CSE 5306
Distributed Systems

Naming
Naming

- Names play a critical role in all computer systems
  - To access resources, uniquely identify entities, or refer to locations
- To access an entity, you have resolve the name and find the entity
  - Name resolution provided by the naming system
- In a distributed system, the naming system itself is implemented across multiple machines
  - Efficiency and scalability are the keys
Addresses

- To access an entity, we need the access point, which is a special entity.
  - The name of an access point is an address.

- An entity may have multiple access points, and its access point may change.
  - Thus, the address of an access point should not be used to name the entity.

- E.g., each person has multiple phone numbers to reach him, and these numbers may be re-assigned to another person.

- Therefore, what we need is a name for an entity that is independent from its addresses.
  - i.e., a location-independent name.
True Identifiers

• Are the names that are used to uniquely identify an entity in a distributed system

• True identifiers has the following property
  – Each identifier refers to at most one entity
  – Each entity referred to by at most one identifier
  – An identifier always refers to the same entity (prohibits reusing an identifier)

• A simple comparison of two identifiers is sufficient to test if they refer to the same entity
The Central Theme of Naming

- How to resolve names and identifiers to addresses
- In principle, a naming system maintains a name-to-address binding in the form of mapping table
  - However, a centralized table in a large network is not going to work
- The name resolution as well as the table is often distributed across multiple machines
  - E.g., www.cse.uta.edu is divided into multiple parts, and several DNS servers are used during the resolution
- Will discuss the resolution of the following names
  - Flat names, structured names, attribute-based names
Flat Names (Identifiers)

• An identifier is often a string of random bits
  - does not contain any information on how to locate the access point of its associated entity

• Two simple solutions to locate the entity given an identifier
  - Broadcasting and multicasting (e.g., ARP)
    • Broadcast is expensive, multicast is not well supported
  - Forwarding pointers
    • When an entity moves, it leaves a pointer to where it went
    • A popular approach to locate mobile entities
Forwarding Pointers

• Advantage:
  - Dereferencing can be made transparent to client - follow the pointer chain

• Geographical scalability problems:
  - Chain can be very long for highly mobile entities
  - Long chains not fault tolerant
  - High latency when dereferencing

• Need chain reduction mechanisms
  - Update client's reference when the most recent location is found
Forwarding via Client-Server Stubs

The principle of forwarding pointers using (client stub, server stub) pairs.
Chain Reduction via Shortcuts

(a) Invocation request is sent to object. Server stub at object's current process returns the current location.

(b) Server stub is no longer referenced by any client stub. Client stub sets a shortcut.
Home-Based Approaches

The principle of Mobile IP.
Issues with Home-Based Approaches

- Home address has to be supported as long as entity lives
- Home address is fixed – unnecessary burden if entity permanently moves
- Poor geographical scalability (the entity may be next to the client)
Distributed Hash Table

- In DHT-based system Chord,
  - Each node has an m-bit random identifier
  - Each entity has an m-bit random key
  - An entity with key k is located on a node with the smallest identifier
    - that satisfies id>=k, denoted as succ(k)

- The major task is key lookup, i.e.,
  - I.e., to resolve an m-bit key to the address of succ(k)
  - Two approaches: linear approach and finger table

- The simplest form of chord does not consider network proximity, leading to performance problem
Key Lookup in Chord

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.
Structured Naming

- Flat names are not convenient for humans to use
- As a result, naming systems often support structured names that
  - are composed from simple, human-readable names
  - E.g., file names, Internet domain names
- Structured names are often organized into what is called a name space
  - A labeled, directed graph with two types of nodes, leaf node and directory node
Name Space

A general naming graph with a single root node.
Name Resolution

• The process of looking up a name in a name space

• Name resolution can take place only if we know where and how to start
  - A closure mechanism
  - E.g., start from a well know root directory, or start from home directory

• Linking
  - Aliases are commonly used in a name space
  - An alias can be a hard link or a symbolic link
Symbolic Link

The concept of a symbolic link explained in a naming graph.
Mounting (1)

- The process of merging different name spaces
- A common approach is to
  - Let a directory node (mount point) store the identifier of a directory node (mounting point) from the foreign name space
- 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 (2)

Mounting remote name spaces through a specific access protocol.
Implementation of a Name Space

- A name space is often implemented by name servers
  - in LAN, a single name server is often good enough
  - in large-scale distributed system, the implementation of a name space is often distributed over multiple name servers

- A name space for large-scale distributed systems is often organized hierarchically
  - Global layer
    - often stable, represents organizations or groups of organizations
  - Administrative layer
    - Represents groups of entities in a single organization or admin. unit
  - Managerial layer
    - Nodes often change frequently, e.g., hosts in a local network
    - May be managed by system administrators or end users
An example partitioning of the DNS name space, including Internet-accessible files, into three layers.
### Name Space Distribution (2)

<table>
<thead>
<tr>
<th>Item</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</td>
<td>Immediate</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>Is client-side caching applied?</td>
<td>Yes</td>
<td>Yes</td>
<td>Sometimes</td>
</tr>
</tbody>
</table>

A comparison between name servers for implementing nodes from a large-scale name space partitioned into a global layer, an administrative layer, and a managerial layer.
The principle of iterative name resolution.
Implementing Name Resolution (2)

The principle of recursive name resolution.
Recursive v.s. Iterative

- Recursive resolution demands more on each name server
- However, it has two advantages
  - Caching is more effective than iterative name resolution
    - Intermediate nodes can cache the result (faster look-up)
    - With iterative solution, only the client can cache
  - Overall communication cost can be reduced
Example: The Domain Name System

- The DNS name space is organized as a root tree
- Each node in this tree stores a collection of resource records

<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>
Decentralized DNS Implementation

- In standard hierarchical DNS implementation, higher-level nodes receive more requests than low-level nodes
  - leading to a scalability problem
- Fully decentralized solution can avoid such scalability problem
  - Map DNS names to keys and look them up in a distributed hash table
  - The problem is that we lose the structure of the original names and make some operations difficult
Attribute-Based Naming

- As more information being made available, it becomes important to
  - locate entities based on merely a description of what is needed

- Attribute-based naming
  - Each entity is associated with a collection of attributes
  - The naming system provides one or multiple entities that match a user’s description

- Attribute-based naming systems are often known as directory services
Hierarchical Implementation LDAP

<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>Comp. Sc.</td>
</tr>
<tr>
<td>CommonName</td>
<td>CN</td>
<td>Main server</td>
</tr>
<tr>
<td>Mail_Servers</td>
<td>—</td>
<td>137.37.20.3, 130.37.24.6, 137.37.20.10</td>
</tr>
<tr>
<td>FTP_Server</td>
<td>—</td>
<td>130.37.20.20</td>
</tr>
<tr>
<td>WWW_Server</td>
<td>—</td>
<td>130.37.20.20</td>
</tr>
</tbody>
</table>

A simple example of an LDAP directory entry using LDAP naming conventions.
Directory Information Tree (DIT)

For a large-scale directory, the DIT is often distributed over several servers as in DNS.

Part of directory information tree.
Each path in attribute-value tree (AVT) produces a hash value and mapped to a DHT

- $h_1 = \text{hash(type-book)}$, $h_2 = \text{hash(type-book-author)}$
- $h_3 = \text{hash(type-book-author-Tolkien)}$, $h_4 = \text{hash(type-book-title)}$
- $h_5 = \text{hash(type-book-title-LOTR)}$, $h_6 = \text{hash(genre-fantasy)}$
Ranged Query in DHT implementation

- **Two phase approach**
  - Separate the name and the attribute in computing the hash value
  - Phase 1: distribute attribute names in DHT
  - Phase 2: for each name, partition the values into subranges and assign a single server for each subrange

- **Drawbacks**
  - Updates may need to be send to multiple servers
  - Load balancing between different subrange servers
Semantic Overlay Networks

- Construct an overlay network where each pair of neighbors are semantically proximal neighbors
  - i.e., they have similar resources

![Diagram showing semantic overlay networks with protocol and random peer connections.]

UTA
Department of Computer Science and Engineering