Names

March 28, 2020

Outline

Server/Object Location

Problem: How does a client know where is the server?

Solution: Not one, but several alternatives:

- *hard coded*, seldom;
- via program arguments: more flexible, but ...;
- via configuration file;
- via *broadcast*/*multicast*;
- via a location/name service:
  - local, e.g. *rmiregistry*.
  - *global*.

Addresses vs. Names

- Names are ... sequences of symbols (bits/characters/...) that refer to entities/objects.
- In the labs, we have used IP addresses (and ports)
- Addresses are **names** of **access points**. Or as Shoch put it:

  *The **name** of a resource indicates **what** we seek,*
  *an **address** indicates **where** it is,*
  *(and a **route** tells us **how** to get there.)*
- Addresses have some limitations:
  - Addresses often are location dependent and change frequently
    - E.g. when a service is moved from one computer to another
- Names have some advantages over addresses:
  - They can be human-friendly.
  - They can hide both complexity and dynamics
    - E.g. they can hide access point changes
- Naming is a layer of indirection
  - Ultimately you need an address to access/operate on an object

Identifiers

- An **identifier** is a name with 3 properties:
  1. an identifier refers to one entity at most;
  2. an entity has at most one identifier;
  3. an identifier refers always to the same entity (it is never reused).
- Identifiers provide a mean to refer to an entity in a precise way, independently of its access points.
- Examples?
  - From the "real" world?
  - From the "virtual" world?

Pure Names

- Are names that contain no information whatsoever about what they refer to:
  - Not only about location, but about anything else
  - They do not commit the system to anything
  - They are useful only for comparison

Problems/challenges of pure names

- where to look them up to find out information about them?
- how do you know that an object does not exist? How can a global search be avoided?
- how to engineer uniqueness reliably in a distributed system?

Problems/challenges of impure names

- what if the information yielded by the name, e.g. location, is not valid anymore?
  - This is specially relevant for mobile systems, and requires appropriate solutions

Bindings, Contexts and Name Resolution

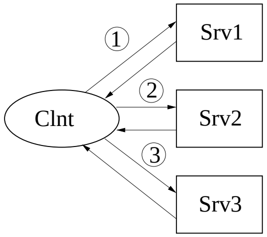
- A **binding** is a mapping from a name to an object/entity (usually identified by a lower-level name, e.g. address)
- A context/name space is a set of **bindings**
- A name space defines:
  - the syntax and structure (flat vs. hierarchical) of a name
  - the rules to find a binding of a name (**name resolution**)
- **Name resolution** is the process of finding a binding for a name
- A name is always resolved in the context of its name space:

file name	->	OS filesystem
Java program variable	->	JVM executing the program
ISBN of a publication	->	ISBN (Intern.Standard Book Number)
Car license plates	->	national/regional license plate regist

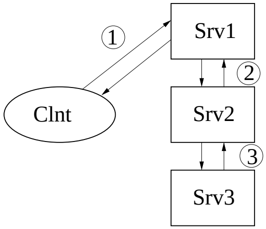
Name Resolution in a Distributed System

- Usually, name resolution is done with the help of a name service
- In small scale distributed systems, name resolution requires only one server:
  - E.g., the `rmiregistry`
- In distributed systems of larger scale, name resolution may require more than one server. In this case, name resolution can use one of 3 strategies:
  1. Iterative
  2. Recursive.
  3. Transitive.

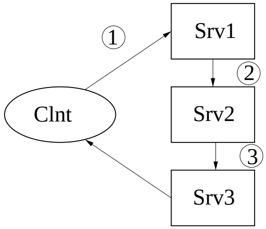
Name Resolution: Strategies



Iterative



Recursive



Transitive

- Recursive name resolution:
  - Allows for caching at servers
    - This may make resolution more efficient (with lower communication costs)
  - But, it:
    - requires servers to keep state
    - makes it harder to set the values of timeouts
- Transitive name resolution also makes it harder for the client to set a timeout value

Name Resolution and *Closure Mechanism*

Names are resolved always in a context

Problem

- How do you get a context that you can use to resolve a name?
    - How do you get a "remote reference to the `rmiregistry`"?
    - How to start the name resolution of a name of a file system: i.e. where is the root directory?
    - How to find the IP address of a DNS server to resolve a DNS name?

Response

Use a **closure mechanism**

- Typically this is an *ad-hoc* and simple solution.

Hierarchical Name Spaces

- Most name spaces have a hierarchical structure:
  - OS filesystem
  - Domain Name System (DNS)
  - Postal addresses
  - Car license plates are resolved in another context – per country, region etc.
- A hierarchical structure simplifies:
  - the assignment;
  - the resolution

  of names
- Allows to partition a name space into naming domains
  - Often, a naming domain has an administrative authority for assigning names within it
  - An administrative authority may delegate name assignments for sub-domains (e.g. in DNS)

Additional Reading

- Chapter 5 of van Steen and Tanenbaum, *Distributed Systems, 3rd Ed.*
  - Section 5.1: *Names, Identifiers and Addresses*
  - Section 5.3: *Structured Naming*
- J. Saltzer, *On the Naming and Binding of Network Destinations*, in RFC 1498, 1993