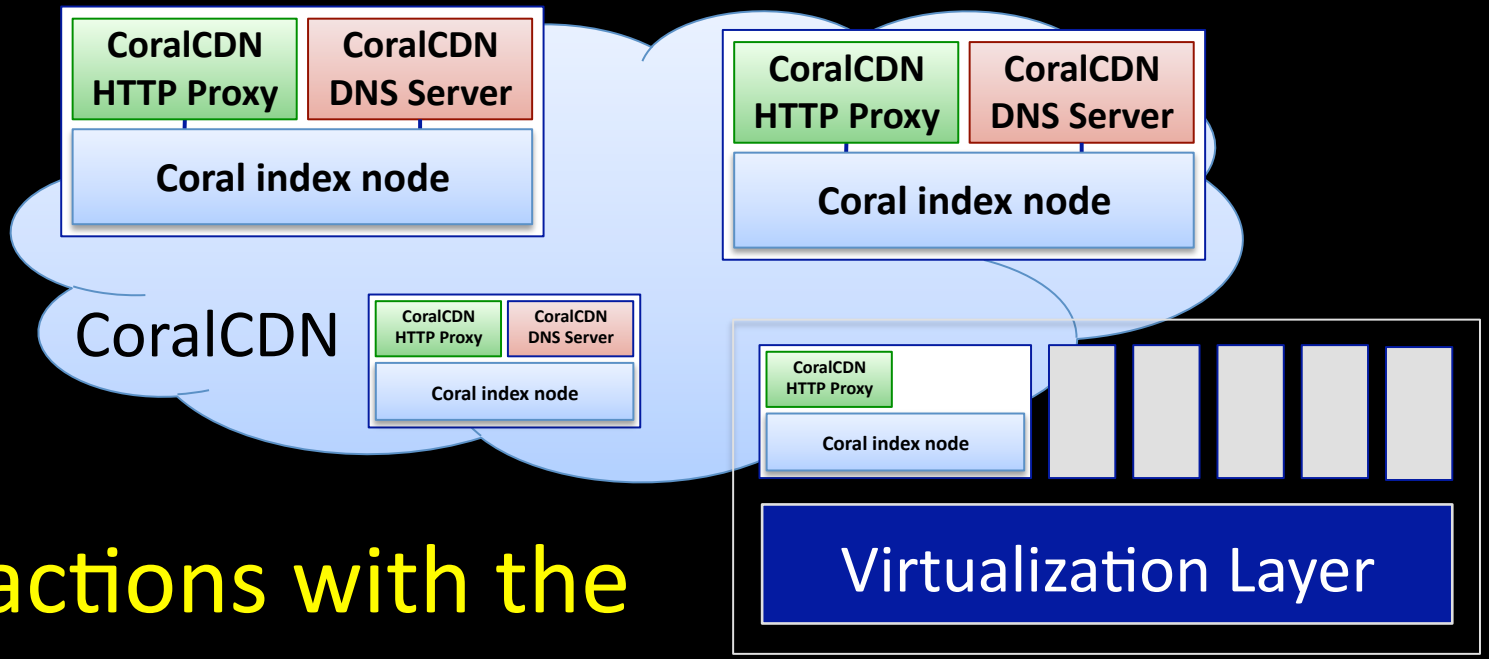
Based on peer-to-peer DHT

1. Weakened consistency + algorithms that prevent tree saturation during lookup
2. Decentralized clustering for locality and hierarchical lookup
3. Cooperative HTTP / DNS that leverages locality



Based on peer-to-peer DHT

1. Weakened consistency + algorithms that prevent tree saturation during lookup
2. Decentralized clustering for locality and hierarchical lookup
3. Cooperative HTTP / DNS that leverages locality



Interactions with the External Environment

Clients

Origin Domains

1. Experiences
   - Naming
   - Fault Tolerance
   - Resource management
2. Revisit CoralCDN's design

Naming

Flexible, open API

Mismatch with domain-based access control policies

CoralCDN's Platform-as-a-Service API

Rewrite rules in origin webservers

```
RewriteEngine on
RewriteCond %{HTTP_USER_AGENT} !^CoralWebPrx
RewriteCond %{QUERY_STRING} !(^|&)coral-no-serve$
RewriteRule ^(.*)$ http://%{HTTP_HOST}.nyud.net%
  {REQUEST_URI} [R,L]
```

CoralCDN's Platform-as-a-Service API

Rewrite rules in origin webservers

```
RewriteEngine on
RewriteCond %{HTTP_USER_AGENT} !^CoralWebPrx
RewriteCond %{QUERY_STRING} !(^|&)coral-no-serve$
RewriteCond %{HTTP_REFERER} slashdot\.org [NC]
RewriteCond %{HTTP_REFERER} digg\.com [NC,OR]
RewriteCond %{HTTP_REFERER} blogspot\.com [NC,OR]
RewriteRule ^(.*)$ http://%{HTTP_HOST}.nyud.net%
  {REQUEST_URI} [R,L]
```

Sites integrate with load/bandwidth monitoring

Naming Conflation

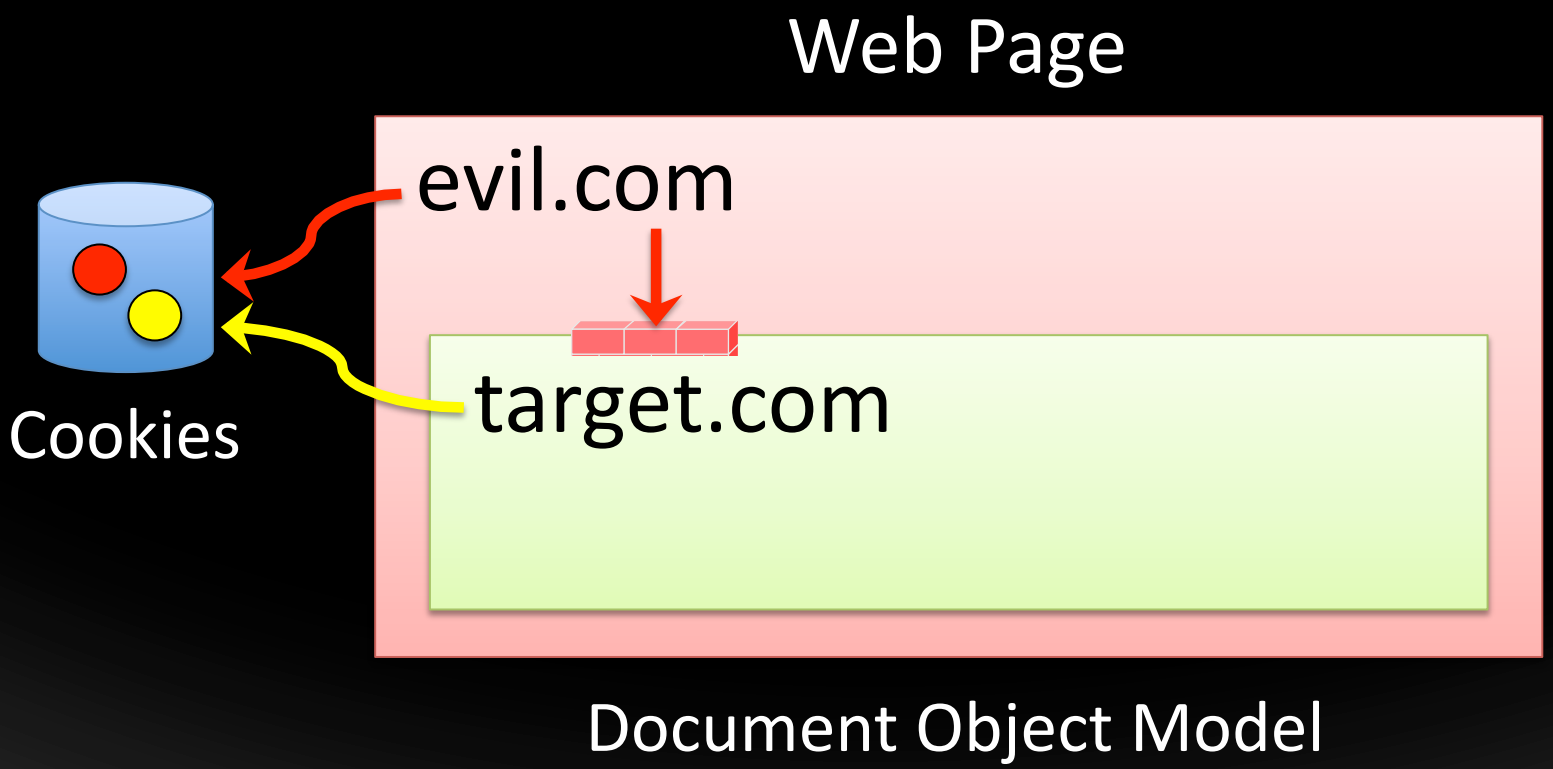
http://domain.service1.service2/path

✓ 1. Location to retrieve content

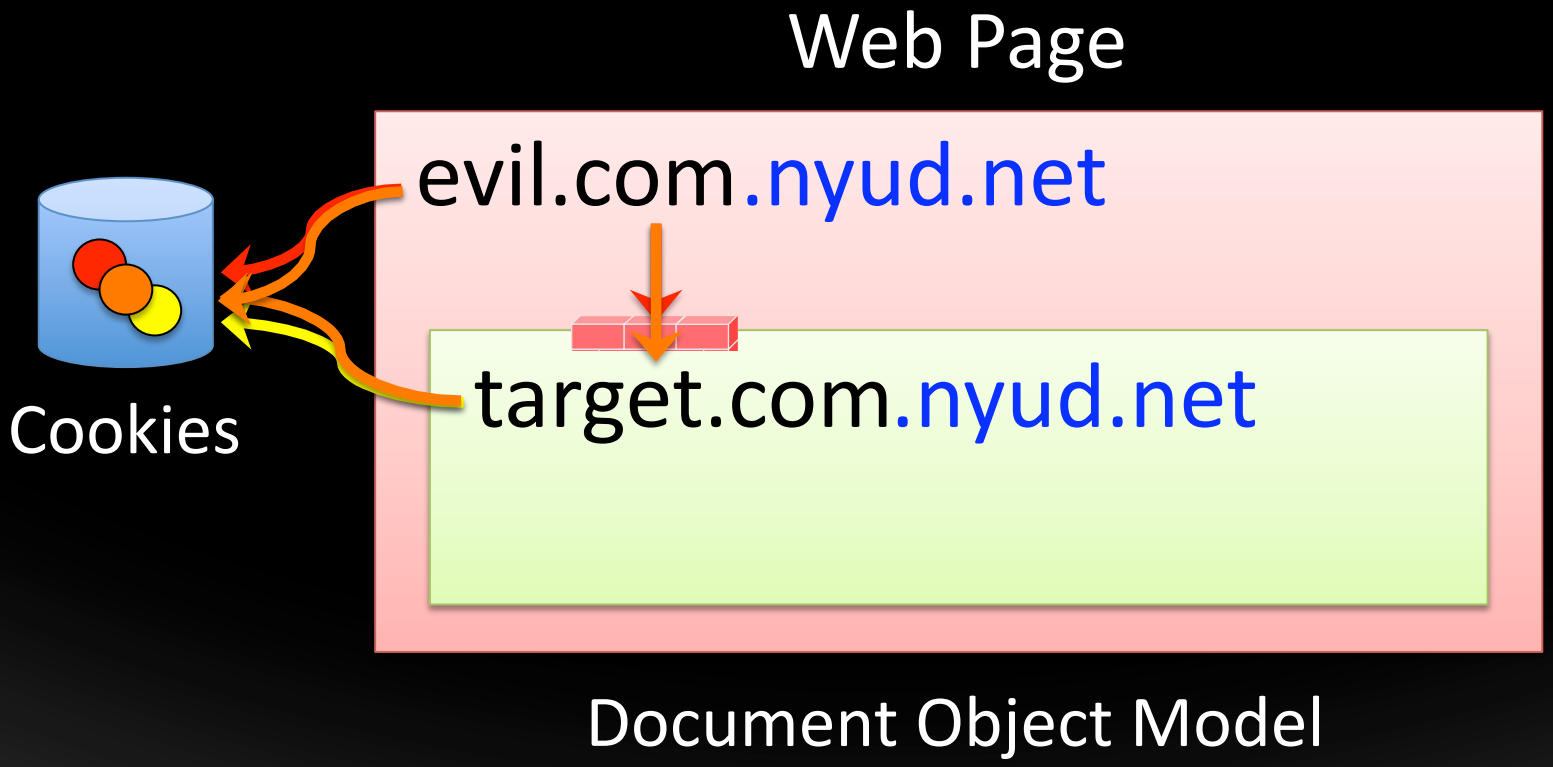
✗ 2. Human-readable name for administrative entity

✗ 3. Security policies to govern objects' interactions

Domain-based Security Policies



Domain-based Security Policies



Defaults violate least privilege

Fault Tolerance: Failure Decoupling

Internal failures:

- DHT nodes
- DNS servers, HTTP proxies
- Management service

External failures:

- Decouple IPs from hosts
- Interactions with origin sites

Slashdot happens!

Origin Status	CoralCDN Reaction
1. Unresponsive	• Cache negative results
2. Returns error code	• Serve stale content
3. Reply truncated	• Use whole-file overwrites

Slashdot happens!

Origin Status	CoralCDN Reaction
1. Unresponsive	• Cache negative results
2. Returns error code	• Serve stale content
3. Reply truncated	• Use whole-file overwrites

Maintain status quo unless improvements are possible

What is “failure”?

Return values should have fail-safe defaults

Resource Management

Control over bandwidth consumption

Control and visibility into environment's resources

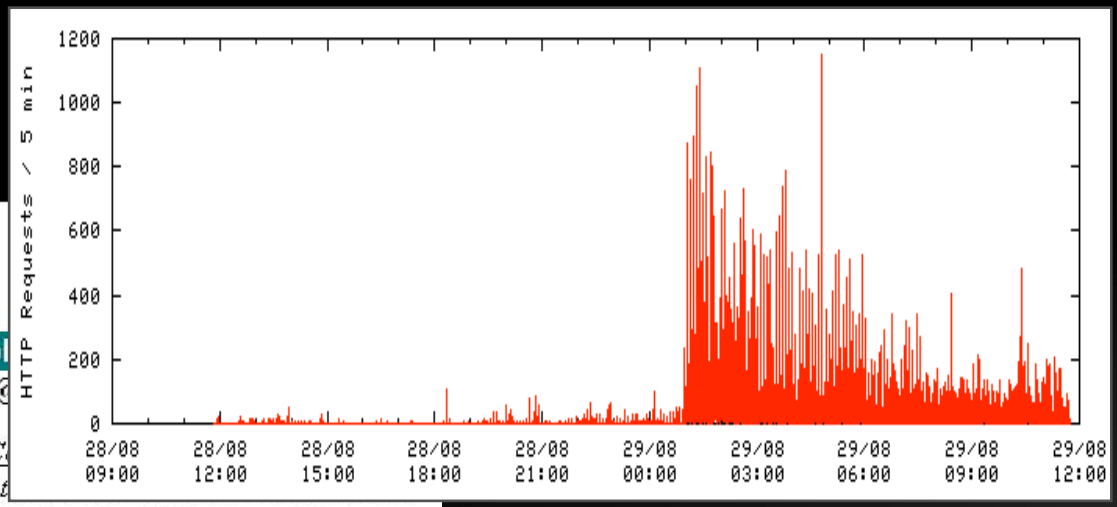
Some timeline…

Mar 2004
CoralCDN
released on
PlanetLab

Some timeline…

Mar 2004
CoralCDN released on PlanetLab

Aug 2004
Slashdotted



Slashdot
News for Nerds. Stuff that matters.

Login
Why Login?
Why Subscribe?

Sections
Main
Apache
Apple
2 more
AskSlashdot
2 more

IT: Coral P2P Cache Enters Publ

Posted by timothy on Saturday August 28, @
from the will-it-scale dept.

Eloquence writes *"infoAnarchy reports that C… system, has gone into public beta. Currently t… Planet-Lab, a large scale distributed research network of 400 servers. You can use Coral right now by appending "nyud.net:8090" to a hostname. View Slashdot through Coral. Is this the end of the Slashdot effect?"*

(Read More... | 45 of 60 comments | it.slashdot.org)

Some timeline…

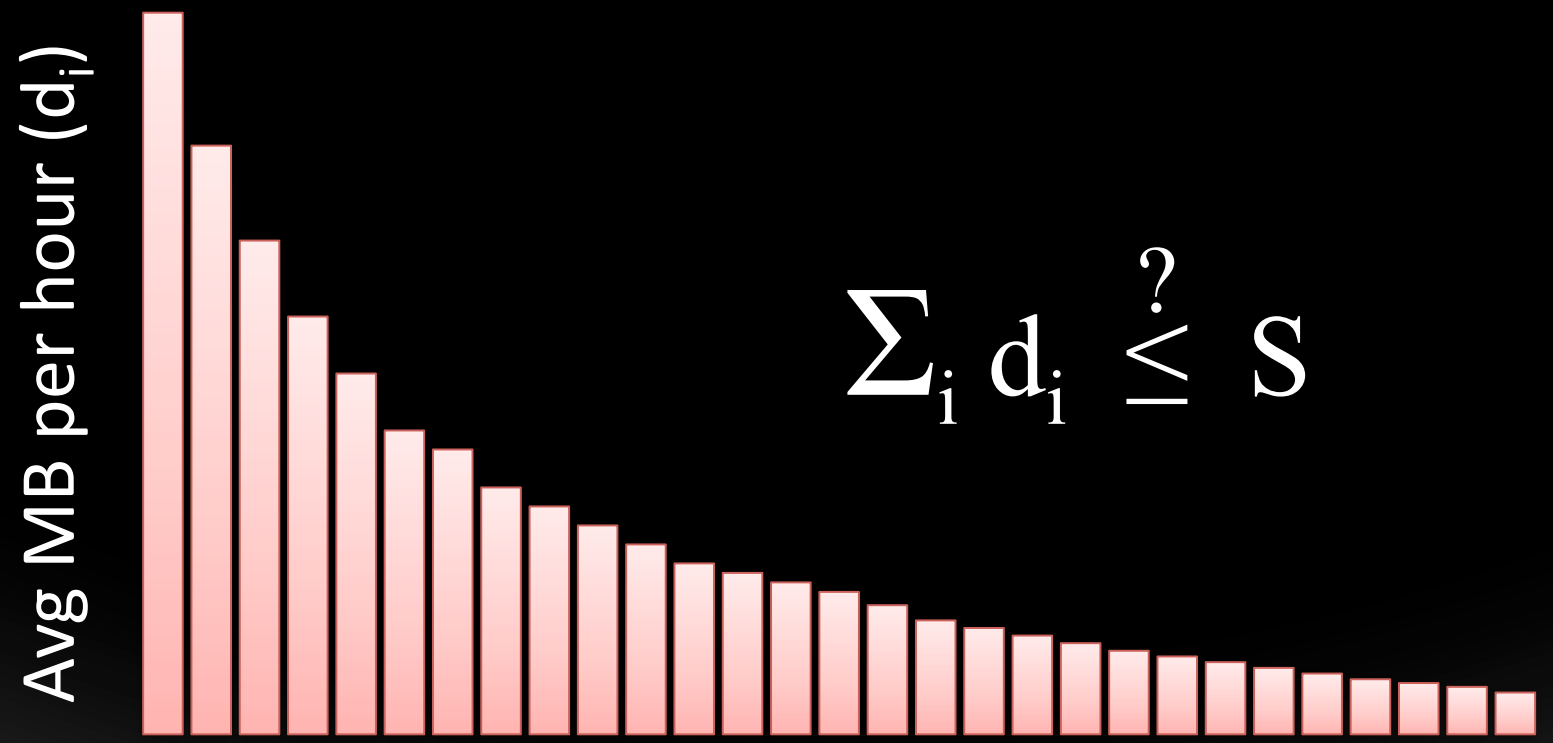
Mar 2004
CoralCDN released on PlanetLab

Aug 2004
Slashdotted

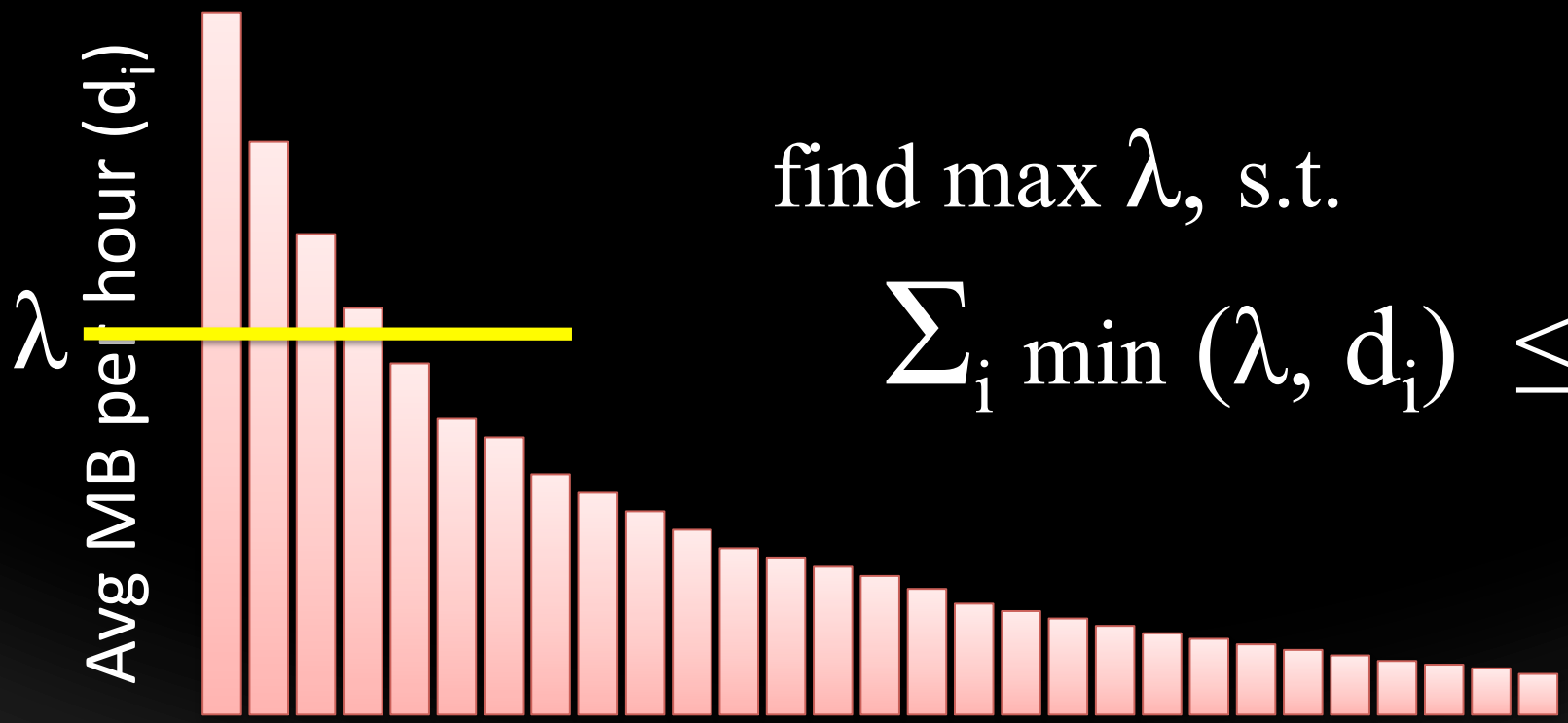
Dec 2004
Asian Tsunami

1. PlanetLab traffic jumps
2. Site threatens to yank PL
3. PL admin kills slice
4. Slice restored next day
5. Initiates discussion of resource limits for slices

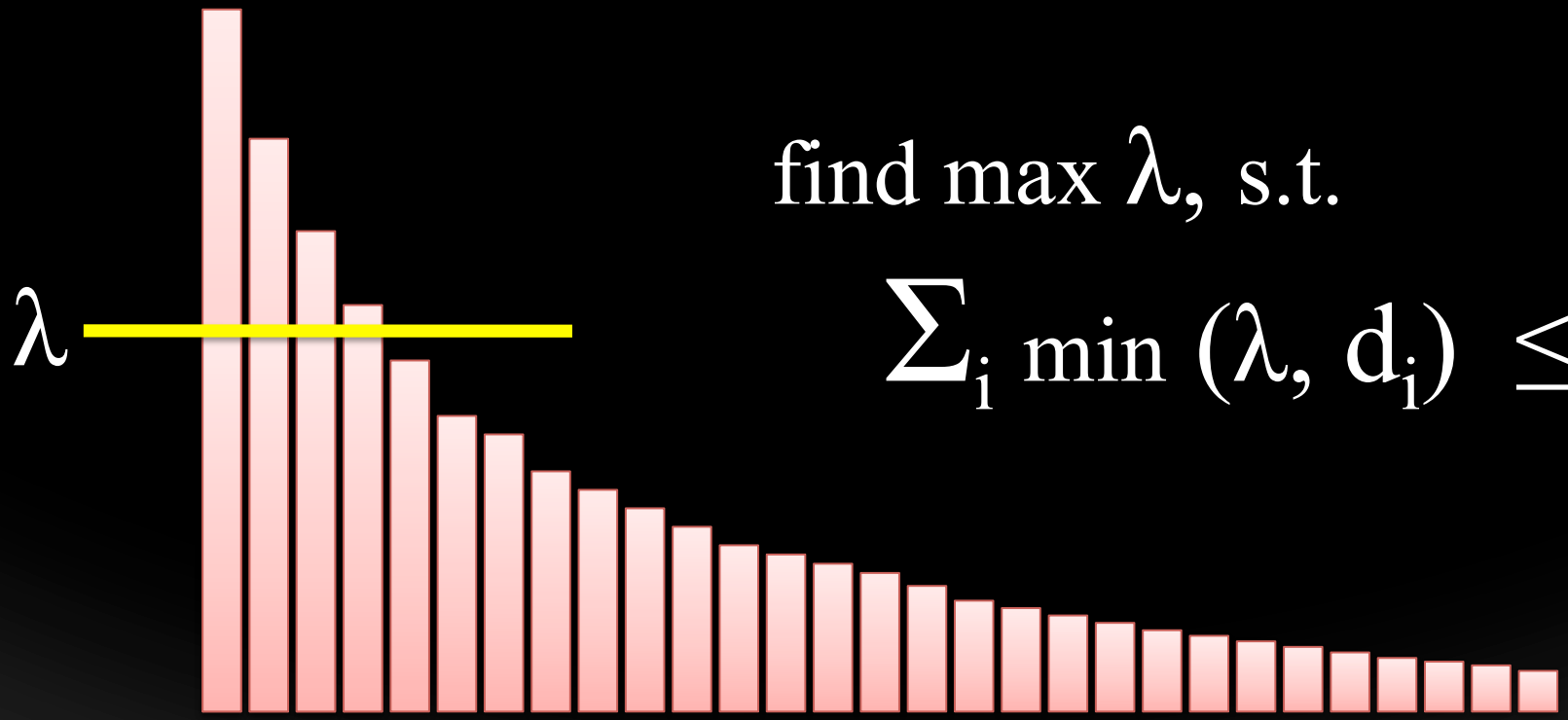
Demand >> Supply:
Enter Fair-Sharing Algorithms



Demand >> Supply:
Enter Fair-Sharing Algorithms



Demand >> Supply:
Enter Fair-Sharing Algorithms



$$\text{find max } \lambda, \text{ s.t.}$$

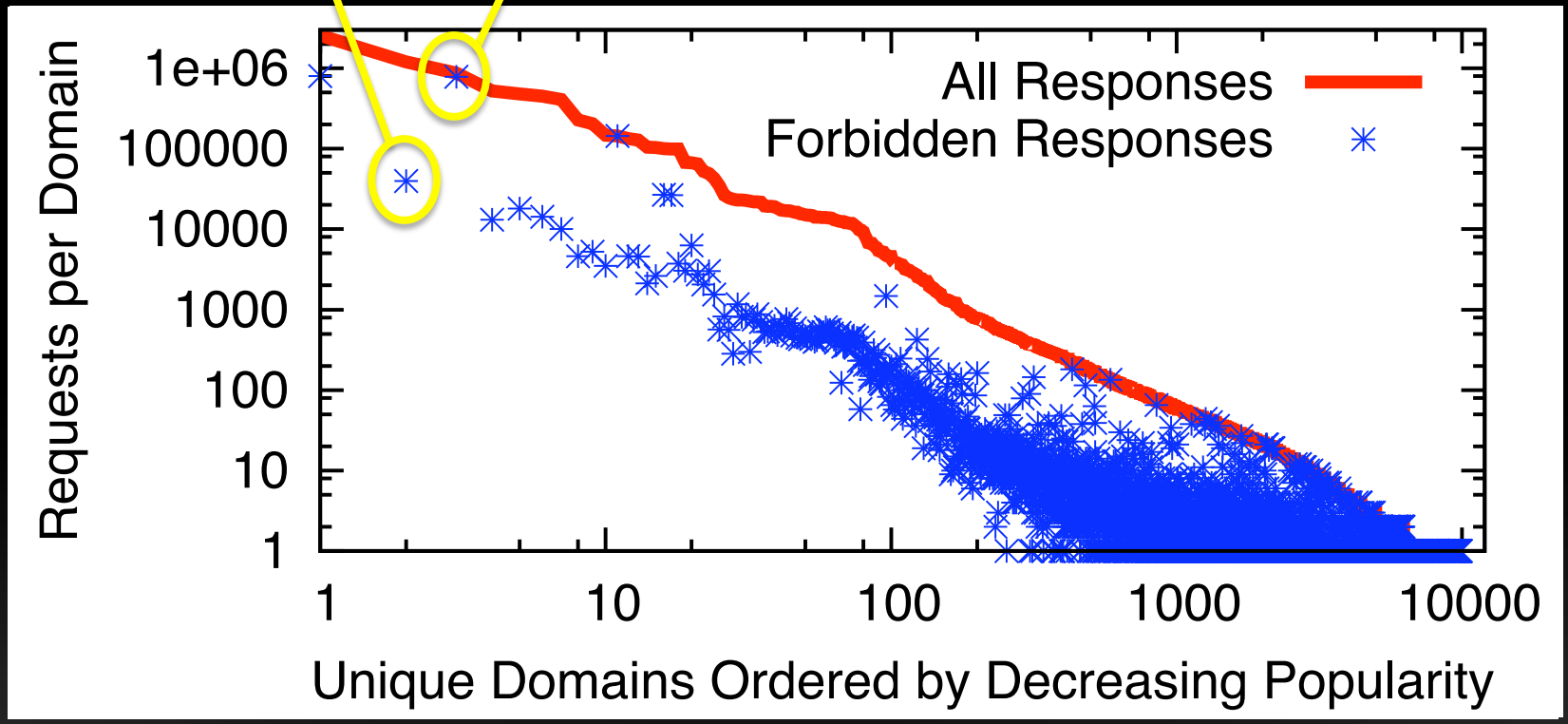
$$\sum_i \min(\lambda, d_i) \leq S$$

Domains with heaviest consumption

Admission Control under Fair-Sharing

~10 kB imgs
3.3% rejected

~5 MB videos
89% rejected



Demand > 10 TB Supply ≤ 2 TB

Some timeline…

Mar 2004
CoralCDN released on PlanetLab

Aug 2004
Slashdotted

Dec 2004
Asian Tsunami

Mar 2006
PL deploys bandwidth throttling

1. PlanetLab traffic jumps
2. Site threatens to yank PL
3. PL admin kills slice
4. Slice restored next day
5. Initiates discussion of resource limits for slices

Resource Management: Us vs. Them

Application Hammer

- Track HTTP traffic
- If site > fair share rate, reject via HTTP 403
- If total > peak rate, close server socket

Platform Hammer

- Track all network traffic
- If total > 80% daily rate, BW shaping in kernel

Resource Management: Us vs. Them

Application Hammer

- Track HTTP traffic
- If site > fair share rate, reject via HTTP 403
- If total > peak rate, close server socket

Platform Hammer

- Track all network traffic
- If total > 80% daily rate, BW shaping in kernel

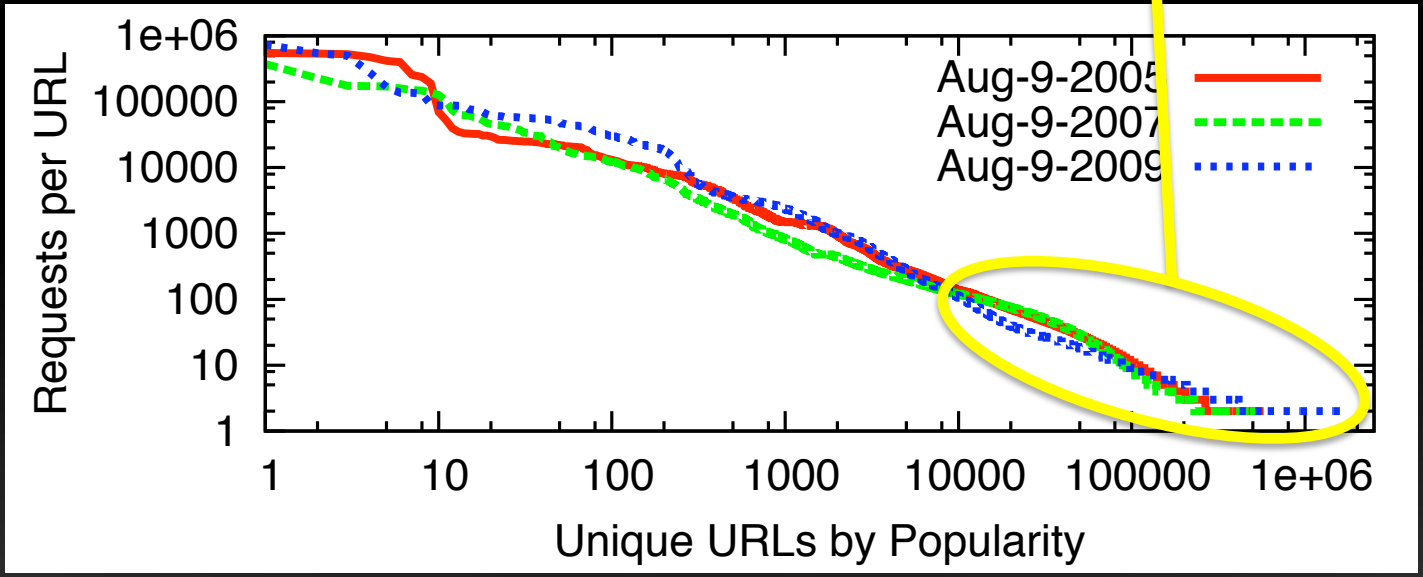
Result: HTTP traffic is 1/2 - 2/3 of all traffic

Lower layers should expose greater visibility and control over resources

1. Experiences
   - Naming
   - Fault Tolerance
   - Resource management
2. Revisit CoralCDN's design

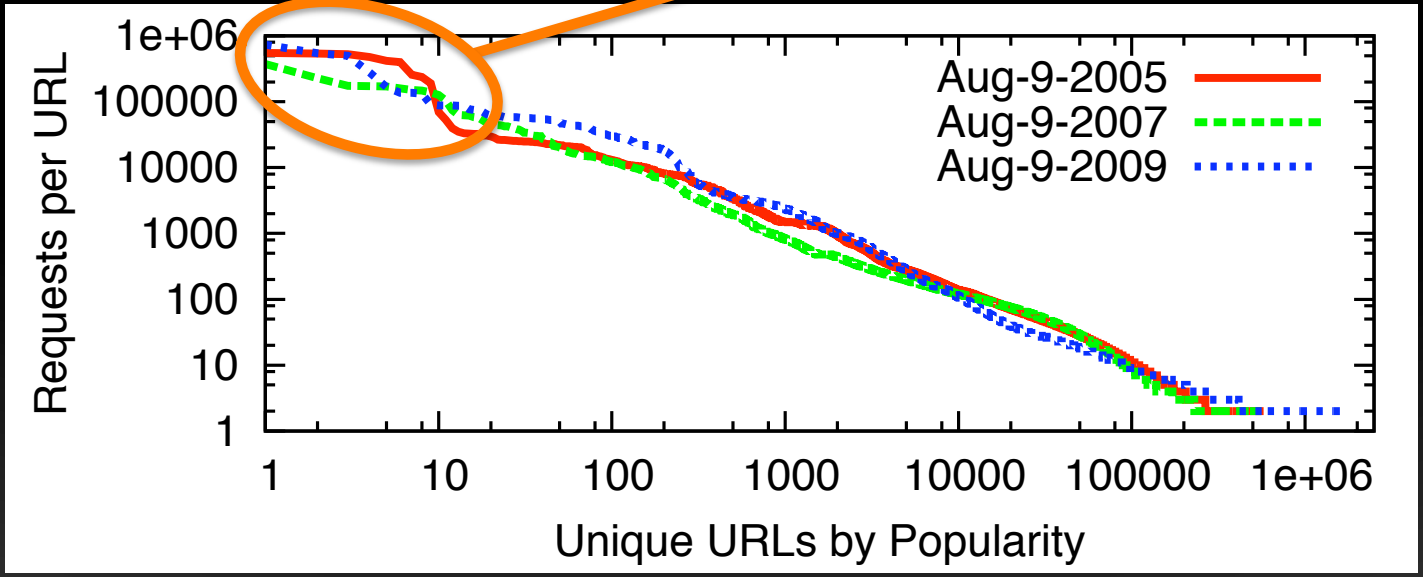
Usage Scenarios

1. Resurrecting old content
2. Accessing unpopular content



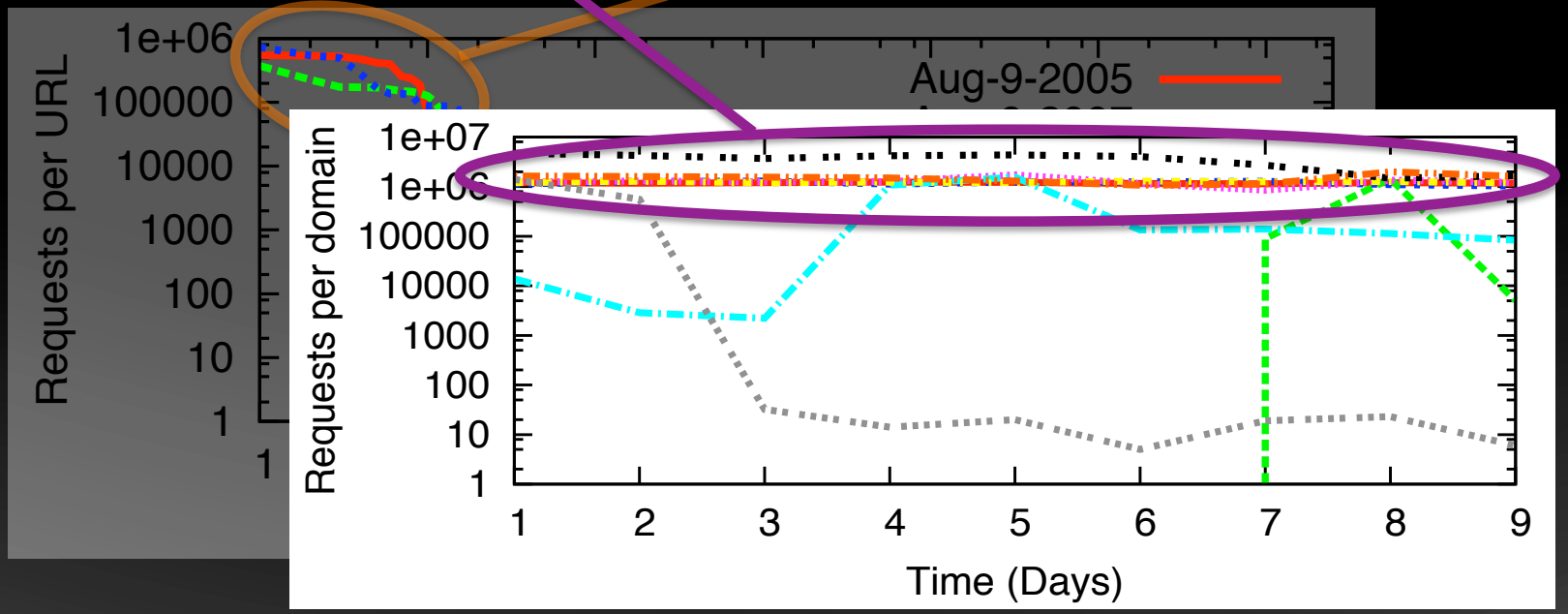
Usage Scenarios

1. Resurrecting old content
2. Accessing unpopular content
3. Serving long-term popular content



Usage Scenarios

1. Resurrecting old content
2. Accessing unpopular content
3. Serving long-term popular content



Usage Scenarios

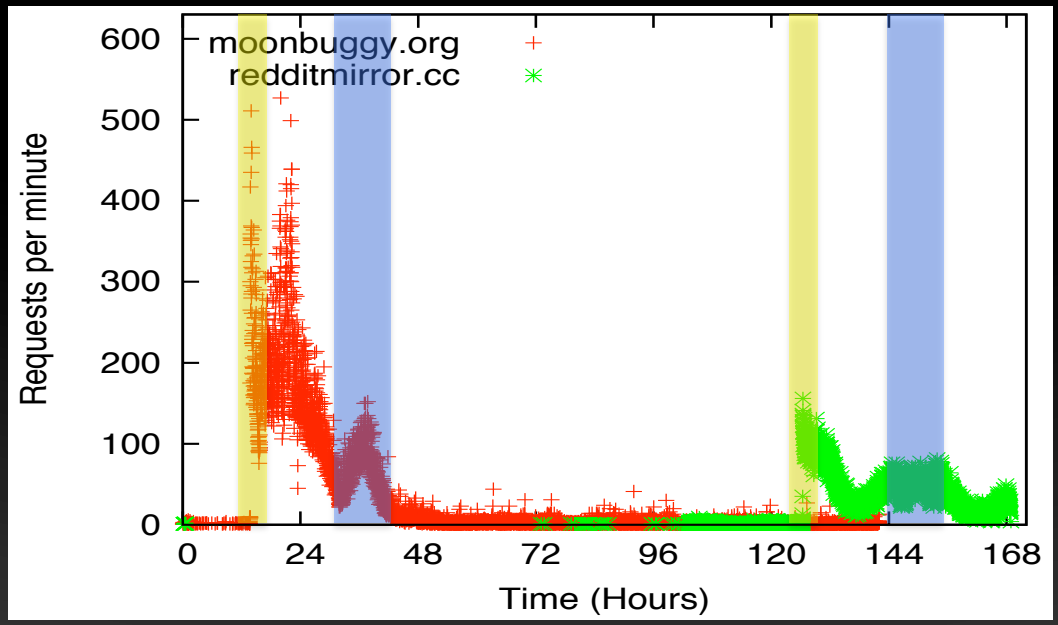
1. Resurrecting old content
2. Accessing unpopular content
3. Serving long-term popular content

Top URLs	% Reqs	Agg Size (MB)
0.01 %	49.1 %	14
0.10 %	71.8 %	157
1.00 %	84.8 %	3744
10.00 %	92.2 %	28734

Result	Frequency
Local Cache	70.4 %
Origin Site	9.9 %
CoralCDN Proxy	7.1 %
4xx/5xx Error	12.6 %

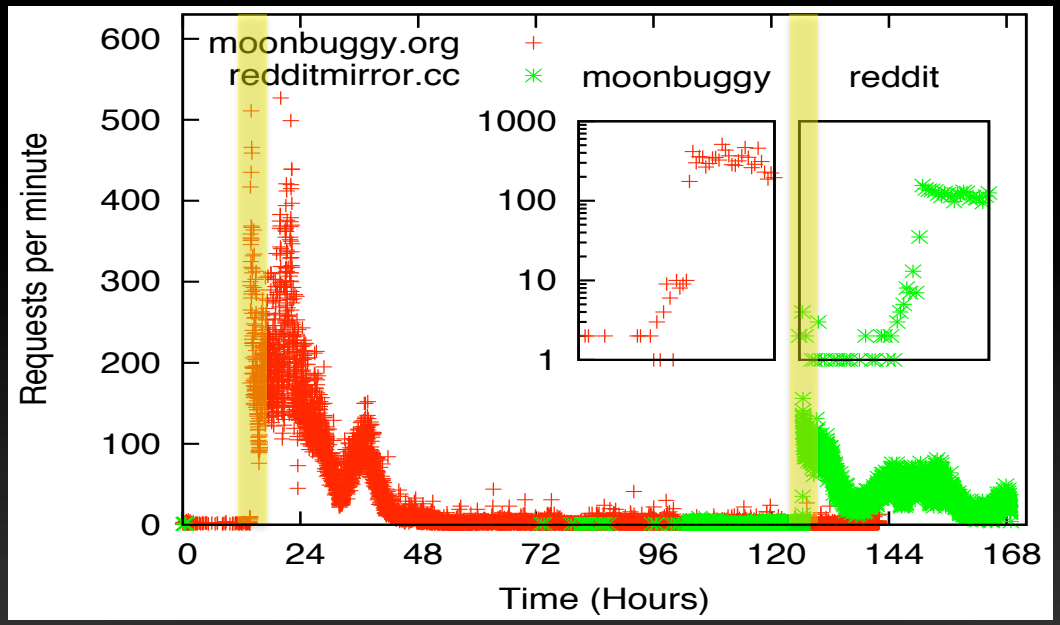
Usage Scenarios

1. Resurrecting old content
2. Accessing unpopular content
3. Serving long-term popular content
4. Surviving flash crowds to content



Usage Scenarios

1. Resurrecting old content
2. Accessing unpopular content
3. Serving long-term popular content
4. Surviving flash crowds to content



Usage Scenarios

1. Resurrecting old content
2. Accessing unpopular content
3. Serving long-term popular content
4. Surviving flash crowds to content

5 second epochs

24% epochs	≥ 1 domain with 10x incr
0.006% epochs	≥ 1 domain with 100x incr
0 % epochs	≥ 1 domain with 1000x incr

10 minute epochs

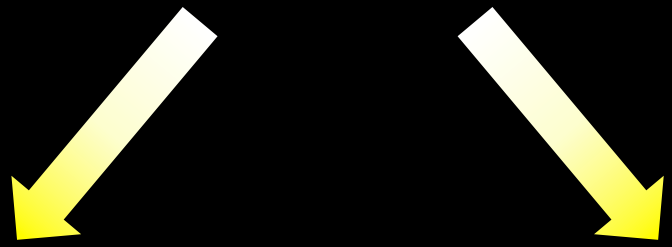
99.93% epochs	≥ 1 domain with 10x incr
28% epochs	≥ 1 domain with 100x incr
0.21% epochs	≥ 1 domain with 1000x incr

Conclusions?

- Most requested content is long-term popular and already cached locally
- “Flash” crowds occur, but on order of minutes

Conclusions?

- Most requested content is long-term popular and already cached locally
- "Flash" crowds occur, but on order of minutes

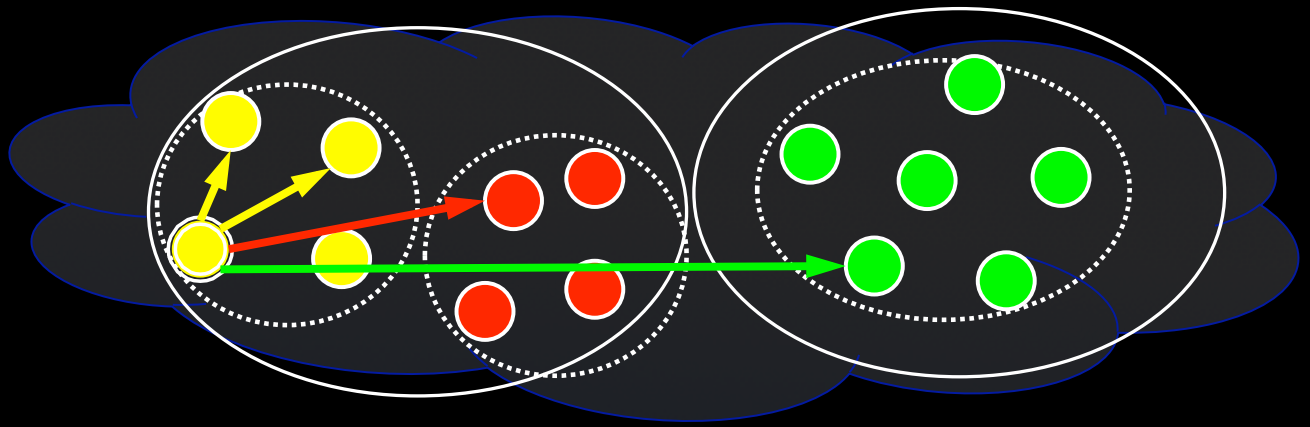


- Focus on long-term popular
- Little / no HTTP cooperation
- Global discovery (e.g., DNS)

- Focus on flash crowds
- Regional coop. as default
- Global coop. as failover

Reconfiguring CoralCDN's design

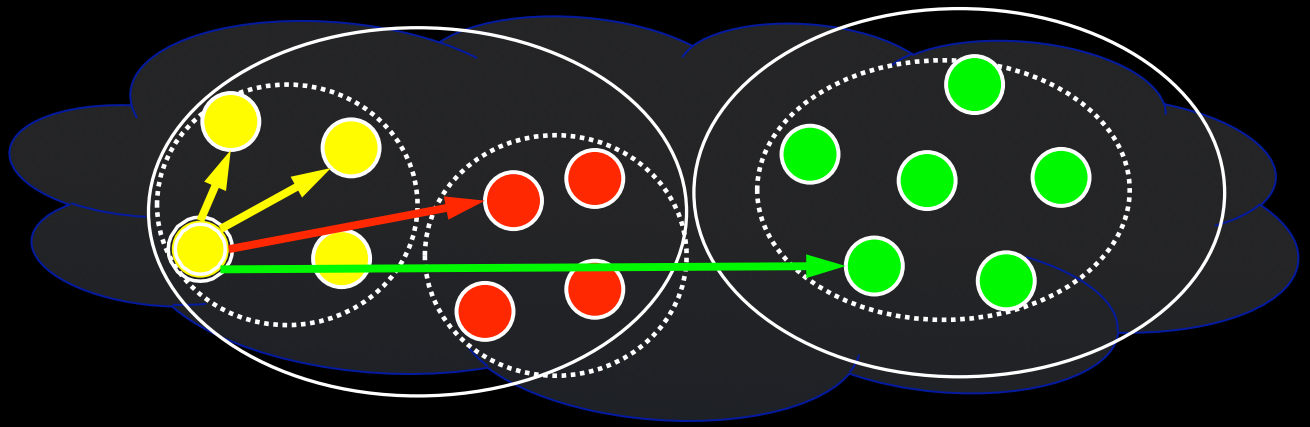
- Leverage Coral hierarchy for lookup



Latency 90% ↓ Origin Load 5% ↑ Failover to global 0.5%

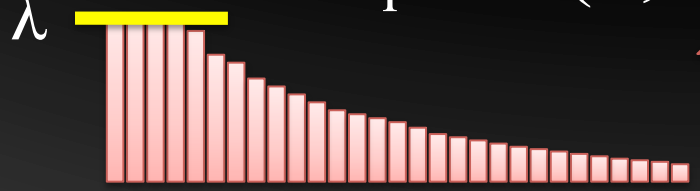
Reconfiguring CoralCDN's design

- Leverage Coral hierarchy for lookup



- During admission control, bias against long-term use

$$\sum_i \min(\lambda, d_i) < S$$



heavily weight history in ewma

Conclusions

1. Experiences
   - Naming
   - Fault Tolerance
   - Resource management
2. Revisit CoralCDN's design
   - Current design unnecessary for deployment / most use
   - Easy changes to promote flash-crowd mitigation

Can we reach Internet scale?

www.firecoral.net

Initial beta-release

of browser-based P2P web cache