Today

- Naming
  - Basics
  - Flat Naming

Naming

- Names used to refer to entities:
  - Files, hosts, processes, devices
- Name resolution:
  - Find the named entity (access it, find its location)
- Distributed naming:
  - Entities are distributed
  - Naming system itself may be distributed

Naming: Issues

- How are names specified?
- Where are names stored?
- How do we locate an entity given its name?
Types of Names

- Addresses
- Identifiers
- Human-friendly names

Addresses

- Name of an entity’s access point
  - Provides a means to access the entity
  - E.g.: IP address, port no., etc.
- We could use an address as the name of an entity
  - E.g.: IP address for a machine, http server’s TCP-tuple, etc.
- Problems?

Location-Independence

- What happens to the name if:
  - The entity is moved?
  - The entity has multiple addresses?
- Location-independent name:
  - Does not depend on the address/location of the entity

Identifiers

- Name used to uniquely identify an entity
  - E.g.: inode, unique key
  - Often specified in machine-readable form
  - Context-dependent: IP address can be considered an identifier or an address
**Identifiers and Human-friendly Names**
- Identifier: Name used to uniquely identify an entity
  - E.g.: inode, unique key
  - Often specified in machine-readable form
  - Context-dependent: IP address can be considered an identifier or an address
- Human-friendly names
  - Typically a sequence of characters
  - E.g.: host name, file name, URL

**Naming Systems**
- Provide name resolution: How do we resolve a name to its address?
- Flat Naming
  - Used for identifiers (names without any implicit structure)
- Structured Naming
  - Used for human friendly names (with structure)
- Attribute-based Naming
  - Used for descriptive names (that describe what the entities are)

**Flat Naming**
- Unstructured names
  - Any entity can have any name
  - All names are semantically equivalent
  - No information about location
  - Typically a sequence of bits. E.g.: IDs, keys

**Flat Name Resolution**
- Broadcasting and Multicasting
- Forwarding Pointers
- Distributed Hash Tables
Broadcasting and Multicasting

- Two steps:
  - Broadcast the name
  - Named entity responds with address
- Example: ARP (Address Resolution Protocol)
  - Convert IP address to MAC address
- Could use multicast instead of broadcast for focused queries
- Limitations?

Forwarding Pointers

- Leave a pointer to the new location if the entity moves
- Name resolution:
  - Get initial location of an entity
  - Follow chain of pointers
- Limitations?

Home-based Approach

- Each entity has a home location
  - Keeps track of current address of entity
- Example: Mobile IP
  - Each host has a home IP address
  - Gets care-of address at new location
  - Clients connect to home IP address initially
    - Packets tunneled to C/O address

Distributed Hash Tables

- An identifier is a key in a hash space
- Each node is assigned a key
- Entities are also assigned keys from the same space
  - Mapped to “closest”-key nodes
  - E.g.: file placed on node with next highest key
Name Resolution

- Use Chord DHT system as example
- Given key $k$, find $\text{succ}(k)$
  - $\text{Succ}(k)$: The node that holds $k$
  - E.g.: node with smallest id $\geq k$
- Name resolution:
  - Each node holds pointer to its successor and predecessor
  - Forwards the key in the appropriate direction

Scalable Name Resolution

- Keep shortcuts to distant parts of the id-space
- Finger Table
  - If $m$-bit hash space, keep table of $m$ entries
  - $\text{FT}_p[i] = \text{succ}(p+2^{i-1})$
  - $i$-th entry corresponds to a distance of at least $2^{i-1}$
- Lookup $k$: Forward to entry $\text{FT}_p[j]$ s.t.
  - $\text{FT}_p[j] \leq k$ and
  - $\text{FT}_p[j+1] > k$

Handling Node Churn

- Node joining
  - Lookup successor
  - Announce itself to successor and predecessor
  - Initialize finger table entries accordingly
- Keeping finger tables updated
  - Periodically send keep-alive messages to successor and predecessor
  - Check that tables are consistent

Increasing Efficiency

- Topology-based key assignment
  - Incorporate network into the hash function
- Proximity neighbor selection
  - If multiple choice of neighbors on join, select the closest neighbor
- Proximity routing
  - Keep multiple choices for each finger table entry
  - Route to the closest node