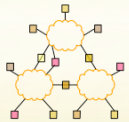
15-446 Distributed Systems Spring 2009



L-6 Naming

1

Today's Lecture

- Naming overview
- DNS
- Service location
- Server selection

2

Names

- Names are associated with objects
  - Enables passing of references to objects
  - Indirection
  - Deferring decision on meaning/binding
- Examples
  - Registers → R5
  - Memory → 0xdeadbeef
  - Host names → srini.com
  - User names → sseshan
  - Email → srini@cmu.edu
  - File name → /usr/srini/foo.txt
  - URLs → http://www.srini.com/index.html

3

Naming Model

- 3 key elements

1) Name space
   - Alphabet of symbols + syntax that specify names
2) Name-mapping
   - Associates each name to some value in…
3) Universe of values
   - Typically an object or another name from original name space (or another name space)

- Name-to-value mapping is called a "binding" i.e. name is bound to value

4

Naming Model (cont.)

- Uniqueness
  - One-to-one mapping
  - One-to-many or many-to-one (name-to-value) mappings
  - Context sensitive resolution
- Stable binding
  - Names that are never reused
  - Values that can only have one name
  - E.g. using MD5 of file contents, bank account numbers
- Reverse lookup support

5

Name Mapping

- Names are mapped to values within some context
  - E.g., different lookup tables for names in different settings
- Two sources for context
  - Resolver can supply default context
  - Name can specify an explicit context to use → qualified name
  - E.g. working directory vs. absolute path name

6

Context

- Common problem → what context to use for names without context
- Consider email from CMU
  - To: srini, dongsu@gmail.com
  - What happens when dongsu replies to all?
    - What context will he email srini
  - Solutions:
    - Sendmail converts all address to qualified names
      - Not in body of message
    - Provide context information in email header
      - E.g. like base element in HTML

7

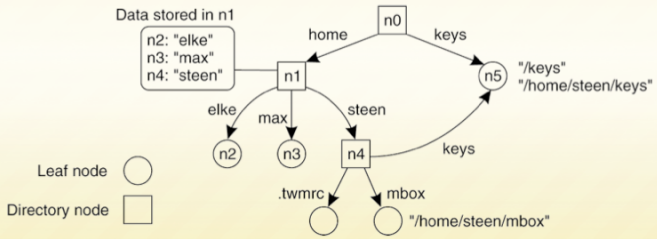
Name Lookup Styles

- Table lookup
  - Simple, table per context
- Recursive
  - Names consist of context + name
  - E.g. path + filename, hostname + domain name
  - Context name must also be resolved
    - Need special context such as “root” built into resolver
- Multiple lookup
  - Try multiple contexts to resolve name → search paths

8

Recursive Name Spaces

- A general naming graph with a single root node.



9

Name Discovery

- Well-known name
  - www.google.com, port 80…
- Broadcast
  - Advertise name → e.g. 802.11 Beacons
- Query
  - Use google
- Broadcast query
  - 802.11 probes
- Use another naming system
  - DNS returns IP addresses
- Introductions
  - Web page hyperlinks
- Physical rendezvous
  - Exchange info in the real world

10

Today's Lecture

- Naming overview
- DNS
- Service location
- Server selection

11

Naming

- How do we efficiently locate resources?
  - DNS: name → IP address
- Challenge
  - How do we scale these to the wide area?

12

Obvious Solutions (1)

Why not centralize DNS?

- Single point of failure
- Traffic volume
- Distant centralized database
- Single point of update

- Doesn't *scale!*

13

Obvious Solutions (2)

Why not use /etc/hosts?

- Original Name to Address Mapping
  - Flat namespace
  - /etc/hosts
  - SRI kept main copy
  - Downloaded regularly
- Count of hosts was increasing: machine per domain → machine per user
  - Many more downloads
  - Many more updates

14

Domain Name System Goals

- Basically a wide-area distributed database
- Scalability
- Decentralized maintenance
- Robustness
- Global scope
  - Names mean the same thing everywhere
- Don't need
  - Atomicity
  - Strong consistency

15

Typical Resolution

- Steps for resolving www.cmu.edu
  - Application calls gethostbyname() (RESOLVER)
  - Resolver contacts local name server (S_1)
  - S_1 queries root server (S_2) for (www.cmu.edu)
  - S_2 returns NS record for cmu.edu (S_3)
  - What about A record for S_3?
    - This is what the additional information section is for (PREFETCHING)
  - S_1 queries S_3 for www.cmu.edu
  - S_3 returns A record for www.cmu.edu

- Can return multiple A records → what does this mean?

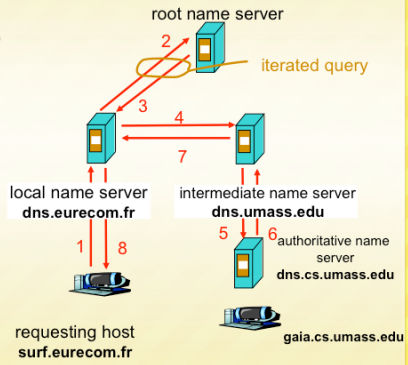
16

Lookup Methods

Recursive query:
- Server goes out and searches for more info (recursive)
- Only returns final answer or "not found"

Iterative query:
- Server responds with as much as it knows (iterative)
- "I don't know this name, but ask this server"

Workload impact on choice?
- Local server typically does recursive
- Root/distant server does iterative



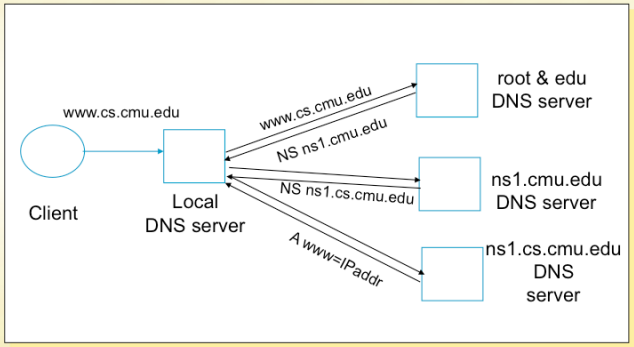
17

Workload and Caching

- Are all servers/names likely to be equally popular?
  - Why might this be a problem? How can we solve this problem?
- DNS responses are cached
  - Quick response for repeated translations
  - Other queries may reuse some parts of lookup
    - NS records for domains
- DNS negative queries are cached
  - Don't have to repeat past mistakes
  - E.g. misspellings, search strings in resolv.conf
- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record

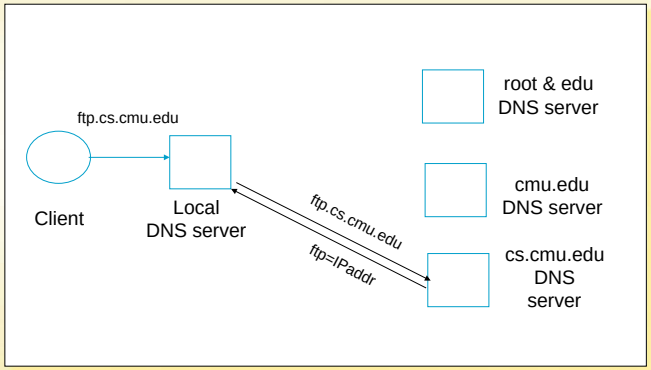
18

Typical Resolution



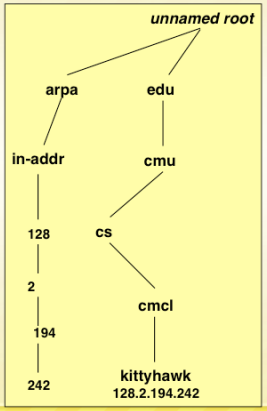
19

Subsequent Lookup Example



20

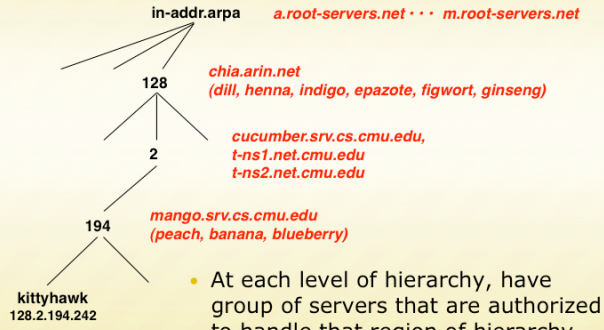
Reverse DNS



- Task
  - Given IP address, find its name
- Method
  - Maintain separate hierarchy based on IP names
  - Write 128.2.194.242 as 242.194.128.2.in-addr.arpa
    - Why is the address reversed?
- Managing
  - Authority manages IP addresses assigned to it
  - E.g., CMU manages name space 128.2.in-addr.arpa

21

.arpa Name Server Hierarchy



- At each level of hierarchy, have group of servers that are authorized to handle that region of hierarchy

22

Prefetching

- Name servers can add additional data to response
- Typically used for prefetching
  - CNAME/MX/NS typically point to another host name
  - Responses include address of host referred to in "additional section"

23

Mail Addresses

- MX records point to mail exchanger for a name
  - E.g. mail.acm.org is MX for acm.org
- Addition of MX record type proved to be a challenge
  - How to get mail programs to lookup MX record for mail delivery?
  - Needed critical mass of such mailers

24

DNS (Summary)

- Motivations → large distributed database
  - Scalability
  - Independent update
  - Robustness
- Hierarchical database structure
  - Zones
  - How is a lookup done
- Caching/prefetching and TTLs
- Reverse name lookup
- What are the steps to creating your own domain?

25

Today's Lecture

- Naming overview
- DNS
- Service location
- Server selection

26

Service Location

- What if you want to lookup services with more expressive descriptions than DNS names
  - E.g. please find me printers in cs.cmu.edu instead of laserjet1.cs.cmu.edu
- What do descriptions look like?
- How is the searching done?
- How will it be used?
  - Search for particular service?
  - Browse available services?
  - Composing multiple services into new service?

Service Descriptions

- Typically done as hierarchical value-attribute pairs
  - Type = printer → memory = 32MB, lang = PCL
  - Location = CMU → building = WeH
- Hierarchy based on attributes or attributes-values?
  - E.g. Country → state or country=USA → state=PA and country=Canada → province=BC?
- Can be done in something like XML

Service Discovery (Multicast)

- Services listen on well known discovery group address
- Client multicasts query to discovery group
- Services unicast replies to client
- Tradeoffs
  - Not very scalable → effectively broadcast search
  - Requires no dedicated infrastructure or bootstrap
  - Easily adapts to availability/changes
  - Can scope request by multicast scoping and by information in request

Service Discovery (Directory Based)

- Services register with central directory agent
  - Soft state → registrations must be refreshed or the expire
- Clients send query to central directory → replies with list of matches
- Tradeoffs
  - How do you find the central directory service?
    - Typically using multicast based discovery!
    - SLP also allows directory to do periodic advertisements
  - Need dedicated infrastructure
  - How do directory agents interact with each other?
  - Well suited for browsing and composition → knows full list of services

Other Issues

- Dynamic attributes
  - Many queries may be based on attributes such as load, queue length
  - E.g., print to the printer with shortest queue
  - Bind to value as late as possible
- Security
  - Don't want others to serve/change queries
  - Also, don't want others to know about existance of services
    - Srini's home SLP server is advertising the $50,000 MP3 stereo system (come steal me!)

Today's Lecture

- Naming overview
- DNS
- Service location
- Server selection

Server Selection

- Service is replicated in many places in network
- How do direct clients to a particular server?
  - As part of routing → anycast, cluster load balancing
  - As part of application → HTTP redirect
  - As part of naming → DNS
- Which server?
  - Lowest load → to balance load on servers
  - Best performance → to improve client performance
    - Based on Geography? RTT? Throughput? Load?
  - Any alive node → to provide fault tolerance

Routing Based

- Anycast
  - Give service a single IP address
  - Each node implementing service advertises route to address
  - Packets get routed routed from client to "closest" service node
    - Closest is defined by routing metrics
    - May not mirror performance/application needs
  - What about the stability of routes?

Routing Based

- Cluster load balancing
  - Router in front of cluster of nodes directs packets to server
  - Must be done on connection by connection basis – why?
    - Forces router to keep per connection state
  - How to choose server
    - Easiest to decide based on arrival of first packet in exchange
    - Primarily based on local load
    - Can be based on later packets (e.g. HTTP Get request) but makes system more complex

Application Based

- HTTP support simple way to indicate that Web page has moved
- Server gets Get request from client
  - Decides which server is best suited for particular client and object
  - Returns HTTP redirect to that server
- Can make informed application specific decision
- May introduce additional overhead → multiple connection setup, name lookups, etc.
- While good solution in general HTTP Redirect has some design flaws – especially with current browsers

Naming Based

- Client does name lookup for service
- Name server chooses appropriate server address
- What information can it base decision on?
  - Server load/location → must be collected
  - Name service client
    - Typically the local name server for client
- Round-robin
  - Randomly choose replica
  - Avoid hot-spots
- [Semi-]static metrics
  - Geography
  - Route metrics
  - How well would these work?

Naming Based

- Predicted application performance
  - How to predict?
  - Only have limited info at name resolution
- Multiple techniques
  - Static metrics to get coarse grain answer
  - Current performance among smaller group
- How does this affect caching?
  - Typically want low TTL to adapt to load changes
  - What does the first and subsequent lookup do?

Summary

- Naming is a powerful tool in system design
  - A layer of indirection can solve many problems
- Wide range of naming styles, resolution techniques
  - Must choose the one appropriate to system needs/ tradeoffs

Next Lecture

- RPC
- Read original Birrell & Nelson paper on RPC