**Introduction and Motivation**

*Example - Data retrieval on Web*

The Netscape source code was made available publicly on March 31 1998 via multiple Web sites.

Problems experienced:

- List of 111 servers - often no clear relationship between domain names and location
- Limited capacity of servers
- Limited reliability of servers

**Proposal: Distributed servers**

Motivation:

Enable very high-volume Internet services.

Server (cluster) selection

Failure recovery and dynamic adaption

No load-distribution and balancing across clusters
Illustration - cluster selection

Given a URL, transparently find best server (cluster)

Benefits

- Service latency improvements (through traffic localization)
- Bandwidth utilization improvements (through localization)
- Scalability (through replication)
- Higher availability (through replication)
- Transparency to user
- Automatic adjustment to changes in provider availability

Overview

Introduction and Motivation

Existing Solutions

A New Approach

Existing Solutions

- User choice
- DNS: shuffle addresses, aliasing, Service Resource Records (SRR), Geographical Positioning information (GPOS)
- IBM: Interactive Network Dispatcher (IND)
- Cisco: DistributedDirector (DD)
- IETF: Service Location Protocol (SLP)
- NTT: HOST Proximity Service (HOPS)
- HTTP: Redirection
- Berkeley: Smart Client Browser Architecture, Shared Passive Network performance Detection (SPAND)
- GA Tech: Application Level Anycasting Service (ALAS)
User choice

User obtains the URL; provides it to the browser

Multiple URLs

http://www1.sun.com/
http://www2.sun.com/
http://www.sun-usa.com/
http://www.sun-uk.com/

User Choice (cont.)

No load-balancing or graceful service degradation
Extra traffic, communication latency and server load
Requires user involvement; not transparent
Can distribute servers all over the network: traffic locality varies
Many providers use this approach today

DNS: shuffle addresses

IP addresses for a host name are given out in round robin fashion

- Addresses of unavailable servers are handed out
- No load balancing
- No optimization based on client location

DNS Service Resource Records

New resource record type for specifying the location of services: SRV RR


- Weight field for load balancing and port number for service location
- No fine grained load balancing information to be returned to the client
- No consideration of client location
- Multiple addresses can be returned. Client choice
**DNS SRV RR (cont.)**

- Quickly changing information eliminates DNS caching advantages
- Weight field useful for “This machine is three times as fast as that one”
- DNS spoofers can now supply false port numbers.

---

**IBM: Interactive Network Dispatcher (IND)**

Dynamic DNS provides load-balancing

Ping triangulation for geographical distribution - steps (2), (3) and (4)

---

**Cisco: Distributed Director (DD)**

Dynamic DNS provides load-balancing and locality improvement

DRP - BGP distance measures - steps (2), (3) and (4)
**DD (cont.)**

Director Response Protocol (DRP) for metrics

All Cisco routers can be DRP agents
- leverages Cisco’s market share

Topological distance
- external: BGP distance (DD - border router)
- internal: IGP distance (border router - agent)
- server: IGP distance (agent - server)

Additional metrics: server capacity, availability

**DD (cont.) - DNS mode**

DD becomes primary name server

Client sends DNS queries to DD

DD provides client IP address to the DR agents

DD uses DR responses to select the “best” server

DD provides server IP address to the client

Disable local DNS name caching

DR agents poll servers for availability

Extra latency, traffic for answering every DNS query

**HTTP redirection**

Browser talks HTTP to the server

Use HTTP redirection to direct clients to different servers

```
GET http://www.sunlabs.com
"HTTP-REDIRECT: 22.22.22.22"
11.11.11.11

"GET http://www.sunlabs.com"
"NO-CACHE: Welcome to Sun Labs"
22.22.22.22
```

**HTTP redirection (cont.)**

First connection always goes to the same initial host

Extra traffic, communication latency, and server load

Does not work with FTP and other services

Suitable for large connections (MBytes of data) - amortize the extra cost

Can distribute servers all over the network; good traffic locality feasible
HOst Proximity Service (HOPS)
Proposal by Paul Francis (NTT Software Labs)
Architecture for such a service with HOPS servers and HOPS probes
Probes determine nearness to address prefixes relative to itself via BGP, IGP, traceroute, measurements, etc.
Very early stage

Application Level Anycasting Service (ALAS)
Anycast domain names denoting an anycast group of IP addresses
Anycast resolvers to provide ADN to IP mapping
Protocol: anycast query and response
Metric databases maintain srv performance data
Combined server push and client probe technique for update of metric database.
- Not application transparent
- Additional probing network traffic

Overview
Introduction and Motivation
Existing Solutions
A New Approach

New Approach
Example service:
- Web service
Infrastructure support for delivery:
- Domain Name System (DNS) - for prototype
Infrastructure support for availability (load) metrics:
- Other components of this project
Platform for client participation:
- DNS resolver or local DNS server - for prototype
**Proposal: 1st step**

Servers create distance/metric tables (e.g., from BGP)

---

**Proposal: 2nd step**

Data and/or code returned by DNS

Client determines *best* server choice

---

**Thesis**

The problem of server (cluster) selection can be solved by the mechanism described above. This solution is superior to existing solutions with respect to characteristics, such as

- Latency (as observed by the client)
- Bandwidth utilization (global effect)
- Scalability
- Adjustment to server availability
- Caching of name to address bindings
- User transparency

---

**Research agenda**

- Prove thesis through prototype and exploration
- What data and/or code has to be distributed?
- What are *good* server choices?
- What does *good* mean, e.g., what are appropriate distance metrics?
- What are appropriate server selection algorithms?
- How can oscillation be prevented?
- Which delivery mechanism for data/code is appropriate?