Naming

Yao Liu
Outline

• Names and Addresses
• Flat Naming
• Structured Naming
• Attribute-based Naming
Naming entities

- A name in a distributed system is a string of bits or characters that is used to refer to an entity

Types of names

- Address: an access point of an entity
- Identifiers: a name that uniquely identifies an entity
  - An identifier refers to at most one entity
  - Each entity is referred to by at most one identifier
  - An identifier always refers to the same entity

Human-friendly names

- Location-independent name: a name that is independent from its addresses
Name resolution

• A naming system maintains a name-to-address binding for resources
• Main problem in naming – how to do name resolution in a distributed system in a scalable way
• Approaches for name resolution are closely tied to naming scheme
• Flat vs. structured naming
  • In a flat name space, identifiers are random bit strings
  • No information on location embedded in name
Naming vs. locating entities

(a) Direct, single level mapping between names and addresses.
(b) Two-level mapping using identities.
Locating mobile entities

- Consider an entity that changes its location
  - E.g., ftp.cs.binghamton.edu moves to another domain
    - Cannot change name
  - Two choices
    - Record the address of the new machine in the DNS database for cs.binghamton.edu
      - If name changes again, DNS entry will have to be changed again
    - Record the name of the new machine in the database, i.e., use a symbolic link
      - Inefficient lookups

- Traditional naming services such as DNS cannot cope well with mobile entities
  - Problems arise because of the direct mapping between human-friendly names and the address of entities
Outline

• Names and Addresses
• Flat Naming
• Structured Naming
• Attribute-based Naming
Name resolution in flat name spaces

• Simple solutions that work in a LAN environment
  • Broadcasting & Multicasting
    • Broadcast a message containing identifier of the entity; machine with an access point for the entity replies with the address of the access point
      • ARP protocol for finding the data-link address of a machine given the IP address
    • When network grows, multicast is more efficient.
  • Forwarding pointers
    • When an entity moves from A to B, it leaves behind a reference to its new location at B

• Home-based Approaches
  • Home agent keeps track of current location of mobile entity

• Distributed Hash Table
  • structured P2P

• Hierarchical Approaches
  • Similar to DNS (but different)
Forwarding pointers (1)

Problem: chain length, and broken chains
Redirecting a forwarding pointer, by storing a shortcut in a proxy.
Home-based approaches

1. Send packet to host at its home
2. Return address of current location
3. Tunnel packet to current location
4. Send successive packets to current location
What is a DHT?

- Single-node hash table:
  - key = Hash(name)
  - put(key, value)
  - get(key) -> value
- Service: \( O(1) \) storage

- How do I do this across millions of hosts on the Internet?
  - Distributed Hash Table
What is a DHT?

• Distributed Hash Table:
  key = Hash(data)
  lookup(key) -> IP address (Chord)
  send-RPC(IP address, PUT, key, value)
  send-RPC(IP address, GET, key) -> value

• Possibly a first step towards truly large-scale distributed systems
  • a tuple in a global database engine
  • a data block in a global file system
  • rare.mp3 in a P2P file-sharing system
The lookup problem

At the heart of all DHTs
Motivation: routed DHT queries (Tapestry, Pastry, Chord, CAN)

Publisher

Key=$H$(audio data)
Value={$artist$, $album$ title, $track$ title}

Lookup($H$(audio data))

Client

N4 → N1 → N2 → N3
Distributed Hash Table

• Chord:
  • Map node ID to a large circular space
  • Map Keys (Hash(Data)) to the same circular space
  • Key $k$ belong to the first node whose identifier is equal to or follows $k$ in the identifier space (successor($k$))
Lookup in Chord

- Every node need to be aware of the next node on the ring
- May traverse all $N$ nodes to find the Key
- $O(N)$ steps
Lookup in Chord: finger table

- The \( i^{th} \) entry in the table at node \( n \) contains the identity of the first node, \( s \), that succeeds \( n \) by at least \( 2^{i-1} \) on the ID circle.
- \( \mathcal{O}(\log N) \) steps

<table>
<thead>
<tr>
<th>Start</th>
<th>Interval</th>
<th>Succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[2, 3)</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>[3, 5)</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>[5, 9)</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>[9, 1)</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Start</th>
<th>Interval</th>
<th>Succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>[8, 9)</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>[9, 10)</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>[11, 15)</td>
<td>12</td>
</tr>
<tr>
<td>15</td>
<td>[15, 7)</td>
<td>15</td>
</tr>
</tbody>
</table>
Lookup in Chord: finger table

Resolving key 26 from node 1 and key 12 from node 28 in a Chord system.
Hierarchical approaches (1)

Hierarchical organization of a location service into domains, each having an associated directory node.
Hierarchical approaches (2)

An example of storing information of an entity having two addresses in different leaf domains.
Hierarchical approaches (3)

Looking up a location in a hierarchically organized location service.
Hierarchical approaches (4)

(a) An insert request is forwarded to the first node that knows about entity $E$.
(b) A chain of forwarding pointers to the leaf node is created.
Outline

• Names and Addresses
• Flat Naming
• Structured Naming
• Attribute-based Naming
Structured naming systems

- Names are organized into name spaces
- A name space can be represented as a labeled, directed graph with two types of nodes
  - Leaf nodes and directory nodes
  - Absolute vs. relative path names
  - Local names vs. global names
- Name Resolution: the process of looking up a name
  - Closure mechanism: knowing where and how to start name resolution
The general organization of the UNIX file system implementation on a logical disk of contiguous disk blocks.
Name spaces

A general naming graph with a single root node.
Linking and mounting (1)

The concept of a symbolic link explained in a naming graph.

Data stored in n1
n2: "elke"
n3: "max"
n4: "steen"

Data stored in n6
n6: "/home/steen/keys"

Directory node
Leaf node
Linking and mounting (2)

Information required to mount a foreign name space in a distributed system

• The name of an access protocol.
• The name of the server.
• The name of the mounting point in the foreign name space.
Mounting remote name spaces through a specific access protocol.
Implementing name spaces

- Naming service: a service that allows users and processes to add, remove, and lookup names
- Name spaces for large-scale widely distributed systems are typically organized hierarchically
- Three layers used to implement such distributed name spaces
  - Global layer: root node and its children
  - Administrative layer: directory nodes within a single organization
  - Managerial layer
An example partitioning of the DNS name space, including Internet-accessible files, into three layers.
A comparison between name servers for implementing nodes from a large-scale name space partitioned into a global layer, as an administrational layer, and a managerial layer.
Implementation of name resolution (1)

• Iterative vs. recursive name resolution
• Recursive name resolution puts a higher performance demand on each name server
  • Too high for global layer name servers
• Advantages of recursive name resolution
  • Caching is more effective
  • Communication costs may be reduced
Implementation of name resolution (2)

• The principle of iterative name resolution.
Implementation of name resolution (3)

• The principle of recursive name resolution.

1. <nl,vu,cs,ftp>
2. <vu,cs,ftp>
3. <cs,ftp>
4. <ftp>
5. #<ftp>
6. #<cs,ftp>
7. #<vu,cs,ftp>
8. #<nl,vu,cs,ftp>

Client's name resolver

Root name server

Name server nl node

Name server vu node

Name server cs node
### Implementation of name resolution (4)

<table>
<thead>
<tr>
<th>Server for node</th>
<th>Should resolve</th>
<th>Looks up</th>
<th>Passes to child</th>
<th>Receives and caches</th>
<th>Returns to requester</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs</td>
<td>&lt;ftp&gt;</td>
<td>#&lt;ftp&gt;</td>
<td>--</td>
<td>--</td>
<td>#&lt;ftp&gt;</td>
</tr>
<tr>
<td>vu</td>
<td>&lt;cs,ftp&gt;</td>
<td>#&lt;cs&gt;</td>
<td>&lt;ftp&gt;</td>
<td>#&lt;ftp&gt;</td>
<td>#&lt;cs&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#&lt;cs, ftp&gt;</td>
</tr>
<tr>
<td>ni</td>
<td>&lt;vu,cs,ftp&gt;</td>
<td>#&lt;vu&gt;</td>
<td>&lt;cs,ftp&gt;</td>
<td>#&lt;cs&gt;</td>
<td>#&lt;vu&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#&lt;vu,cs&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#&lt;vu,cs,ftp&gt;</td>
</tr>
<tr>
<td>root</td>
<td>&lt;ni,vu,cs,ftp&gt;</td>
<td>#&lt;nl&gt;</td>
<td>&lt;vu,cs,ftp&gt;</td>
<td>#&lt;vu&gt;</td>
<td>#&lt;nl&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#&lt;nl,vu&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#&lt;nl,vu,cs&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#&lt;nl,vu,cs,ftp&gt;</td>
</tr>
</tbody>
</table>

Recursive name resolution of <\textit{nl}, vu, cs, ftp>. Name servers cache intermediate results for subsequent lookups.
Implementation of name resolution (5)

- The comparison between recursive and iterative name resolution with respect to communication costs.

Diagram:
- Recursive name resolution
  - Client -> Name server nl node (R1)
  - Name server nl node -> Name server vu node (I1)
  - Name server vu node -> Name server cs node (I2)
  - Name server cs node -> Client (I3)

- Iterative name resolution
  - Client -> Name server nl node (R2)
  - Name server nl node -> Name server vu node (I2)
  - Name server vu node -> Name server cs node (I3)
  - Name server cs node -> Client (R3)

Long-distance communication
Example system: DNS

- Domain Name System (DNS)
  - Host name to IP address translation
  - Name space organized as a hierarchical rooted tree
    - Name space divided into non-overlapping zones
  - Name servers implement the global and administrational layers
    - Managerial layer is not part of DNS
    - Each zone has a name server, which is typically replicated
    - Updates take place at the primary name server for a zone
      - Secondary name servers request the primary name server to transfer its content
The DNS name space

<table>
<thead>
<tr>
<th>Type of record</th>
<th>Associated entity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOA</td>
<td>Zone</td>
<td>Holds information on the represented zone</td>
</tr>
<tr>
<td>A</td>
<td>Host</td>
<td>Contains an IP address of the host this node represents</td>
</tr>
<tr>
<td>MX</td>
<td>Domain</td>
<td>Refers to a mail server to handle mail addressed to this node</td>
</tr>
<tr>
<td>SRV</td>
<td>Domain</td>
<td>Refers to a server handling a specific service</td>
</tr>
<tr>
<td>NS</td>
<td>Zone</td>
<td>Refers to a name server that implements the represented zone</td>
</tr>
<tr>
<td>CNAME</td>
<td>Node</td>
<td>Symbolic link with the primary name of the represented node</td>
</tr>
<tr>
<td>PTR</td>
<td>Host</td>
<td>Contains the canonical name of a host</td>
</tr>
<tr>
<td>HINFO</td>
<td>Host</td>
<td>Holds information on the host this node represents</td>
</tr>
<tr>
<td>TXT</td>
<td>Any kind</td>
<td>Contains any entity-specific information considered useful</td>
</tr>
</tbody>
</table>

The most important types of resource records forming the contents of nodes in the DNS name space.
**DNS implementation (1)**

An excerpt from the DNS database for the zone `cs.vu.nl`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Record type</th>
<th>Record value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs.vu.nl</td>
<td>SOA</td>
<td><code>star (1999121502,7200,3600,2419200,86400)</code></td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>NS</td>
<td><code>star.cs.vu.nl</code></td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>NS</td>
<td><code>top.cs.vu.nl</code></td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>NS</td>
<td><code>solo.cs.vu.nl</code></td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>TXT</td>
<td>&quot;Vrije Universiteit - Math. &amp; Comp. Sc.&quot;</td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>MX</td>
<td><code>1 zephyr.cs.vu.nl</code></td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>MX</td>
<td><code>2 tornado.cs.vu.nl</code></td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>MX</td>
<td><code>3 star.cs.vu.nl</code></td>
</tr>
<tr>
<td>star.cs.vu.nl</td>
<td>HINFO</td>
<td>Sun Unix</td>
</tr>
<tr>
<td>star.cs.vu.nl</td>
<td>MX</td>
<td><code>1 star.cs.vu.nl</code></td>
</tr>
<tr>
<td>star.cs.vu.nl</td>
<td>MX</td>
<td><code>10 zephyr.cs.vu.nl</code></td>
</tr>
<tr>
<td>star.cs.vu.nl</td>
<td>A</td>
<td><code>130.37.24.6</code></td>
</tr>
<tr>
<td>star.cs.vu.nl</td>
<td>A</td>
<td><code>192.31.231.42</code></td>
</tr>
<tr>
<td>zephyr.cs.vu.nl</td>
<td>HINFO</td>
<td>Sun Unix</td>
</tr>
<tr>
<td>zephyr.cs.vu.nl</td>
<td>MX</td>
<td><code>1 zephyr.cs.vu.nl</code></td>
</tr>
<tr>
<td>zephyr.cs.vu.nl</td>
<td>MX</td>
<td><code>2 tornado.cs.vu.nl</code></td>
</tr>
<tr>
<td>zephyr.cs.vu.nl</td>
<td>A</td>
<td><code>192.31.231.66</code></td>
</tr>
<tr>
<td><a href="http://www.cs.vu.nl">www.cs.vu.nl</a></td>
<td>CNAME</td>
<td><code>soling.cs.vu.nl</code></td>
</tr>
<tr>
<td>ftp.cs.vu.nl</td>
<td>CNAME</td>
<td><code>soling.cs.vu.nl</code></td>
</tr>
<tr>
<td>soling.cs.vu.nl</td>
<td>HINFO</td>
<td>Sun Unix</td>
</tr>
<tr>
<td>soling.cs.vu.nl</td>
<td>MX</td>
<td><code>1 soling.cs.vu.nl</code></td>
</tr>
<tr>
<td>soling.cs.vu.nl</td>
<td>MX</td>
<td><code>10 zephyr.cs.vu.nl</code></td>
</tr>
<tr>
<td>soling.cs.vu.nl</td>
<td>A</td>
<td><code>130.37.24.11</code></td>
</tr>
<tr>
<td>laser.cs.vu.nl</td>
<td>HINFO</td>
<td>PC MS-DOS</td>
</tr>
<tr>
<td>laser.cs.vu.nl</td>
<td>A</td>
<td><code>130.37.30.32</code></td>
</tr>
<tr>
<td>vucs-das.cs.vu.nl</td>
<td>PTR</td>
<td><code>0.26.37.130.in-addr.arpa</code></td>
</tr>
<tr>
<td>vucs-das.cs.vu.nl</td>
<td>A</td>
<td><code>130.37.26.0</code></td>
</tr>
</tbody>
</table>
DNS implementation (2)

<table>
<thead>
<tr>
<th>Name</th>
<th>Record type</th>
<th>Record value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs.vu.nl</td>
<td>NS</td>
<td>solo.cs.vu.nl</td>
</tr>
<tr>
<td>solo.cs.vu.nl</td>
<td>A</td>
<td>130.37.21.1</td>
</tr>
</tbody>
</table>

Part of the description for the *vu.nl* domain which contains the *cs.vu.nl* domain.
Outline

• Names and Addresses
• Flat Naming
• Structured Naming
• Attribute-based Naming
Attribute-based naming

• An entity is described in terms of (attribute, value) pairs
• Each entity can have several attributes
• Attribute-based naming services are called directory services
• LDAP (Lightweight directory access protocol) de facto industry standard
  • Based on OSI X.500 directory service
• Another example: UDDI for web services
Example system: LDAP

• An example of a directory service
  • Analogy: LDAP is to DNS as the yellow pages are to a telephone book

• Each directory entry is made up of a collection of (attribute, value) pairs
  • Attributes can be single-valued or multiple-valued

• Collection of all directory entries is called a Directory Information Base (DIB)

• Each entry has a globally unique name formed by a sequence of naming attributes (Relative Distinguished Names or RDN)

• Lookup operations
  • Read: Read a single record given its pathname in the Directory Information Tree (DIT), i.e., hierarchical name space formed by directory entries
  • List: return the names of all outgoing edges of a given node in the DIT
The LDAP name space (1)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Abbr.</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country</td>
<td>C</td>
<td>NL</td>
</tr>
<tr>
<td>Locality</td>
<td>L</td>
<td>Amsterdam</td>
</tr>
<tr>
<td>Organization</td>
<td>O</td>
<td>Vrije Universiteit</td>
</tr>
<tr>
<td>OrganizationalUnit</td>
<td>OU</td>
<td>Math. &amp; Comp. Sc.</td>
</tr>
<tr>
<td>CommonName</td>
<td>CN</td>
<td>Main server</td>
</tr>
<tr>
<td>Mail_Servers</td>
<td>--</td>
<td>130.37.24.6, 192.31.231.42,192.31.231.66</td>
</tr>
<tr>
<td>FTP_Server</td>
<td>--</td>
<td>130.37.21.11</td>
</tr>
<tr>
<td>WWW_Server</td>
<td>--</td>
<td>130.37.21.11</td>
</tr>
</tbody>
</table>

A simple example of a LDAP directory entry using LDAP naming conventions.
The LDAP name space (2)

Part of the directory information tree (DIT).
The LDAP name space (3)

- Two directory entries having *Host_Name* as RDN.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country</td>
<td>NL</td>
<td>Country</td>
<td>NL</td>
</tr>
<tr>
<td>Locality</td>
<td>Amsterdam</td>
<td>Locality</td>
<td>Amsterdam</td>
</tr>
<tr>
<td>Organization</td>
<td>Vrije Universiteit</td>
<td>Organization</td>
<td>Vrije Universiteit</td>
</tr>
<tr>
<td>CommonName</td>
<td>Main server</td>
<td>CommonName</td>
<td>Main server</td>
</tr>
<tr>
<td>Host_Name</td>
<td>star</td>
<td>Host_Name</td>
<td>zephyr</td>
</tr>
<tr>
<td>Host_Address</td>
<td>192.31.231.42</td>
<td>Host_Address</td>
<td>192.31.231.66</td>
</tr>
</tbody>
</table>
LDAP implementation

- Becoming a de facto standard for Internet-based directory services
- Similar to DNS
  - The DIT is partitioned and distributed across several servers known as Directory Service Agents (DSA)
  - Clients are represented by name resolvers called Directory User Agents (DUA)
- Differences from DNS
  - Operations for searching through a DIB given a set of criteria that attributes should meet
  - Searching is an expensive operation since several leaf nodes of a DIT will need to be accessed
Reading

• Chapter 5 of Tbook
• Paper on course homepage about issues in naming in distributed systems