CS 555: DISTRIBUTED SYSTEMS [NAMING]

In distributed systems the implementation of a name space spans multiple domains

- Usually organized hierarchically
- Partitioned into layers
  - Global
  - Administrative
  - Managerial

Topics covered in this lecture

- Implementation of a name space
- Resolution in name spaces
- Examples
  - DNS
  - LDAP

Frequently asked questions from the previous class survey

- Hierarchical organization of topics: DFS or BFS?
- How can this scale if the root holds a lot of info?

Instructor: SHRIDEEP PALLICKARA
Computer Science
Colorado State University

October 29, 2015

GLOBAL LAYERS

- This is formed by the highest level nodes
  - Root node and other nodes logically close to the root
  - Represents organizations or a group of organizations
  - Directory tables rarely change

CS55: Distributed Systems [Fall 2015]
Dept. Of  Computer  Science, Colorado  State  University
**Administrational Layer**

- Formed by directory nodes that are managed within a single organization
- Constituent directory nodes may represent an organization
- Directory node changes occur more frequently than that in the global layer

**Managerial Layer**

- Comprises nodes that change regularly
- Includes nodes representing
  - Shared files
  - User-defined directories and files
- Maintained by system administrators & users

---

**Example partitioning of the DNS name space**

- **Global Layer**
  - Performance and availability
    - Server replication and client caching
    - Responses do not have to be particularly fast
    - Throughput is far more important

- **Administrational Layer**
  - High availability is critical
    - Failures result in a large portion of the name space becoming unavailable

- **Managerial Layer**
  - Lookup results must be returned in a few milliseconds
  - Updates must be processed faster
    - User account cannot take hours to become effective

---

**Performance & availability requirements: Global Layer**

- High availability is critical
  - Failures result in a large portion of the name space becoming unavailable
  - Performance
    - Flux is low
    - Results of lookups are valid for long durations
    - Results can be cached

---

**Global layer: Performance and availability**

- Server replication and client caching
- Responses do not have to be particularly fast
- Throughput is far more important

---

**Administrational Layer: Performance and availability**

- Caching is still possible
- Lookup results must be returned in a few milliseconds
- Updates must be processed faster
  - User account cannot take hours to become effective
Administrational Layer:
Performance and availability
- No need to replicate
- Can risk temporary unavailability
- Performance requirements are stringent
- Operations must take place immediately

Contrasting the layers

<table>
<thead>
<tr>
<th></th>
<th>Global</th>
<th>Administrative</th>
<th>Managerial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geographical scale of network</td>
<td>Worldwide</td>
<td>Organization</td>
<td>Department</td>
</tr>
<tr>
<td>Total number of nodes</td>
<td>Few</td>
<td>Many</td>
<td>Vast numbers</td>
</tr>
<tr>
<td>Responsiveness to lookups</td>
<td>Seconds</td>
<td>Milliseconds (10s)</td>
<td>Milliseconds (few)</td>
</tr>
<tr>
<td>Update propagation</td>
<td>Lazy</td>
<td>Immediate</td>
<td>Immediate</td>
</tr>
<tr>
<td>Number of replicas</td>
<td>Many</td>
<td>None or few</td>
<td>None</td>
</tr>
<tr>
<td>Client-side caching</td>
<td>Yes</td>
<td>Yes</td>
<td>Sometimes</td>
</tr>
</tbody>
</table>

Resolution in distributed name spaces
- Distribution of namespace across multiple name servers affects name resolution
- Each client has access to local name resolver
- Approaches:
  - Iterative
  - Recursive

Namespace resolution

Iterative resolution
- Client hands complete name to the root name server
- Root resolves as far as it can, and returns result
- Client then passes remaining path to the returned name server

The iterative lookup process for
root:<edu, colostate, cs, ftp, pub, globe, index.html>
Recursive name resolution

- Do not return intermediate results to client
- Name server passes results to the next name server
- Last resolution step carried out as a separate process by client
  - E.g. contacting FTP server to request transfer of the file

The recursive lookup process for:

```
root:<edu, colostate, cs, ftp, pub, globe, index.html>
```

1. `edu.colostate.cs.ftp`
2. `colostate.cs.ftp`
3. `cs.ftp`
4. `<ftp>`
5. `<ftp>`
6. `ftp`
7. `cs, ftp`
8. `<edu, colostate, cs, ftp>`


Performance implications of recursive lookups

- Name server required to handle complete name resolution
- Higher performance demands on each name server
- Name servers in the global layer only support iterative lookups

Advantages of recursive name resolution

- Caching results is more effective
- Communication costs are reduced

How caching is more effective in recursive lookups

- Each name server gradually learns addresses of lower-level nodes
- Addresses are also returned by recursion
  - So caching is possible at intermediate nodes too

Caching intermediate results
The Domain Name System

- One of the largest distributed naming services in use
- Primarily used for looking up IP addresses of hosts and mail servers
- Designers had a deep understanding of keeping things simple

Domain Name System

- Follows a hierarchy of 13 well-known root servers
- Ends in millions of servers at the leaves

The DNS namespace

- Hierarchically organized as a rooted tree
- Labels
  - Case-insensitive, maximum length = 63
  - Representation of path name
    - Separate labels with a . (dot)
    - Maximum path name length = (255 characters)

The DNS namespace

- Each node has exactly one incoming edge
  - Except root, which has none
- Label of incoming edge is the name of the node
- Sub-tree
  - Domain
- Path name to the root node?
  - Domain name

Contents of a node in DNS

- Collection of resource records
- Several types of records
  - 9 major ones
Resource Records: SOA and A

- SOA (Start of Authority)
  - E-mail address of administrator for a zone
  - Host name from where info on zone can be fetched
- A (address) record
  - Represents a host on the Internet
  - Contains IP address of host
  - If the host is multi-homed?
    - Multiple A records: one for each address

Resource Records: MX (Mail Exchange)

- Node representing the mail server
  - Like a symbolic link to a node representing mail server
- Server will handle all incoming mail addressed to cs.colostate.edu
  - For e.g. for cs.colostate.edu
    - mail.cs.colostate.edu

Resource Records: SRV (service)

- Name of a server for a specific service
  - Name + Name of protocol
- Web server in the cs.colostate.edu domain
  - SRV record _http._tcp.cs.colostate.edu
  - SRV record would refer to actual name of the server
    - parsons.cs.colostate.edu

Resource Records: NS (Name Server)

- Contains name of the NAME SERVER that implements the zone
- Only nodes representing zones in DNS need to store NS records

Resource Records: CNAME (Canonical Name)

- Every host has a canonical or primary name
- DNS distinguishes aliases from canonical names
- To implement alias
  - Node stores CNAME record with canonical name of host

Resource Records: PTR (Pointer)

- Used to maintain inverse mapping of IP addresses to host names
- DNS maintains a domain in-addr.arpa to support inverse lookups
- IP of www.cs.colostate.edu is 129.82.45.114
  - DNS creates a node 114.45.82.129.in-addr.arpa
  - Stores canonical name parsons.cs.colostate.edu in a PTR record
Decentralized DNS

- We now understand how a huge collection of flat names can be easily supported
- Compute hash of a DNS name
- Take hash as a key value to look up in DHT
- Disadvantages:
  - Difficult to find children of a specific domain

Quick summary of flat/structured naming

- Unique and location-independent names
  - Flat and structured
- Human friendly?
  - Structured

Do we need something beyond uniqueness and human friendliness?

- How about we provide only a description of what we are looking for?
- Popular approach is to use <attribute, value> pairs
  - Attribute-based naming

The elements of attribute naming

- An entity has a collection of attributes
- Each attribute says something about the entity
- Constrain the set of entities you look for by
  - Specifying values that an attribute can take

Attribute-based naming systems

- Are called directory services
- Allow attribute-based search
But different things end up getting described differently

- Can we make sense of all this?
- **Unifying** the ways in which resources can be described
  - Resource description format (RDF)

### Description of resources in RDF

- Done using **triplets** \{subject, predicate, object\}
  - **Predicate**: Textual description
  - **Subjects, Objects**: Anything including references (URL)
  - \{PERSON, NAME, ALICE\}
    - Describes a resource PERSON whose name is ALICE
    - When you query for info associated with ALICE
      - Returns reference to the PERSON resource

### Performance issues in attribute-based naming systems

- Lookup requires an **exhaustive** search through all descriptors
- In a distributed environment?
  - Combine structured naming with attribute-based naming

### Lightweight directory access protocol (LDAP): Origins

- Telecom firms were the pioneers in directory services
  - Well-developed understanding of needs
  - 100 years of experience
- In 1980, the International Telecommunications Union produced the X.500 protocol suite

### LDAP [LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL]

- The directory access protocol (DAP) was used to access X.500 directory services
  - Required the OSI protocol stack (ISO/ITU)
- LDAP was intended for **lightweight** access using TCP/IP
LDAP directory services

- Contains a number of records: entries
- Each entry is a collection of (attribute, value) pairs
  - There is a type for each attribute
  - Attributes can be multi-valued (arrays/lists)

An example directory entry

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Abbreviation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country</td>
<td>C</td>
<td>US</td>
</tr>
<tr>
<td>Locality</td>
<td>L</td>
<td>Fort Collins</td>
</tr>
<tr>
<td>Organization</td>
<td>O</td>
<td>Colorado State University</td>
</tr>
<tr>
<td>OrganizationalUnit</td>
<td>OU</td>
<td>Computer Science</td>
</tr>
<tr>
<td>CommonName</td>
<td>CN</td>
<td>Main server</td>
</tr>
<tr>
<td>Mail_Servers</td>
<td></td>
<td>129.82.45.30, 129.82.45.3x</td>
</tr>
<tr>
<td>FTP_Server</td>
<td></td>
<td>129.82.45.33</td>
</tr>
<tr>
<td>WWW_Server</td>
<td></td>
<td>129.82.45.114</td>
</tr>
</tbody>
</table>

Managing entries/records in a LDAP directory

- Each record must be uniquely named
- Each naming attribute is called a relative distinguished name (RDN)
- Globally unique name
  - Sequence of naming attributes in each record
  - C=US/L=Fort Collins/O=CSU/OU=CS/CN=MS

Listing RDNs in a sequence leads to a hierarchy

- C=US
- O=Colorado State University
- OU=Computer Science
- Host_Name=broccoli, Host_Name=cabbage

Let's look at 2 more entries

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country</td>
<td>US</td>
<td>Country</td>
<td>US</td>
</tr>
<tr>
<td>Locality</td>
<td>Fort Collins</td>
<td>Locality</td>
<td>Fort Collins</td>
</tr>
<tr>
<td>Organization</td>
<td>Colorado State University</td>
<td>Organization</td>
<td>Colorado State University</td>
</tr>
<tr>
<td>OrganizationalUnit</td>
<td>Computer Science</td>
<td>OrganizationalUnit</td>
<td>Computer Science</td>
</tr>
<tr>
<td>CommonName</td>
<td>Main server</td>
<td>CommonName</td>
<td>Main server</td>
</tr>
<tr>
<td>Host_Name</td>
<td>broccoli</td>
<td>Host_Name</td>
<td>cabbage</td>
</tr>
<tr>
<td>Host_Address</td>
<td>129.82.47.205</td>
<td>Host_Address</td>
<td>129.82.47.207</td>
</tr>
</tbody>
</table>

LDAP supports two different lookup operations

- **read**
  - Retrieve a single record given its path name
- **list**
  - Lists the names of outgoing edges for a given node
The LDAP operations

- Client initiates an LDAP session
  - LDAP server runs on TCP port 389 (default)
- Client does not wait for responses to previous requests before sending any request
  - Server may send responses in any order

LDAP operations

- Bind – authenticate
- Search
- Compare
- Add/delete/modify an entry
- Modify distinguished name
- Abandon – abort a previous request
- Unbind

Searching within a directory service is expensive

- Find all main servers in CSU
  - Search all entries at each department
  - Combine results and respond
- We need to access several leaf nodes to get an answer

In a distributed setting, we could let multiple trees to co-exist

- Microsoft’s Active Directory does this
  - They call it a forest of LDAP domains
- Maintain a global index server
  - Tells you which LDAP domains need to be searched

Other directory services

- Universal directory and discovery integration (UDDI)
  - Web services and Grid computing
- Scalability has the same underlying principle
  - Distributed databases
  - Query relevant databases
  - Aggregate results before responding

The contents of this slide-set are based on the following references