Week 3: Introduction
Naming & binding

Paul Krzyzanowski
Naming things

• Naming: map names to objects

• Examples
  – User names
  – Machine names
  – Files
  – Devices
  – Classes, functions, variables in programs
  – Network services
What’s a name?

**Name**: identifies what you want

**Address**: identifies where it is

**Route**: identifies how to get there

**Binding**: the association of a name with the object

“choose a lower-level-implementation for a higher-level semantic construct”

— RFC 1498: Inter-network Naming, addresses, routing
What’s a name?

**Binding**: the association of a name with the object

“choose a lower-level-implementation for a higher-level semantic construct”

`ls.cs.rutgers.edu → 128.6.13.171`
Pure & Impure Names

Pure names – *identify*

– The name contains no information aside from the name

– Examples:
  
  • c8:2a:14:3f:92:d1  my computer’s ethernet MAC address
  • p_k  my Twitter handle
  • 908-555-3836  phone # (this used to be an impure name)
Impure names – *guide*

- The name contains context information

- Examples:
  - `pk@pk.org`, `pxk@cs.rutgers.edu`
  - `happyuser@verizon.net`
  - `/home/paul/bin/qsync`
    - File pathname changes if we move the object
Uniqueness of names

• Easy on a small scale – problematic on a large scale

• 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
    • IP address: network & host (variable partition)

• Uniqueness for impure names — use a hierarchy
  – Compound name: iterative list of pure names connected with separators
    • Domain name: www.cs.rutgers.edu
    • URLs: http://pk.org/417/lectures/intro.html
    • File pathnames: /usr/share/dict/words
Naming convention determines syntax for names

Ideally, a format that will suit the application and user

- UNIX file names:
  /home/paul/src/gps/gui.c

- Internet domain names:
  • www.cs.rutgers.edu

- LDAP names
  • cn=Paul Krzyzanowski, o=Rutgers, c=US
A particular set of *name* → *object* bindings

- Names are unique within the context
  - E.g., `/etc/postfix/main.cf` on a specific computer

- Each context has an associated naming convention
Terms: Naming System = service

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)
Terms: Namespace = entire set of names

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

- A namespace may be tree structured (hierarchical)
  - Fully-qualified or hierarchical names may be used to identify names outside the local namespace
Terms: Resolution

- **Resolution** = name lookup
  - Return the underlying representation of the name
  - Look up the **binding** of the name to its object

- For example,
  - `www.rutgers.edu → 128.6.4.5`

- **Iterative** resolution
  - Example: parse a pathname

- **Recursive** resolution
  - Example: parse a distribution list: each entity may be expanded
When do you should you do a resolution?

**Static binding**
- Hard-coded

**Early binding**
- Look up binding before use
- Cache previously used binding

**Late binding**
- Look up just before use

These can cause problems!
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

• For example,
  – LDAP (Lightweight Directory Access Protocol)
  – Directory can be an object store:
    • E.g., look up printer object and send data stream to it
The End