Naming

Prof. Dave Bakken

Cpt. S 464/564 Lecture

November 15+27, 2000

Naming: © 2000 David E. Bakken

Naming in Context

- "What's in a **name**? That which we call a rose by any other name would smell as sweet"
  - Shakespeare, *Romeo and Juliet*
- "And if his **name** be George, I'll call him Peter; for new-made honour doth forget men's **names**"
  - Shakespeare, *King John*
- "Call things by their right **names**.... Glass of brandy and water! That is the current but not the appropriate **name**: ask for a glass of liquid fire and distilled damnation."
  - Robert Hall, *Gregory's Life of Hall*
- "And last of all an Admiral came,

  A terrible man with a terrible **name**, --

  A **name** which you all know by sight very well,

  But which no one can speak, and no one can spell."
  - Robert Southey, *The March to Moscow, Stanza 8*
- The Borg (from Star Trek, not Redmond) seems to have the most scaleable naming system ever devised….

Outline

- **Introduction to Naming (9.1)**
- Name Services and the Domain Name System (9.2)
- Directory and Discovery Services (9.3)
- Case study: Global Name Service (9.4)

Overview of Naming

- Two reasons to name things
  - Make them human readable
  - Support late binding to resources or services
- Lots of things (mostly resources) are named in a distributed system
  - Computers
  - Services
  - Remote objects
  - Remote files
  - Users
- Names facilitate communication and resource sharing
- Users can't communicate with one another in a DS unless they can name one another
  - Email address
- <u>Identifier</u>: name interpreted only by a program
  - Example: object reference, NSF file handle
  - Identifiers chosen for their efficiency of lookup and storage

Pure and Impure (non-pure) Names

- Pure name: a bit pattern used for an identifier
  - Only direct use is for comparing against another identifier
- Impure (non-pure) name: one which is not pure, e.g.
  - /net/ted/bakken
  - joe@foo.bar.com
  - Impure names carry commitments (structure above…)
  - Pure names are attractive because they commit you to nothing
- Recall purpose of names was to identify values
- In practice, lookup tables (or directories) are often replicated
  - System robustness
  - Ease of access (can't find one nearby; phone books good example…)
- But replication creates complications….
  - What directory to use? Not always obvious….
  - E.g., "Arnold Q. Snailwright, 13 Meadow Lane" – What directory to find it in?
- Ergo, pure names not that good in a distributed system
  - Have to be looked up to be of any user
  - But where to look one up? Root servers…

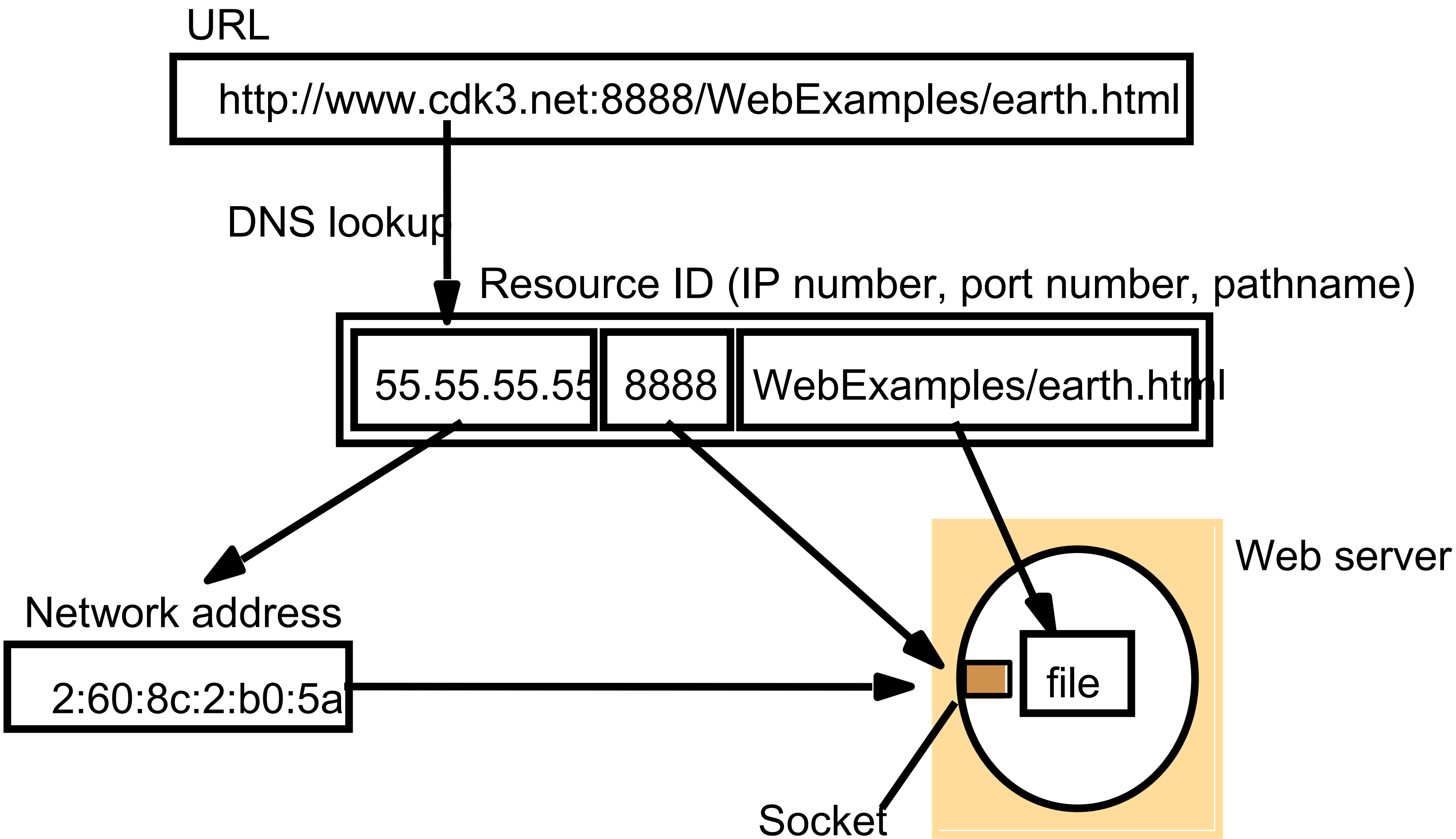
Names, Addresses, and Attributes

- A name is <u>resolved</u> when it is translated into data about the named item
  - Note: book uses the term “object” here, I prefer “item”
- Associating a name and value is called <u>binding</u>
- Names are generally bound to <u>attributes</u> of the named item, instead of the items themselves
- <u>Attribute</u>: value of a property associated with an item
- Most common attributed: address of an item
- Example 1: Domain Name Service attributes of a computer name
  - IP Address
  - Type of entry (mail server or normal host)
  - Length of time entry is valid
- Example 2: X.500 directory service example attributes for a person
  - Email address
  - Telephone number
- Example 3: CORBA Naming service: maps name onto object reference
- Example 4: CORBA Trading service: maps name onto object reference plus arbitrary number of other attributes

Names, Addresses, and Attributes cont.

- An ‘address’ can often be just another name that must be looked up
- Or the name may contain another name to be looked up
  - E.g., IP Address must be looked up to obtain a network address (ethernet)
  - Web browsers and email clients use the DNS to interpret names embedded in URLs or email addresses
- Figure 9.1 follows
  - Note the URL could have been looked up from some higher level service…

Figure 9.1: Composed naming domains used to access a resource from a URL



Names and Services

- Two categories of names…

1. Names names used in a distributed system are specific to some particular service
   - Client passes that along when requesting something
   - E.g., filename
   - E.g., process ID
2. Names valid beyond the scope of a single service (often globally valid)
   - User names
   - Email addresses
   - Computer names
   - Service names
   - Note: all of these names must be readable to and meaningful to humans
     - **Q: Why?**

Uniform Resource Locators (URLs)

- URLs are a kind of Uniform Resource Identifier (URI)
- URLs have some key properties
  - Scale to an unlimited set of web resources
  - Efficient handles for resources
- Disadvantage of URLs
  - Basically a lot like an address, and suffer from the same disadvantages
  - Resource deleted or moved: dangling link
- Other type of URI: Uniform Resource Name (URN)
- Goals
  - Solve dangling link problem
  - Provide richer modes of finding resources on the web
- Idea: have a URN that persists, even if resource/item moves
- Owner
  - Registers name and current URL
  - Registers the new URL if it is moved

Naming: © 2000 David E. Bakken

URNs (cont.)

- URN syntax: urn:nameSpace:nameSpace-specificName
- URN examples
  - urn:ISBN:0-201-62433-8
  - urn:foo.bar.edu:TR-2000-58
  - (any other examples from the real world???)
- Uniform Resource Characteristics (URCs): subset of URNs
- URC is description of a Web resource consisting of attributes of the resource
  - ‘author=Leslie Lamport’
  - ‘keywords=name,rose,Shakespeare’
- URCs are for
  - Describing web resources
  - Looking up web resources that match their attribute specification

Replication and Consistency in Naming

- Distributed database semantics: when an update occurs, the next queries get the latest information (strong semantics)
- But nothing for free …. One of two costs must be paid
  – Sometimes an update cannot be done because can’t contact enough replicas
  – Sometimes can’t do a read because an update is not yet stable or not enough replicas can be contacted
- Naming system requirements
  – Accessability (availability) deemed much more important than consistency
  – I.e., its more important to get an answer than to be guaranteed to (eventually) get the absolute last one
- Underlying assumptions
  1. Naming data do not change fast, so inconsistencies rare
  2. If you get an obsolete/inconsistent name and try to use it, it won’t work
  3. Even if it does somehow work, it won’t hurt anything
- Q: Are these assumptions true? Examples?

Outline

- Introduction to Naming (9.1)
- **Name Services and the Domain Name System (9.2)**
- Directory and Discovery Services (9.3)
- Case study: Global Name Service (9.4)

Name Services

- A name service stores a collection of one or more naming contexts
- Naming context: set of bindings between textual names of items and their names attributes
- Operations of a name service
  - Resolve a name (most important one)
  - Create new bindings
  - Delete bindings
  - List all bound names
  - Delete contexts
- Name management is separate from other services in a DS, because
  - Resources managed by different services can use the same naming scheme
    - **Examples?**
  - Can't always predict scope of sharing … may need to share (and thus name) items created in different administrative domains

General Name Service Requirements

- Name services originally very simple (single domain for one LAN)
- Global Name Service (DEC, 1986) first major naming service; goals
  - Handle arbitrary number of names and serve an arbitrary number of administrative organizations
  - Long lifetime: handle lots of changes in structure
  - High availability
  - Fault Isolation (contain local failures)
  - Tolerance of mistrust
- Other examples
  - Globe name service (Vjrie University, late 90s)
  - Internet Doman Name System (DNS)

Name Spaces

- Name space: collection of all valid names recognized by a particular service
  - Valid means the service will try to look it up; not that it is bound for sure
  - Name spaces require a syntactic definition
- Internal structures of names: one of
  - Hierarchical namespace: Unix files /etc/hosts
  - Organizational namespace: Internet DNS: eecs.wsu.edu
  - Flat set of letters and numbers
- Advantages of hierarchical names
  - Each part of the name resolves to a separate context
  - Same name may be used with different meanings in different contexts
  - Potentially infinite namespace: can grow indefinitely
  - Different contexts can be managed by different people

Aliases and Naming Domains

- Aliases
  - Similar to a Unix symbolic link
  - Allows a convenient name to be substituted for a more complicated one
  - DNS allows aliases: one name stands for another name
- Main reason for aliases: location transparency
  - Standardized name (within an org) for mail or ftp or other servers
  - E.g., mail.eecs.wsu.edu instead of thalia.eecs.wsu.edu
- Naming domain: name space for which there exists a single overall administrative authority for assigning names within it
  - The authority can delegate some parts of it to others (subdomains)

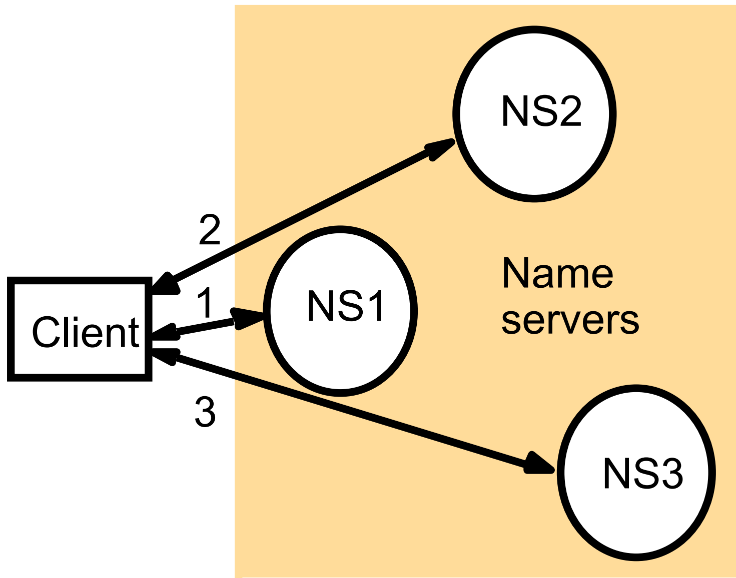
Customization of Name Spaces

- Namespaces can be customized in a lot of ways…
- File system mounting: lets users import files stored on servers with a local name on their server
  - Yet the local names on one server can still be managed autonomously from the shared fileservers
  - A shared file may be accessed from different names in different namepaces
  - Same name on different services/hosts can refer to different items
  - Example 1: file “/etc/passwd” on different hosts
  - Example 2: /bin/netscape bound to /bin/Linux/netscape or /bin/W2K/netscape

Name Resolution

- Resolution: iterative process of presenting a name to naming contexts
- Naming context choices
  - Map a given name onto attributes directly
  - Maps it onto a further naming context
- Partitioning of the data (namespace) means that a local name server cannot resolve all names without help from other name servers
  - Example: eecs.wsu.edu cannot provide IP address for foo.bar.com
- Navigation: process of locating naming data from multiple name servers to resolve a name
- Navigation design choices
  - DNS support iterative navigation model (Figure 9.2)
  - Multicast navigation: broadcast to all name servers, the one with the named attributes replies
  - Recursive navigation (Figure 9.3)

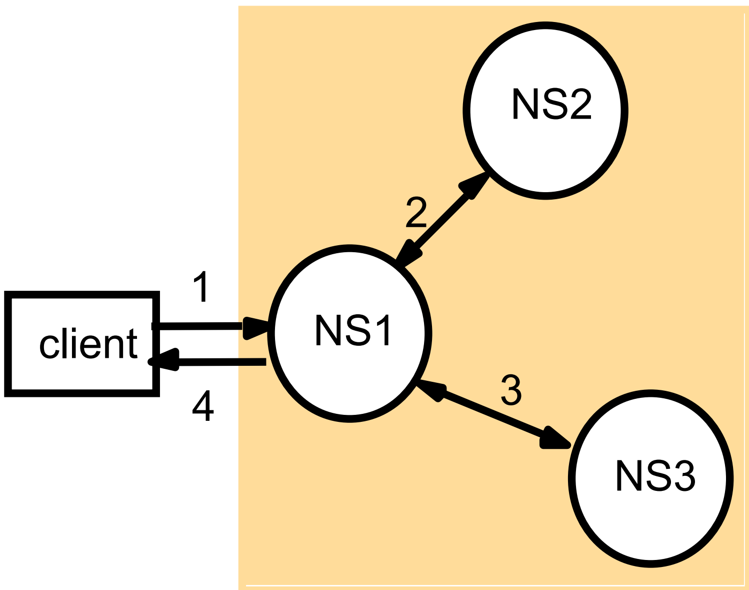
Figure 9.2
Iterative navigation



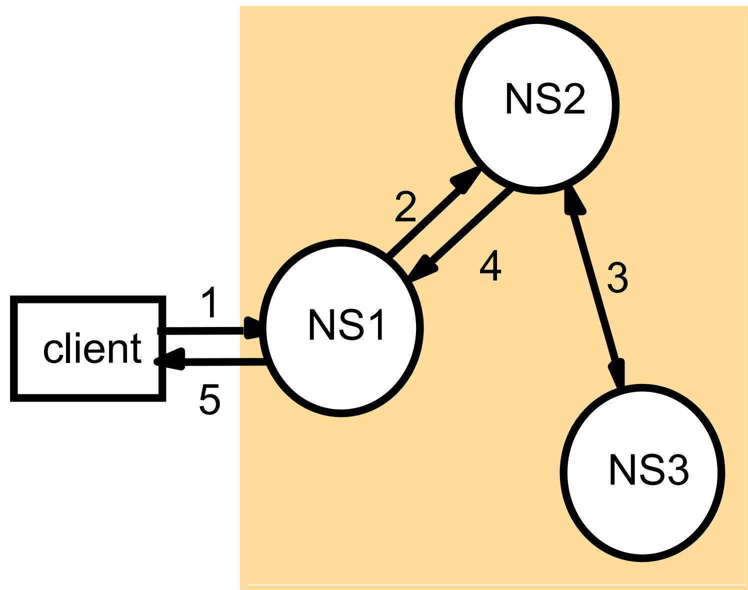
A client iteratively contacts name servers NS1–NS3 in order to resolve a name

Naming: © 2000 David E. Bakken

Figure 9.3: Non-recursive and recursive server-controlled navigation



Non-recursive
server-controlled



Recursive
server-controlled

A name server NS1 communicates with other name servers on behalf of a client

Caching

- Client name resolution software maintains a cache of results from previous resolutions
- Caching is key for performance and availability
- Caching only really works because names change rarely

Domain Name System (DNS)

- Original Internet naming setup (till circa 1985): centralized master file downloaded to all computers that needed them
- Problems
  - Scalability
  - Did not allow any local autonomy and administration
  - Could only be used for names of computer addresses, not other things
- Domain Name System replaced this
- Items of interest are mainly computers
  - IP addresses stored as attributes
  - Could store any other kind of item with any kind of attribute with DNS, but not often done
- Millions of names bound by Internet DNS, resolvable from any client
- Scalability achieved by
  - Hierarchical partitioning of the name database
  - Replication of the naming data
  - Caching results from name resolution queries

Domain Names

- Internet DNS partitioned by
  - Organization
  - Geography
- Top-level organizational domains
  - .com
  - .edu
  - .gov
  - .mil
  - .net
  - .org
  - .tv (new)
  - …. (soon probably .sex, others)
- Top-level geographical (country) domains
  - .us
  - .uk,
  - .jp
  - ….
  - Geographic names are not always in the country….

Kinds of DNS Queries

- Host name resolution: resolve a hostname into its IP address
  - nif-c1.eecs.wsu.edu to 134.121.64.1
  - Can use program **`nslookup`** on many OSs (Unix, Linux, …)
- Mail host location query:
  - Example: need to send mail to joe@foo.bar.com
  - DNS query to resolve foo.bar.com with type designation 'mail'
  - Returns a list of domain names of hosts that can accept mail for foo.bar.com
  - Returns a preference (integer) for each host to tell client preferred order
  - Can optionally return the IP addresses too
- Reverse resolution: give the domain name for an IP address
- Host information: DNS can store host info: architecture, OS, …
  - This can be a security hazard …. some suggest this not be implemented
- Well-known services: returns
  - List of services run by a computer: telnet, ftp, ….
  - Protocol used to access them (UDP, TCP)

DNS Name Servers

- Recall DNS scalability achieved by
  - Hierarchical partitioning of the name database
  - Replication of the naming data
  - Caching results from name resolution queries
- DNS database is distributed across a logical network of servers
  - Each server holds part of the naming database
  - Locality common: most queries are for local computers
  - Each server also records domain names and addresses of other name servers to help satisfy non-local queries
- DNS naming data are divided into zones, with
  - Attribute data for names in the direct domain (not sub-domains)
  - Names and addresses for at least two name servers that provide <u>authoritative</u> data for the zone
  - Names of name servers that hold authoritative data for delegated sub-domains, and their IP addresses
  - Zone management parameters governing caching and replication of zone data, etc.

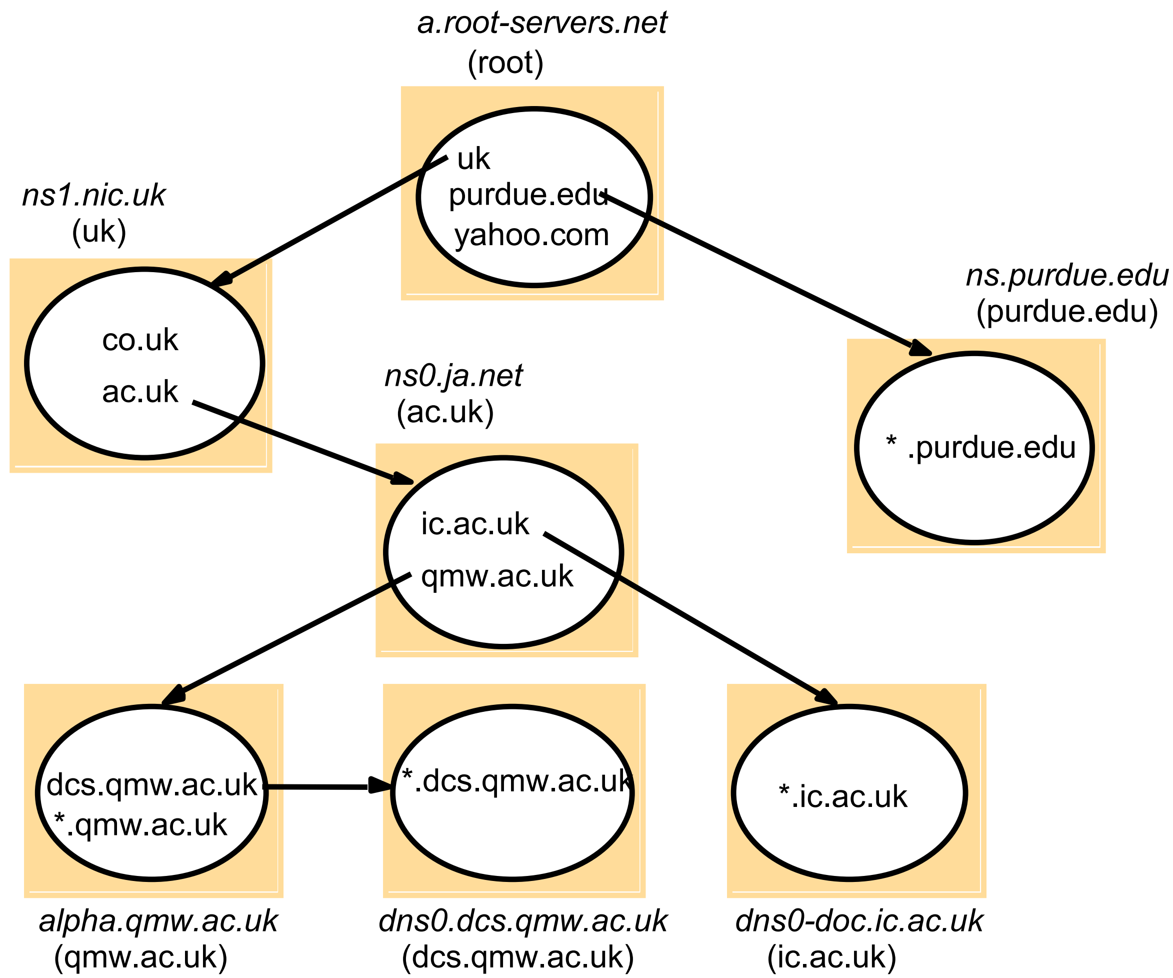
DNS Name Servers (cont.)

- A server may hold authoritative data for zero or more zones
- Each zone must be replicated authoritatively on at least two servers
- System administrators enter data for a zone into a master file
  - Serves as the source of authoritative data from the zone
- Two kinds of servers that provide authoritative data
  - <u>Primary server</u> (a.k.a. <u>master server</u>) reads zone data directly from local master file
  - <u>Secondary server</u> downloads zone data from a primary server
- Any server may cache data from other servers
  - Clients given the cached data must be told it is non-authoritative
  - Each entry in zone has time-to-live value to invalidate cached data eventually

Figure 9.4
DNS name servers

Note: Name server names are in italics, and the corresponding domains are in parentheses.

Arrows denote name server entries



Navigation and Query Processing

- DNS client is called a resolver
  - Usually implemented as library software
- Actions of the resolver
  - Accepts queries
  - Formats them into messages of legal DNS syntax
  - Communicates with one or more name servers
    - Uses simple request-reply protocol with UDP and well-known port
  - Times out and resends query if needed
- DNS architecture allows for both recursive and iterative navigation
  - Resolver specifies which kind of navigation required when contacting name server
  - Name servers are not bound to implement recursive navigation: ties up threads

Discussion of DNS

- It works pretty well!
- Gives pretty short response times for lookups, given the scale
- DNS allows naming data to become inconsistent
  - Contained in caches, secondary servers
  - Does not hurt anything until its used
  - DNS does not address how staleness is detected
- DNS database represents pretty much the lowest common denominator of what would be considered useful by Internet communities
- DNS limitations
  - Rigidity w.r.t. changes in the structure of the name space
  - Lack of ability to customize the name space to suit local needs
  - Both overcome in research system global name service (Sec 9.4)

Outline

- Introduction to Naming (9.1)
- Name Services and the Domain Name System (9.2)
- **Directory and Discovery Services (9.3)**
- Case study: Global Name Service (9.4)

Directory and Discovery Services

- Name servers
  - Store collections of <name, attribute>
  - Attributes generally looked up from a name
  - Dual is obvious: looking up a name from an attribute
  - Nice: sometimes don't know name of what you want, but know attributes
- Example user queries
  - "What is the name of the person with phone 509-335-2399?"
  - "What are names of printers, in my building, that can print PostScript, that can print in color, that has hi-res and lots of memory? How busy and available are they now?"
- Directory service: service that lets you look up names from attributes
  - Examples: Microsoft Active Directory, X.500, LDAP
  - Also called Examples: Microsoft Active Directory, X.500, LDAP
  - Also called yellow pages services
  - Traditional name services also called white pages services
  - Directory services also called attribute-based name services

Discovery Services

- Discovery service: directory service that registers services provided in a spontaneous networking envirnoment
  - Interface for servers: registering and de-registering services
  - Interface for clients: look up services they need
- Example: occasional visitor to a company or hotel needs to print a doc
  - User can't be expected to configure names of printers or to guess them
  - User looks up the printer it needs
    - Q: what conventions are involved here?
- Looking up a service may not involve a user
  - Refrigerator discovers and contacts error-logging service when having problems (TV commercial…)
- Context for discovery in a discovery service is called its scope
  - Often local network reachability defines the scope
  - Contrast with more general directory services: global scope

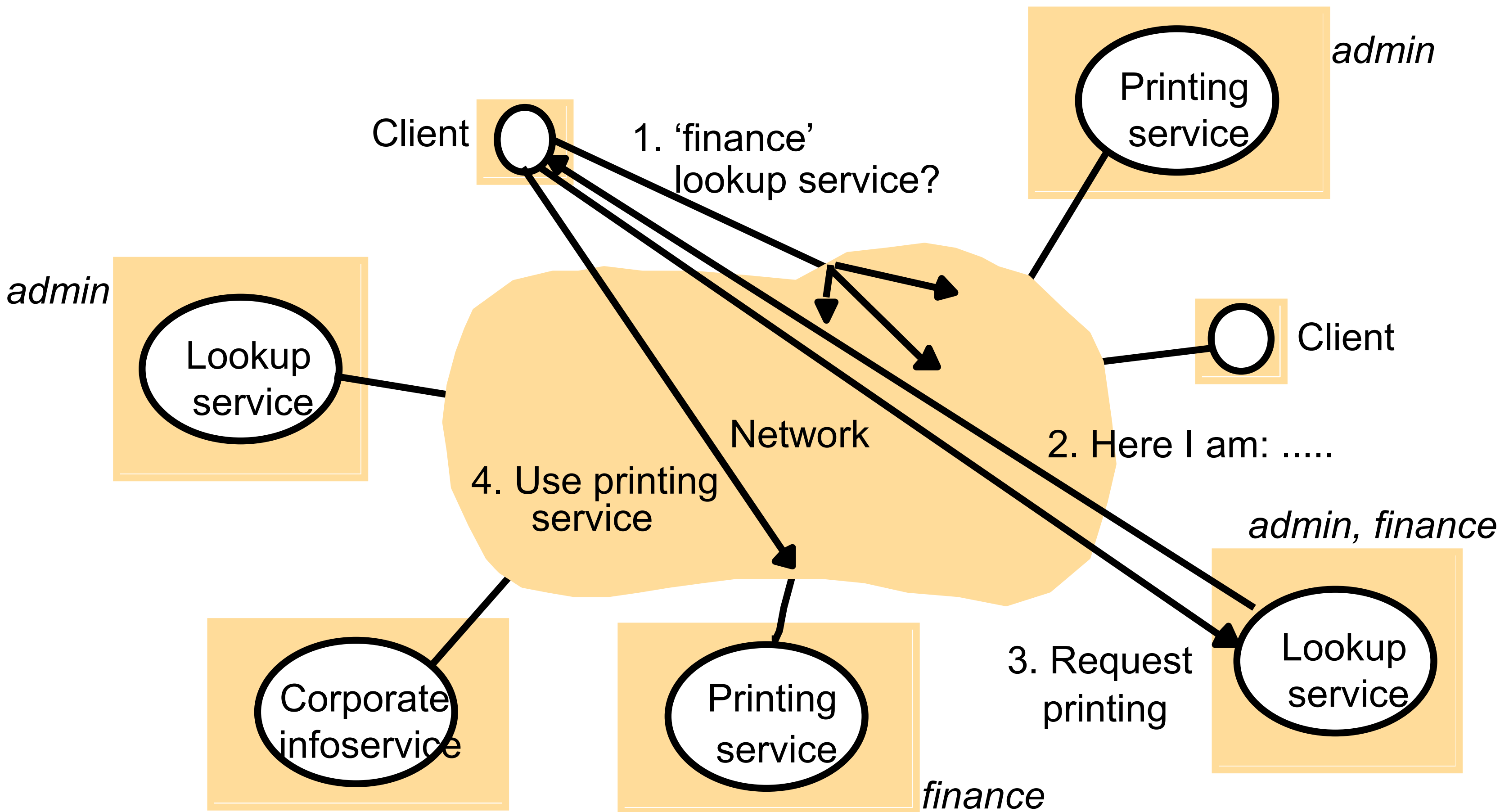
Jini

- Jini [Waldo, Arnold, and others] is Java-based
  - Assumes JVMs running on all computers
  - Can thus use RMI for the remote invocations
  - Can thus download code as necessary
- Jini provides
  - Service discovery
  - Ransactions
  - Shared dataspaces called JavaSpaces (like Linda's Tuple Spaces)
  - Events
  - (Only cover Jini's discovery service here….)
- Note: a Jini service may be registered with more than one lookup service

Jini Lookup, Bootstrapping, and Leases

- Lookup matching of service offers to client requests can be based on
  - Attributes (like any directory service)
  - Java types
    - E.g., request a color printer to which the client has the Java interface
- Bootstrapping: how can a client locate the lookup service?
  - Choice 1: know addresses of the lookup service ahead of time
  - Choice 2: multicast to well-known IP multicast address
  - Choice 3: lookup services can announce their existence to same multicast address
    - Clients can subscribe to learn of new lookup services
- Jini uses leases (we've seen them before)
  - When services register, they are given minimum amount of time their entry will be valid
  - Must contact the lookup service before that time has passed, or they are assumed to have failed and the lookup service can delete the entry

Figure 9.6
Service discovery in Jini



Outline

- Introduction to Naming (9.1)
- Name Services and the Domain Name System (9.2)
- Directory and Discovery Services (9.3)
- **Case study: Global Name Service (9.4)**

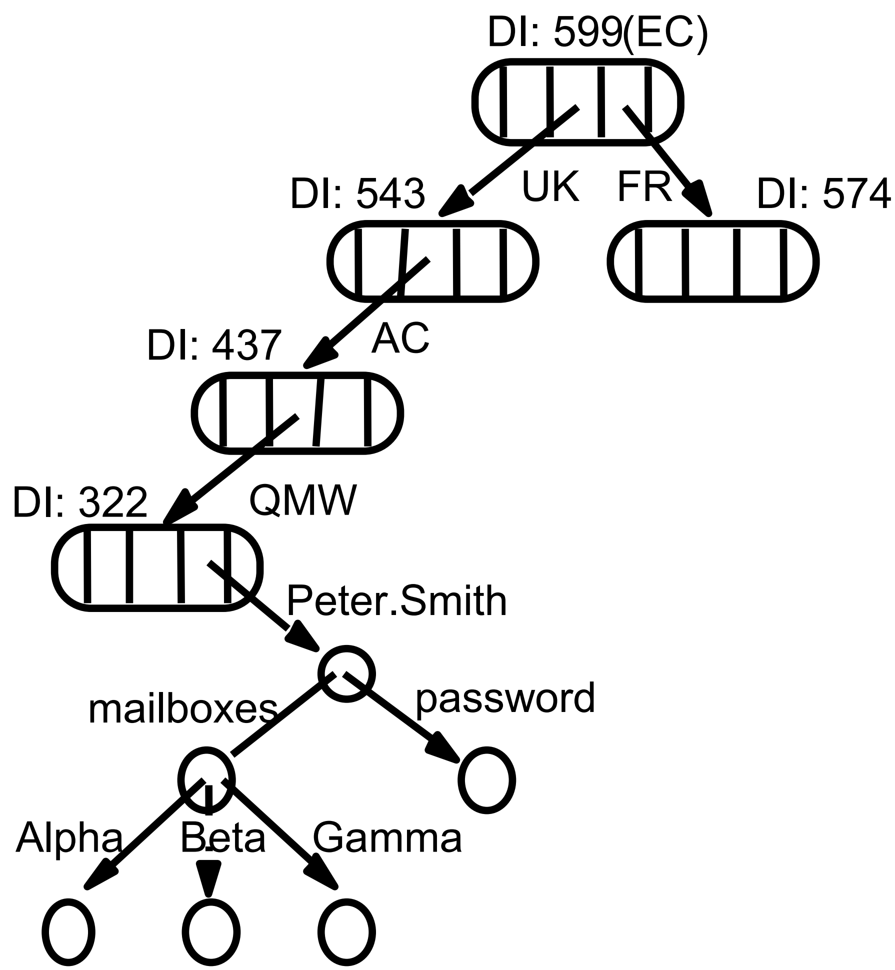
Case Study: Global Name Service

- Global Name Service (GNS) by Lampson et al at DEC Systems Research Center (1986)
- Goals
  - Must scale to millions of computers and billions of users
  - Long lifetime: must work well when
    - Grows from small to large
    - Network it uses evolves
  - Support change in structure of name space: reflect changes in org structures
  - Accommodate changes in names of individuals, organizations, and groups
  - Accommodate changes in naming structure: takeover, merger, …
    - Focus of Section 9.4

GNS Details

- Directories are named by multi-part pathnames, relative to
  - Root
  - A working directory (like Unix filenames0
- Each directory is assigned a unique directory identifier (DI): an integer
- Directory contains list of names and references
- Values stored at leaves are organized into value trees
  - Attributed thus can be structured values
- Names in GNS have two parts: **<directory name, value name>**
  - First identifies a directory
  - Second part identifies a value tree (or part of one)
- E.g., (Fig 9.7)
  - Attributes of a user Peter Smith stored in a value tree named **<ECUK/AC/QMW, Peter.Smith>**
- Directory tree is partitioned, and each partition replicated
  - Consistency maintained even with two or more updates (only one succeeds)

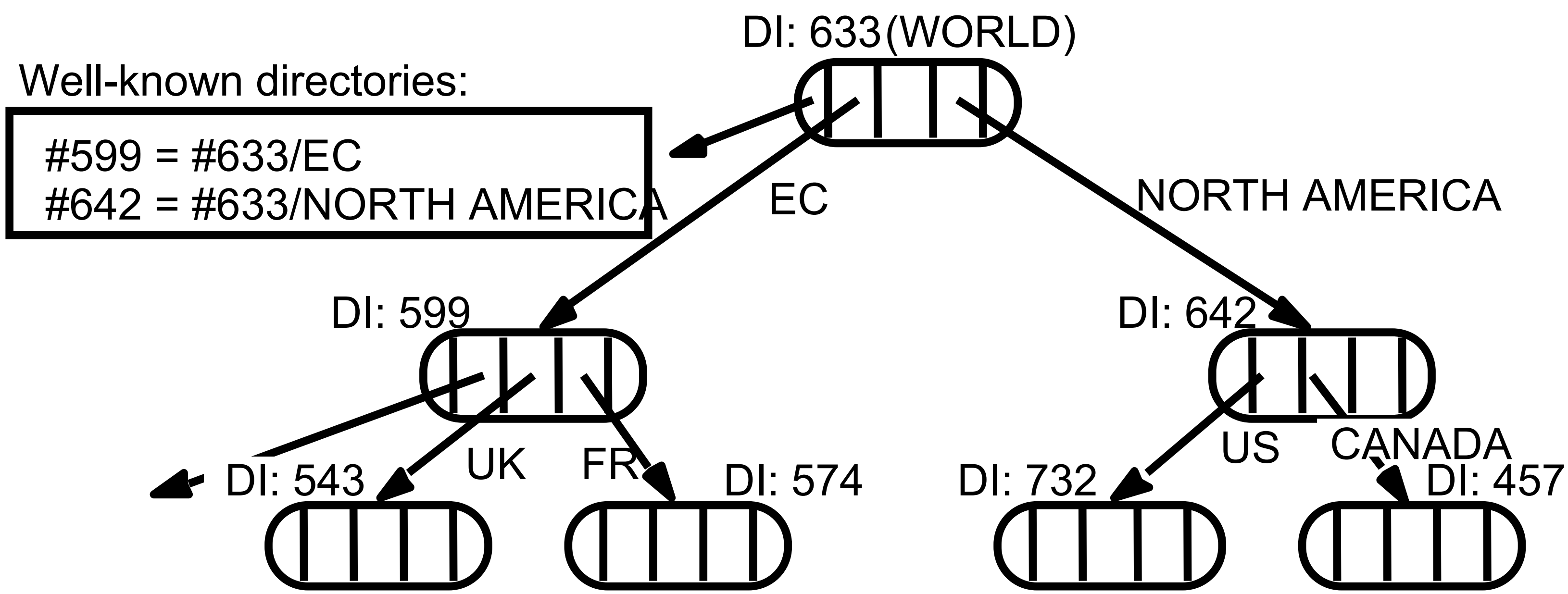
Figure 9.7: GNS directory tree and value tree for user Peter.Smith



Accommodating Change

- What happens if we need to change the hierarchy?
- E.g., need to merge two formerly-top-level directories “EC” and “North America” under “World”
- Easy to do, tree-wise, but how does it affect names still in use that use the old root?
  - <UK,AC,QMW, Peter.Smith>
- Solution uses uniqueness of directory identifiers
  - Working root maintained for each program’s environment (like $PWD etc.)
  - User agent (library code to do the query) knows working directory, and passes it on to the GNS server
- Implementation problem: in a big distributed database, how can GNS find a directory given only its identifier (#599)
- Solution: GNS tracks all directories used as working roots in a table of “well-known directories” in the real root

Figure 9.8
Merging trees under a new root



Restructuring Database for Org. Change

- Scenario: the US becomes part of the European Community (EC)
  - Reality check: only in the British authors' dreams….
- Can just move <North America/….> tree under <EC…>
  - But then old existing names break
- Solution: put a symbolic link in old entry, pointing to new directory

Figure 9.9
Restructuring the directory