Naming

527950-1
Fall 2019
10/24/2019
Kyoung Shin Park
Applied Computer Engineering
Dankook University

Definition

- **Name** identifies what you want
- **Address** identifies where it is
- **Route** identifies how to get there
- **Binding** is the association of a name with the object
  - "choose a lower-level implementation for a higher-level semantic construct"
- **Context** is a particular set of bindings. A name only has meaning relative to some context.
- **Directory** or **Naming Network** is a set of catalogs (name to object binding tables) that may include other directories.

Definition

- **Naming hierarchy** is a naming network organized in a tree-structured form.
- **Pathname** is a multi-component name traversing a path in a naming hierarchy.
- **Root** is a starting catalog in a naming network.
- **Indirect entry** is an entry in a catalog that binds to a name instead of the underlying object.
- **Name service** is a service that provides a binding function. A computing environment will have many names, each relevant within a specific context.

Naming

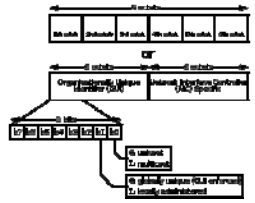
- Naming
  - **Process of mapping a name to an object**
  - Helps with using, sharing, and communicating information
- Examples
  - **User name**: used for system login, email, chat
  - **Machine name**: used for ssh, email, web
  - **Filename/Pathname**
  - **Device name**
  - **Objects, functions, variables** in programs
  - **Network services**

Name

- **Pure names - identify**
  - The name contains no information (about the object or where that object may be found) aside from the name
  - For example
    - **User ID (kpark)** is a pure name
    - **Ethernet MAC address** is pure name since it does not tell you anything about the device or where it located.
- **Impure names - guide**
  - The name contains context information (about the underlying object encoded within the name)
  - If the object is moved, the name is no longer relevant.
  - For example
    - **Email address** (kpark@dankook.ac.kr) is an impure name since it contains a domain name as a context
    - **Domain name** (www.dankook.ac.kr) is another example

Name

- **Uniqueness of names**
  - Easy on a small scale - problematic on a large scale
    - It can be difficult to make globally unique names
- Uniqueness for pure names
  - Designate a bit pattern or naming prefix that does not convey information
    - **Ethernet MAC address:**
      3 bytes: organization, 3 bytes: controller



- Uniqueness for impure names
  - Use a **hierarchy**
  - Globally unique components (pure names)
    - **Compound name** - **iterative list of pure names connected with separators**
      - Domain name use .
      - File pathnames use /

Naming Convention

- Naming system determines **syntax** for names
- Naming convention can take any format
  - Ideally one that will suit the application and user
  - UNIX file names
    - Parse components from **left to right** separated by **/**
    - /home/park/test.txt
  - Internet domain names
    - Ordered **right to left** and delimited by **.**
    - www.dankook.ac.kr
  - LDAP names
    - **Attribute/value pairs** ordered right to left, delimited by **,**
    - cn=Kyoung Park, o=dankook, c=kr

Context

- **A particular set of name to object bindings**
  - Names are unique within the context
    - /home/park/test.txt on a specific computer
  - Each context has an associated naming convention
  - A name is always interpreted relative to some context
    - Directory /usr in Linux file system on a specific computer

Naming System

- Naming System
  - **Connected set of contexts** of the same type (same naming convention) along with a common set of operations
- For example
  - System that implements DNS (Internet domain names)
  - System that implements LDAP (Directory of people)

Namespace

- A container for a **set of names** in the naming system
- A namespace has a scope
  - **Scope** = region where the name exists & refers to the object
  - E.g. Names of all files in a directory
  - E.g. All domain names within dankook.ac.kr
  - E.g. Java package, local variables
- A namespace may be tree structured (hierarchical)
  - Fully-qualified or hierarchical names may be used to identify names outside the local namespace
  - Global namespace = root of the tree

Name Resolution

- Resolution = **name lookup**
  - Return the underlying representation of the name
  - Look up the **binding** of the name to its object
- E.g. dis.dankook.ac.kr -> 220.149.232.78

```
~>nslookup dis.dankook.ac.kr
Server:    203.237.226.1
Address:  203.237.226.1#53

Name:    dis.dankook.ac.kr
Address:  220.149.232.78
```

Name Resolution

- **Iterative** resolution
  - E.g. parse a pathname
- **Recursive** resolution
  - E.g. parse a distribution list - each entity may be **expanded**
- Binding
  - **Static binding** – hard-coded
  - **Early binding** – look up binding before use; cache previously used binding
  - **Late binding** – look up just before use

Name Service

- The **service** that performs **name resolution**
- Allows you to resolve names
  - Looking up a name gives the corresponding address as a response
- Can be implemented as
  - Search through file
  - Database query
  - Client-server program (name server) – may be distributed
  - …

Directory Service

- Extension of name service
  - Associates names with objects
  - Allows objects to have attributes
  - Can search based on attributes
- E.g. **LDAP (Lightweight Directory Access Protocol)**
  - Directory can be an object store
  - Lookup printer object and send data stream to it

Case Study: DNS (Domain Name System)

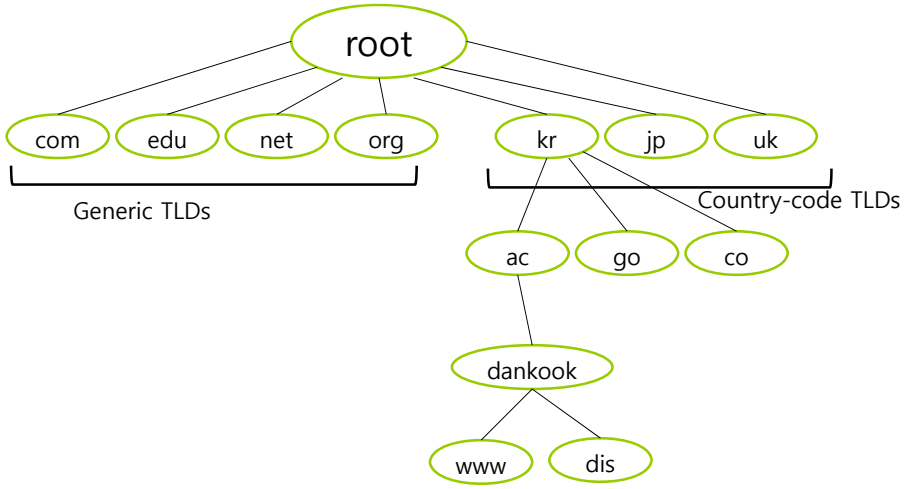
- IP addresses are distributed hierarchically
- Internet Assigned Numbers Authority (IANA) at the top
  - IANA is currently run by ICANN (Internet Cooperation for Assigned Names and Numbers)
- RIR (Regional Internet Registry)
  - Manages the allocation and registration of Internet number resources within a particular region of the world

RIR (Region Internet Registry) Map

Domain Name Hierarchy

- Early ARPANET
  - Globally unique names per machine (e.g. UCBVAX)
  - Kept track at the Network Information Center (NIC) at the Stanford Research Institute (SRI)
  - That doesn't scale!
- A **domain name hierarchy** was created in 1984 (RFC 920)
  - Domains are administrative entities – divide name management
  - Tree-structured global name space
  - Human readable textural representation of domain names
    - E.g. www.dankook.ac.kr

Domain Name Hierarchy



Top Level Domains (TLDs)

- There are currently 1097 top-level domains
- Each top-level domain has an administrator assigned to it
- Assignment is delegated to various organizations by the Internet Assigned Numbers Authority (IANA)
- **IANA** keeps track of the **root servers**
- http://www.iana.org/domains/root/db
- **.kr -> Korea Internet & Security Agency (KISA)**

Shared Registration

- Domain name registry
  - This is the database
  - Keeps track of all domain names registered under a top-level domain
- Domain name registry operator
  - This is the company that runs the database
  - NIC (Network Information Center)
    - organization that keeps track of the registration of domain names under a top-level domain
    - Keeps the database of domain names
- Domain name registrar
  - This is the company you use to register
  - Company that lets you register a domain name
  - Registrars update the registry databased at the NIC

Shared Registration

- Problem
  - Every device connected to the internet has a unique Internet Protocol (IP) address
  - How do you **resolve** user-friendly machine names to IP addresses? www.dankook.ac.kr -> 220.69.176.17
- Original solution
  - Through the 1980s, search **/etc/hosts** file for machine name (RFC 606)
  - File periodically downloaded from NIC at SRI
  - **This doesn't scale** with millions of hosts on the Internet
    - A lot of data
    - A lot of churn in the data – new hosts added, deleted, changed
    - Maintenance
    - Traffic volume

Domain Name System (DNS)

- **Distributed database - a hierarchy of name servers**
- **DNS** is an application-layer protocol
  - Name-address resolution is handled at the edge
  - The network core is unaware of hostnames
  - There is **no special relationship between names and addresses**

Domain Name System (DNS)

- **DNS is a distributed, hierarchical database.**
- DNS provides
  - Name to IP address translation
  - Aliasing of names (called **canonical names**)
  - Identification of name servers
  - Mail server names
  - Load distribution
    - Multiple name servers may handle a query for a domain
    - Caching - store past lookups
    - Ability to provide a set of IP address for a name

Hierarchy of Name Server

- Root name server
  - The **root name server** answers can return a list of authoritative name servers for top-level domains
  - 13 root name server addresses – 386 servers
    - Each has redundancy (via anycast routing or load balancing)
    - Each server is a set of machines
- Top level server
  - The **top name server** is responsible for com, org, edu, and all top level country domains.
  - It has information about authoritative domain servers and know names and IP addresses of each authoritative name server.
- Authoritative name server
  - This is **organization(or service provider)'s DNS server**.

Authoritative DNS Server

- An **authoritative name server** is responsible for answering queries about its zone
  - Provides real answers vs. cached answers
  - Configured by the administrator
- **Zone**
  - **Group of machines under a node in the tree** (such as organization or service provider)
  - E.g. dankook.ac.kr

DNS Server

- DNS server returns answers to queries
  - Key data that a DNS server maintains (partial list)

Information	Abbreviation	Description
Host	A	Host address (name to address) including name, IP address, time-to-live (TTL)
Canonical name	CNAME	Name for an alias
Mail exchanger	MX	Host that handles email for the domain
Name server	NS	Identifies the name server for the zone: tell other servers that yours is the authority for info within the domain
Start of Zone Authority	SOA	Specifies authoritative server for the zone. Identifies the zone, time-to-live, and primary name server for the zone

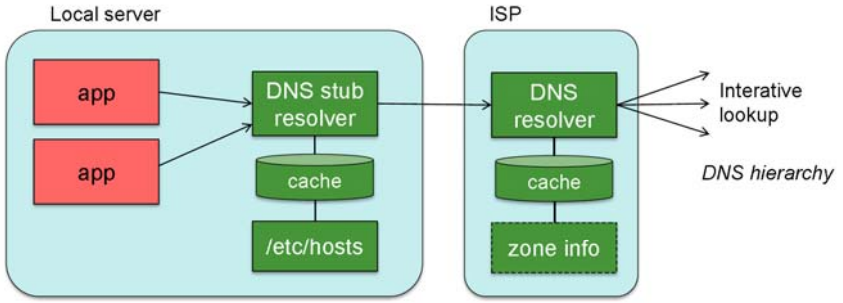
Name Resolution Approaches

- **Iterative** (non-recursive) name resolution
  - Send query to root name server
  - Send query to kr name server
  - Send query to ac name server
  - Send query to dankook name server
  - **Advantage - stateless**
- **Recursive** name resolution
  - Send query to root name server
  - Root name server sends query to kr name server
  - Kr name server sends query to ac name server
  - Ac name server sends query to dankook name server
  - **Advantages – increased caching opportunities, reduced communication**

DNS Resolver: Local Name Server

- **The client side of DNS** is called a **DNS resolver**.
  - Not really a part of DNS hierarchy
  - Acts as an intermediary between programs that need to resolve names and the name servers
  - **A resolver is responsible for performing the full resolution of the query**
- Where are DNS resolvers?
  - Each local system has one: that's what applications contact
    - Local cache - may be a process or a library
    - On Linux & Windows, these are limited DNS servers (called stub resolvers) – Usually not capable of handling referrals and expect to talk with a name server that can handle recursion (full resolution)
  - ISPs (and organizations) run them on behalf of their customers
    - Including a bunch of free ones (OpenDNS, Google Public DNS)
- Resolvers cache past lookups – they are not responsible for zones

DNS Resolver



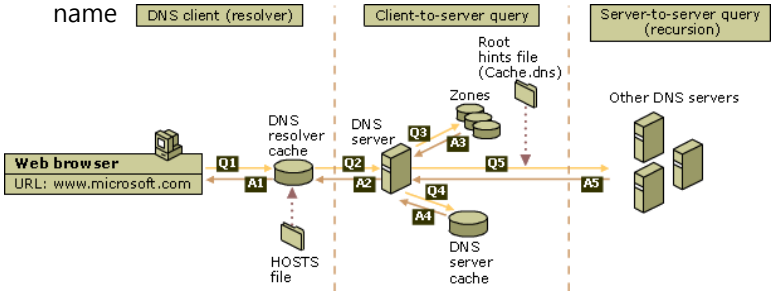
Local stub resolver:
- check local cache
- check local hosts file
- send request to external resolver

E.g., on Linux: resolver is configured via the /etc/resolv.conf file

External resolver
- DNS server that accepts recursion
- Running at ISP, Google Public DNS, OpenDNS, etc.

DNS Query

- When a DNS client needs to look up a name used in a program, it queries DNS servers to resolve the name.
- DNS query process
  1. **Local resolver**: A name query begins at a client computer and is passed to a resolver (DNS client service) for resolution
  2. **Query a DNS server**: When the query cannot be resolved locally, DNS servers can be queried as needed to resolve the name



Caching

- Starting every query at the root would place a huge load on root name servers
- A name server can cache results of previous queries
  - Save query results for a time-to-live amount of time
  - The time-to-live value is specified in the domain name record by an authoritative name server

Distributed File System

- Goals of Distributed File System
  - Network Transparency
    - Looks like a traditional file system on a mainframe
    - User need not know a file's location
  - High Availability
    - Users should have easy access to files, wherever the users or files are located
    - Tolerant of failures

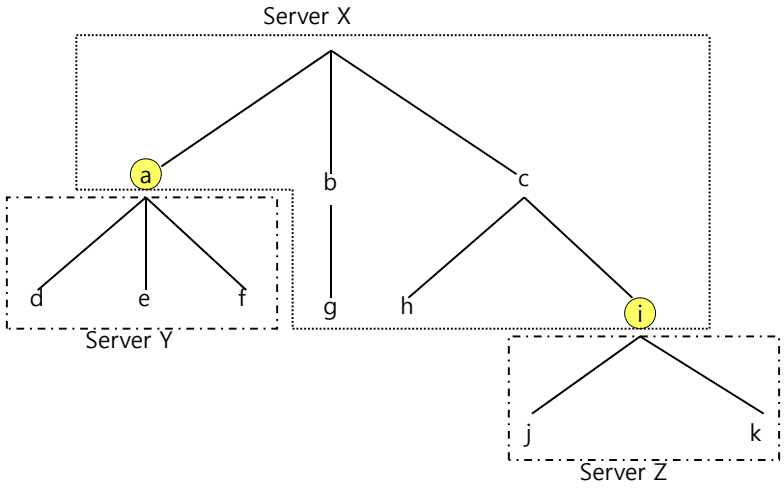
DFS Architecture

- On the network
  - File servers - hold the files
  - Clients - make accesses to the servers
- Name Server
  - Maps names to directories/files
- Cache Manager
  - Implements file caching
  - Often at both server and clients
  - Coordinates to avoid inconsistent file copies

Mechanisms of a DFS

- Mounting
  - Binding together of different filename spaces to form a single name space
  - A name space is mounted to (or bounded to) a mount point (or node in the name space)
  - Need to maintain mount information
    - Keep it at the clients
    - Keep it at the servers

Name Space Hierarchy

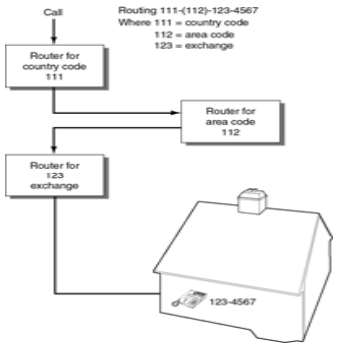


Naming

- **Name Resolution**
  - "The process of mapping a name to an object, or in the case of replication, multiple objects"
- **Name Space**
  - "A collection of names which may or may not share an identical resolution mechanism"

Location Transparency

- Must be provided via **global naming**
- Dependent on a name being **location independent**, (i.e., a universal name)
- E.g. phone number



File Naming

- On a PC, the filename consists of a drive letter followed by a pathname
  - G:₩MM₩courses₩DistributedSystem2019₩lecture0.pdf
- In Unix, the filename does not contain a drive letter, but the mount table enables the OS to discover on what drive the file is located
  - /home/park/test.txt
  - File pathname changes if you move the object

Process Naming

- Processes that want to communicate must have a way to refer to each other. They can either use *direct communication* or *indirect communication*.
- **Direct**
  - **send** (*P*, *message*) send a *message* to *process P*
  - **receive** (*Q*, *message*) receive a *message* from *process Q*
  - **receive** (*id*, *message*) receive a *message* from any process
    - where variable *id* is set to the name of the process from which communication has taken place.
- **Indirect**
  - **send** (*A*, *message*) send a *message* to mailbox *A*
  - **receive** (*A*, *message*) receive a *message* from mailbox *A*
    - where A is a shared mailbox or port

System Naming

- The first component of network communication is the **naming of the systems** in the network.
  - For a process at site A to exchange information with a process at site B, they must be able to specify each other.
  - Processes on remote systems are generally identified by the pair <**hostname**, identifier>
- **Connected set of context** of the same type (same naming convention) along with a common set of operations
- E.g. System that implements **DNS (Domain Name System)**
- E.g. System that implements **LDAP (Lightweight Directory Access Protocol)**

Global Naming Considerations

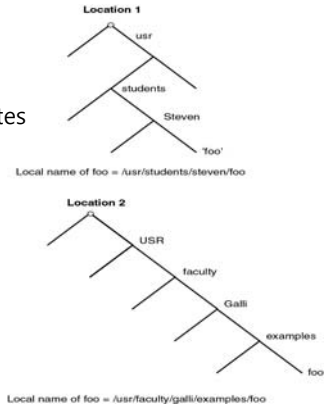
- A global name space requires
  - Name resolution
  - Location resolution
- **Name resolution**
  - Maps symbolic filenames to computer filenames
- **Location resolution**
  - Involves mapping global names to a location
- This can be difficult if name transparency and location transparency are both supported

Naming Approaches

- **Add hostname to names of files** on that host
  - Provides unique names
  - Loses network transparency
  - Loses location transparency
  - Moving file to a different host causes change of filename
    - Possible changes to applications using that file
  - Easy to find a file

Naming Approaches

- **Mount remote directories** onto local directories
  - To do the mount, need to know host
  - Once mounted, references are location transparent
  - Can resolve filenames easily
  - However, a difficult approach to do
    - Not fault tolerant
    - File migration requires lots of updates



Naming Approaches

- Use **a single global directory**
  - Does not have disadvantages of previous approaches
  - Need a single computing facility or a few with lots of cooperation
    - Need system-wide unique filenames
  - Not good on a heterogeneous system
  - Not good on a wide geographic system

Naming Issues

- Contexts
  - Used to **partition a namespace**
    - To avoid problems with system-wide unique names
    - Geographical, organizational, etc.
  - A name space in which to resolve a name
  - A filename has two parts
    - Context
    - Local filename
  - Almost like another level of directory

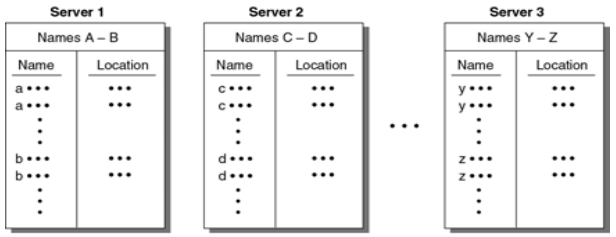
Naming Issues

- **Name Server**
  - Maps names to files and directories
  - Centralized
    - Easy to use
    - A bottleneck
    - Not fault tolerant
  - Distributed
    - Servers deal with different domains
    - Several servers may be needed to deal with all the components in a filename

Removing Unreferenced Entities

- Problem of unreferenced objects
- Reference counting
- Reference listing
- Identifying unreachable entities

Distributed Solution for Name Resolution



Server Location Mapping Table

Name Range	Master Server	Additional
A – B	Server 1	
C – D	Server 2	
⋮		
Y – Z	Server N	

References

- https://www.cs.rutgers.edu/~pxk/417/notes/naming.html
- www.cs.colostate.edu/~cs551/CourseNotes/Naming/NamingTOC.html (DFS)