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?
How bad can it get?

qmail mirror selection: choose one close to you!

USA
This site
Cerberus (Miami, Orlando, San Antonio)
Alaska
Arizona
California 1: CRL, Above
California 2: UUNET & BBN & AT&T
California 3
California 4: Level3 & XO
California 5: San Jose
Florida 1: UUNET
Florida 2: Level3
Georgia, Nivis
Illinois
Indiana, Sprint & AT&T
Kansas
Kentucky
Massachusetts
Michigan UUNET
Missouri 1: SBC & Sprint
Missouri 2: Sprint & C&W
New Jersey: Algo & XO
North Carolina
New York: Sprint
New Jersey

ASIA/OCEANIA
Australia 1
Australia 2
Australia 3
Australia 4
Australia 5
Australia 6
China 1
Hong Kong 1
Hong Kong 2
Hong Kong 3
Hong Kong 4
Hong Kong 5
Hong Kong 6
Indonesia 1
Indonesia 2
Indonesia 3(last updated Nov 2)
Indonesia 4
Indonesia 5
Iran
Japan 1
Japan 2
Japan 3
Japan 4

EUROPE
Austria 1
Austria 2
Belgium ([last updated Oct 23])
Belgium 2
Bosnia and Herzegovina
Bosnia and Herzegovina 2
Bulgaria 1
Bulgaria 2
Croatia
Czech Republic 1
Czech Republic 2
Denmark 1
Denmark 2
Denmark 3
Estonia
Finland 1
France 1
France 2([timestamp.html bad])
France 3
France 4
France 5
France 6

WORLD
Argentina 1
Argentina 2
Argentina 3
Argentina 4
Brazil 1
Brazil 2
Canada 1
Canada 2
Canada 3
Canada 4
Canada 5
Canada 6
Canada 7
Chile
Egypt([timestamp.html bad])
Israel
Mexico
South Africa
Venezuela
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

ICMP
Strawman: probe & find nearest

mycdn

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

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*?

mycdn

---

yourcdn

![CDN Diagram]

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

18.0.0.0/8
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

mycdn

Large set of replicas that assist in measurement

Reliable core of hosts that implement anycast

replica proxy

mycdn

DNS

RPC

OASIS node
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?

Using OASIS via HTTP
How does core answer anycast?

Bucketing

IP addr: 18.71.0.3

Name: mycdn.nyuld.net

Service

Policy

Replicas

18.26.4.9
171.66.3.181
216.165.109.81

Response

18.26.4.9

Proximity

IP prefix: 18.0.0.0/8
How to map IP prefix to coords?

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

- Location
- Accuracy
How to map IP prefix to coords?

- Two-pronged approach
  - Find closest replica proxy

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

location accuracy

(42N,71W)

18.0.0.0/8
How to map IP prefix to coords?

- Two-pronged approach
  - Find closest replica proxy
  - Use closest replica’s geo-coords + error RTT as location

18.0.0.0/8: (42N, 71W), 6.0 ms
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

18.0.0.0/8: (42N,71W), 6.0 ms
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

18.0.0.0/8: (42N,72W), 3.0 ms
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?

- IP addr
- name
- IP prefix

Local info from gossiping (stale data okay)

response

replicas

service

proximity

bucketing
How to find “nearby” nodes?

- IP addr: 18.26.4.9
- name: mycdn.nyuld.net

bucketing

- IP prefix: 18.0.0.0/8

service

proximity

replicas

Local info from gossiping (stale data okay)

Clients react poorly to stale data
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

- 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

290% faster than Meridian

500% faster than RR

Cached virtual coords highly inaccurate
Client RTT to chosen replica

Outperforms Meridian 60% of time
OASIS minimizes bandwidth spikes

95% bandwidth usage per replica (MB)

<table>
<thead>
<tr>
<th>Metric</th>
<th>CA</th>
<th>TX</th>
<th>NY</th>
<th>Germany</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency Only</td>
<td>23.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Load + Latency</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 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)

<table>
<thead>
<tr>
<th>Metric</th>
<th>CA</th>
<th>TX</th>
<th>NY</th>
<th>Germany</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency Only</td>
<td>23.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Load + Latency</td>
<td>9.0</td>
<td>11.3</td>
<td>9.6</td>
<td>9.2</td>
</tr>
</tbody>
</table>

- 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

1-2 orders of magnitude

# DNS reqs to CoralCDN
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
Netops have low tolerance for probing

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

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

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

- **dns**
- **nakika**
- **listen(7060)**
- **replica proxy**
- **OASIS core node**

**ServiceName**: nakika
**LocalPort**: 7060
**SecretCode**: 555555

**ServiceName**: nakika
**ServiceAlias**: nakika.net
**SortType**: latencycap
**MaxAddrs**: 2
**AddrTTLs**: 120
Make it easy to integrate

- ServiceName: nakika
- LocalPort: 7060
- SecretCode: 555555

Clients immediate use nakika.nyuld.net
Current services using OASIS…

- **Chunkcast** block anycast (Berkeley)
- **CoralCDN** (NYU)
- **Na Kika** content distribution (NYU)
- **OASIS**
  - RPC, DNS, HTTP interfaces
- **OCALA** 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/