Distributed Systems

Naming

Dongman Lee

Dept of CS

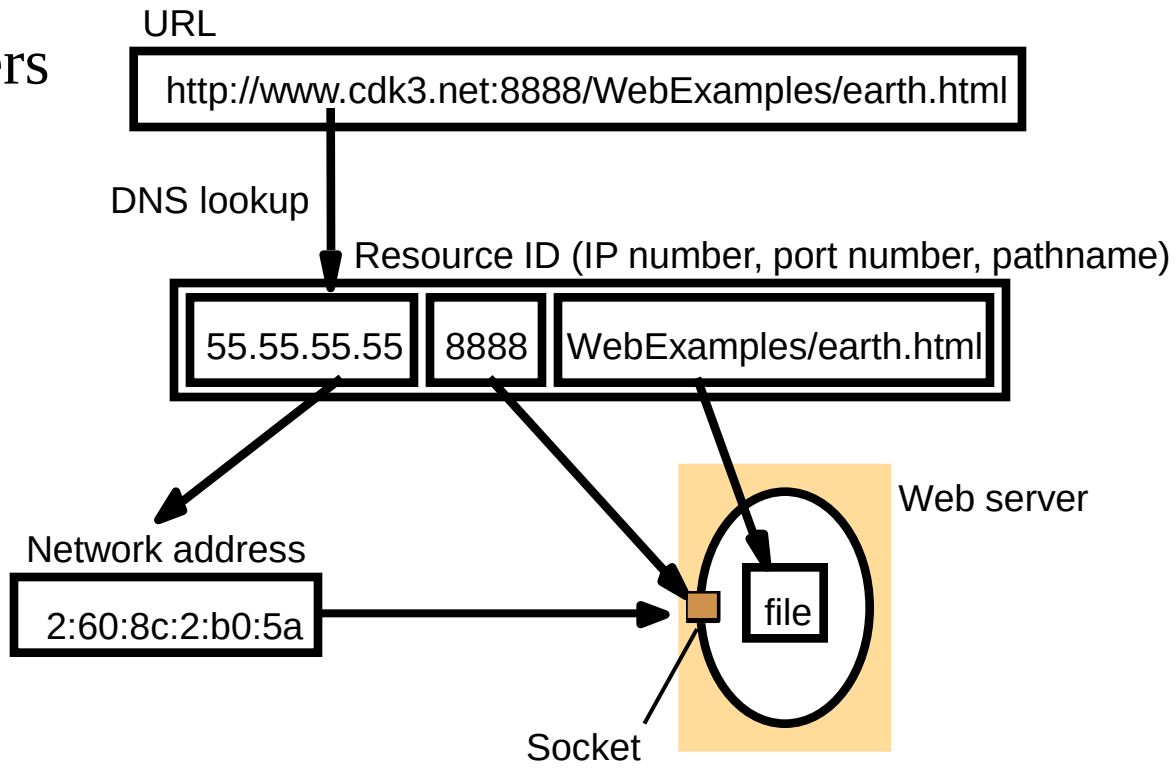
KAIST

Class Overview

- Why Naming?
- Naming Fundamentals
- Name Services
- Directory & Discovery Services
- Locating Mobile Entities
- Case Studies
  - Host naming: DNS
  - Content naming: Naming in ICN
  - Directory: X.500
  - Discovery: SLP

Names are

- Network addresses
- Process identifiers
- File names
- User names
- Service names
- Class identifiers
- …



Why Naming?

- Names are used to
  - facilitate sharing
    - a name is a common way for other objects to communicate with an object with that name
    - a single name may represent a set of multiple objects doing the same job
  - uniquely identify particular members of some large set
    - a name may specify roles played by the object(s) named
    - a name may give hints about the location of the object named
    - a name may indicate a set of operations allowed (authorized) by the object named

Naming in Distributed Systems

- Characteristics of naming in distributed systems
  - locations
    - are unintuitive
    - make object migration/mobility difficult
  - naming system could be distributed as well
    - its efficiency and scalability may depend on how distribution is done
- What we can learn from telephone networks [Birrel]
  - decentralized
  - changes are done by authorized entities
  - database is replicated
  - multiple names are allowed
  - names are context-dependent

Naming Fundamentals

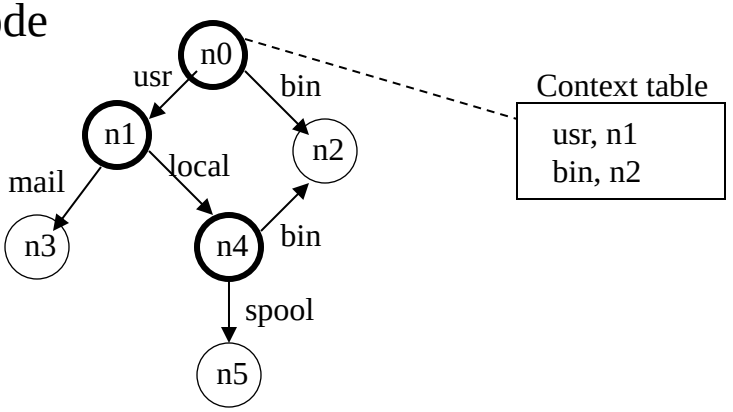
- Different types of names
  - addresses
    - name of an access point to an associated entity
    - an entity may be accessible at more than one address
    - an entity may change its address and the address is reassigned to other entity
  - identifiers
    - properties of identifier
      - an identifier refers to at most one entity
      - an entity is referred to by at most one identifier
      - an identifier always refers to the same entity (immutable)
  - human readable names
    - represented as character-string
    - used in many different forms
      - DNS
      - file names

Naming Fundamentals (cont.)

- Name spaces
  - a collection of all valid names recognized by a name service
  - naming domain
    - a name space for which there exists a single overall naming authority for managing names within it
    - e.g. DNS TLD
  - naming context
    - an abstraction that either maps a given name to attributes or further naming context and derived name
    - e.g. CWD in UNIX
  - naming convention
    - absolute vs. relative
    - flat vs. hierarchical

Naming Fundamentals (cont.)

- Name spaces (cont.)
  - represented as a labeled, directed graph
    - regular node
      - represents a named entity and has no outgoing edges
      - contains information on the entity (address or identifier) or content of an entity
    - context node
      - has a number of outgoing edges with labels
      - contains context table of (edge label, node id) pairs
  - path name
    - a sequence of labels refers to a node
    - absolute path name
    - relative path name



Naming Fundamentals (cont.)

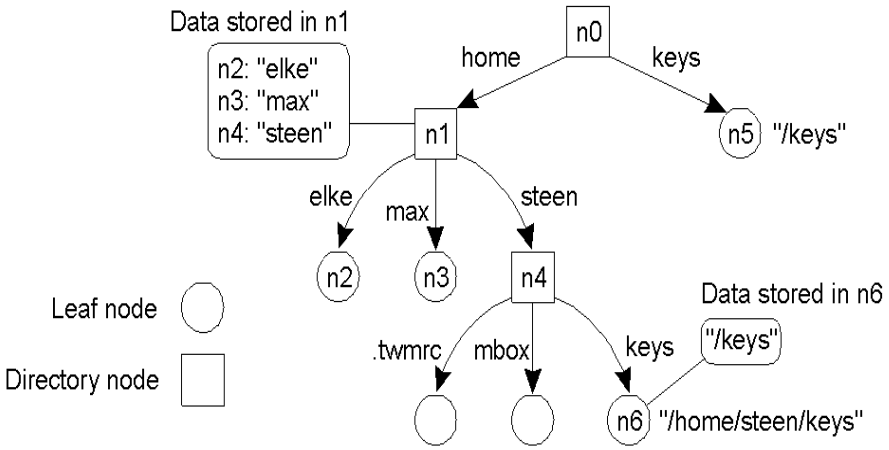
- Names and location
  - Non-pure names
    - names convey information on where to find the objects named
      - object location
      - name service location
  - Pure names
    - no interpretation of names themselves
    - pure names don’t convey location

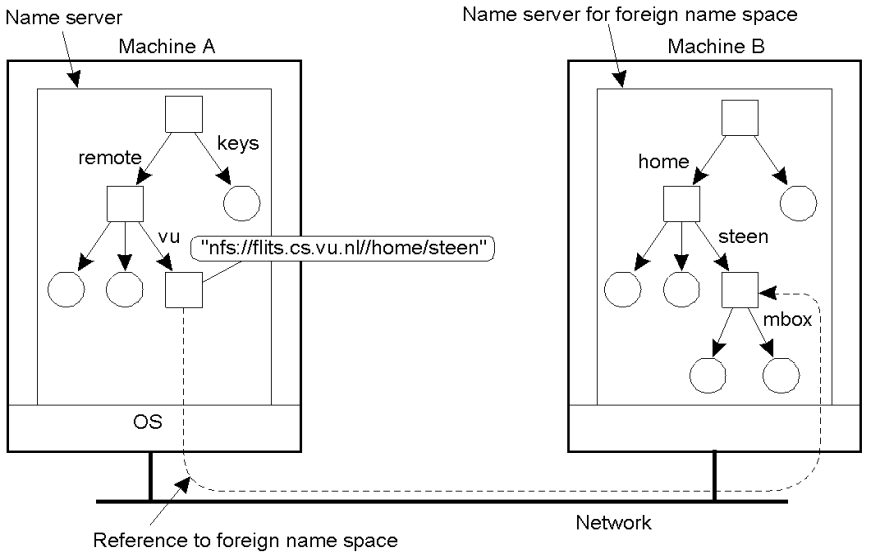
Naming Fundamentals (cont.)

- Name resolution
  - a process that looks up a name in the name database to obtain associated attributes
    - an object identifier or address
    - services provided by the object named
    - a server managing the object named
  - closure mechanism is required
    - knowing how and where to start name resolution
    - select the initial node in a name space to figure out where to start
      - system should provide a way of where to start
      - examples
        - » address of a root node of a file system
        - » pre-defined syntax in naming (e.g. phone numbers)
        - » name context hints (e.g. CWD, HOME)

Naming Fundamentals (cont.)

- Name resolution (cont.)
  - multiple levels of navigation
    - linking - aliases to another name contexts (e.g. symbolic or hard links)
    - mounting - merged name spaces (e.g. NFS or GNS)





Name Services

- Name service
  - a service that allows users and processes to add, remove, and look up names
  - main tasks
    - name space/database management
    - name resolution
- Desirable properties [Lampson]
  - scalable
  - long lifetime
  - high availability
  - fault isolation
  - tolerance of mistrust

Name Services (cont.)

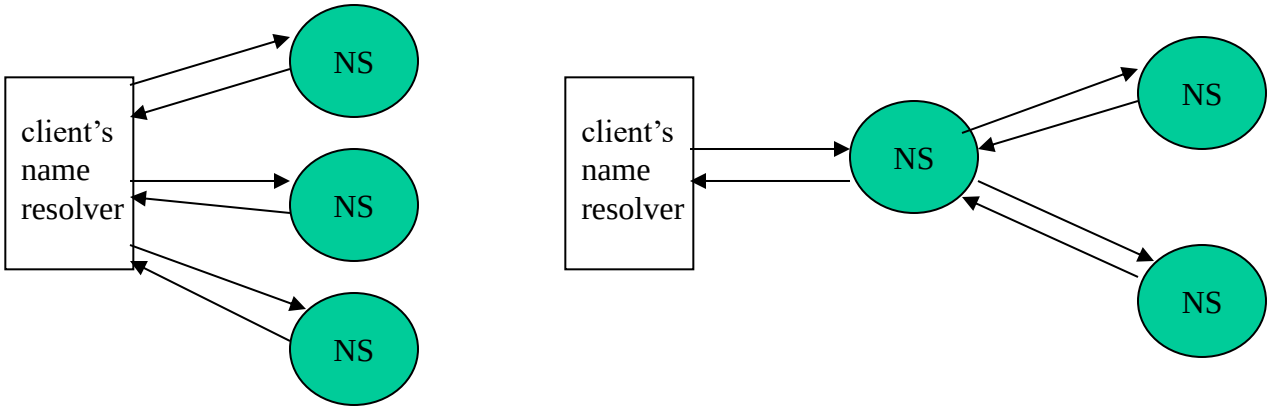
- Design considerations [Needham]
  - bindings
    - relax binding time since distributed systems are more dynamic
    - cache frequently used bindings if possible and update on use
  - name resolution
    - name servers are usually replicated for availability and reliability
    - pure names should be used carefully considering the size of networks, broadcasting capability, etc.
  - consistency
    - accessibility is considered more important than absolute correct result -> asynchronous consistency
  - scalability
    - never assume that the size of the name space is liable to get

Name Services (cont.)

- Name space management
  - name space can be partitioned into three levels [Cheriton/Mann]
    - global level
      - top level nodes in the name space
        - » context node encompasses many organizations
      - rare changes and large number of users
        - » cacheable in the lower levels - performance may be not relatively critical
        - » stability and high availability are important - replication
    - administrational level
      - context nodes representing group of nodes belonging to same organization in the name space
      - relatively stable but more frequent changes than global level nodes
        - » availability and performance requirements similar to global level
    - managerial level
      - context nodes for small group and regular nodes in the name space
      - frequent changes and high access rate
        - » lesser availability requirement but immediate response required

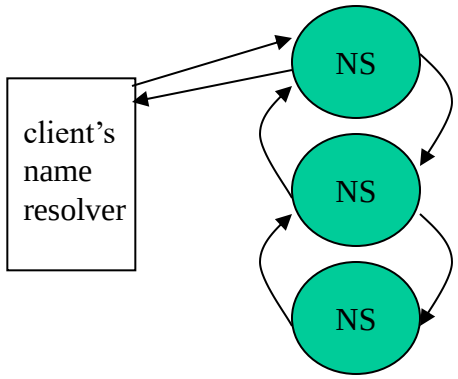
Name Services (cont.)

- Name resolution mechanisms
  - name resolver
    - agent that is responsible for ensuring name resolution process
  - iterative name resolution vs. recursive name resolution
  - Iterative name resolution
    - client's name resolver contacts as many name servers as it can until name resolution is done
    - each name server has partial information
    - intermediate server may play a role of name resolver



Name Services (cont.)

- Name resolution mechanisms (cont.)
  - recursive name resolution
    - name server passes the result to next name server as it can until name resolution is done
      - may put higher performance demand on each name server
    - allow each name server to gradually learn the address of other name servers responsible for part of names that it was not able to resolve
      - cached resolution results can be leveraged next time
    - may provide cheaper communication cost
      - administrational nodes are usually closer to their managerial nodes



Name Services (cont.)

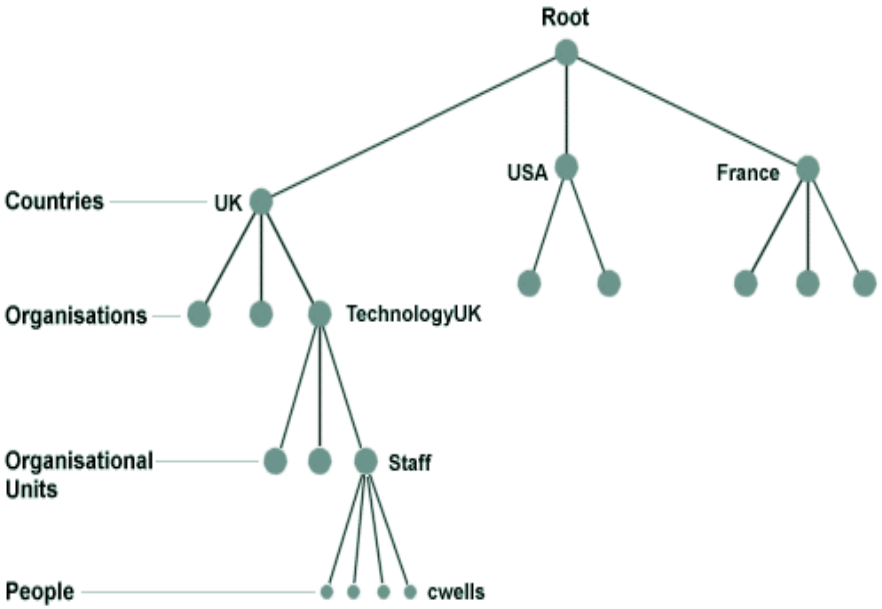
- Other name resolution mechanisms
  - multicast navigation
    - multicast name resolution request and corresponding server returns the result
  - integrated name service in V systems
    - name service is integrated with the object operation
      - server's address is returned with result of operation

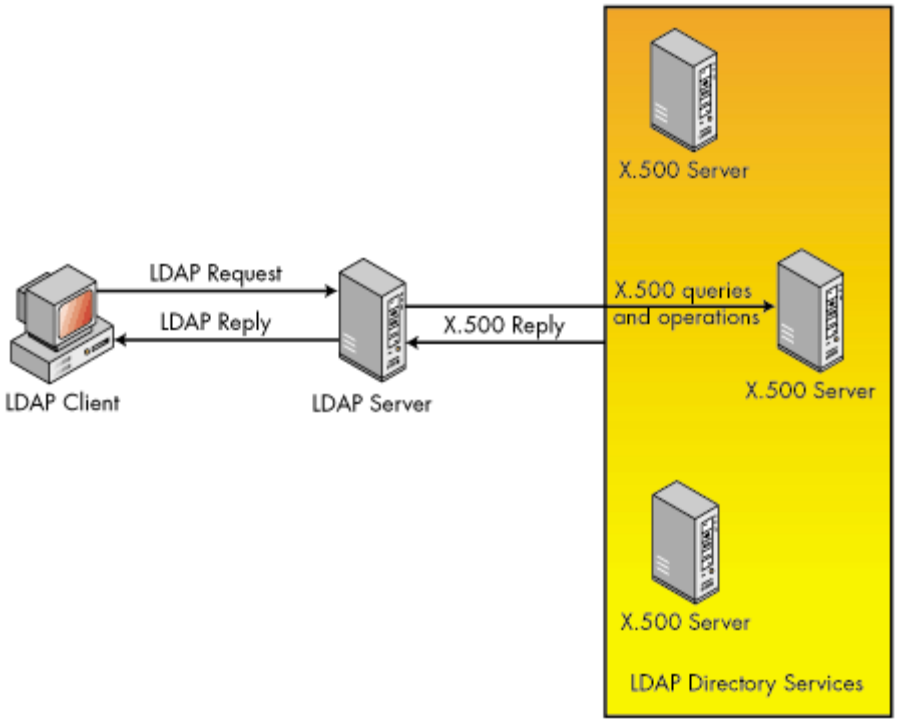
Directory and Discovery Services

- Directory service
  - store collections of bindings between names and attributes
  - look up entries that match *attribute-based* specifications
- Directory service vs. name service
  - yellow page vs. white page
- Advantages of attributed-based services
  - select objects according to precise attribute specifications where names might not be known
  - hide the structure of organizations to the outside
- Discovery service
  - look up service without prior binding and user intervention for an environment where clients and services dynamically change (for mobile networks)
  - services
    - an interface automatically registering and de-registering services
    - an interface for clients to look up the services that they require from those that are currently available

Directory and Discovery Services (cont.)

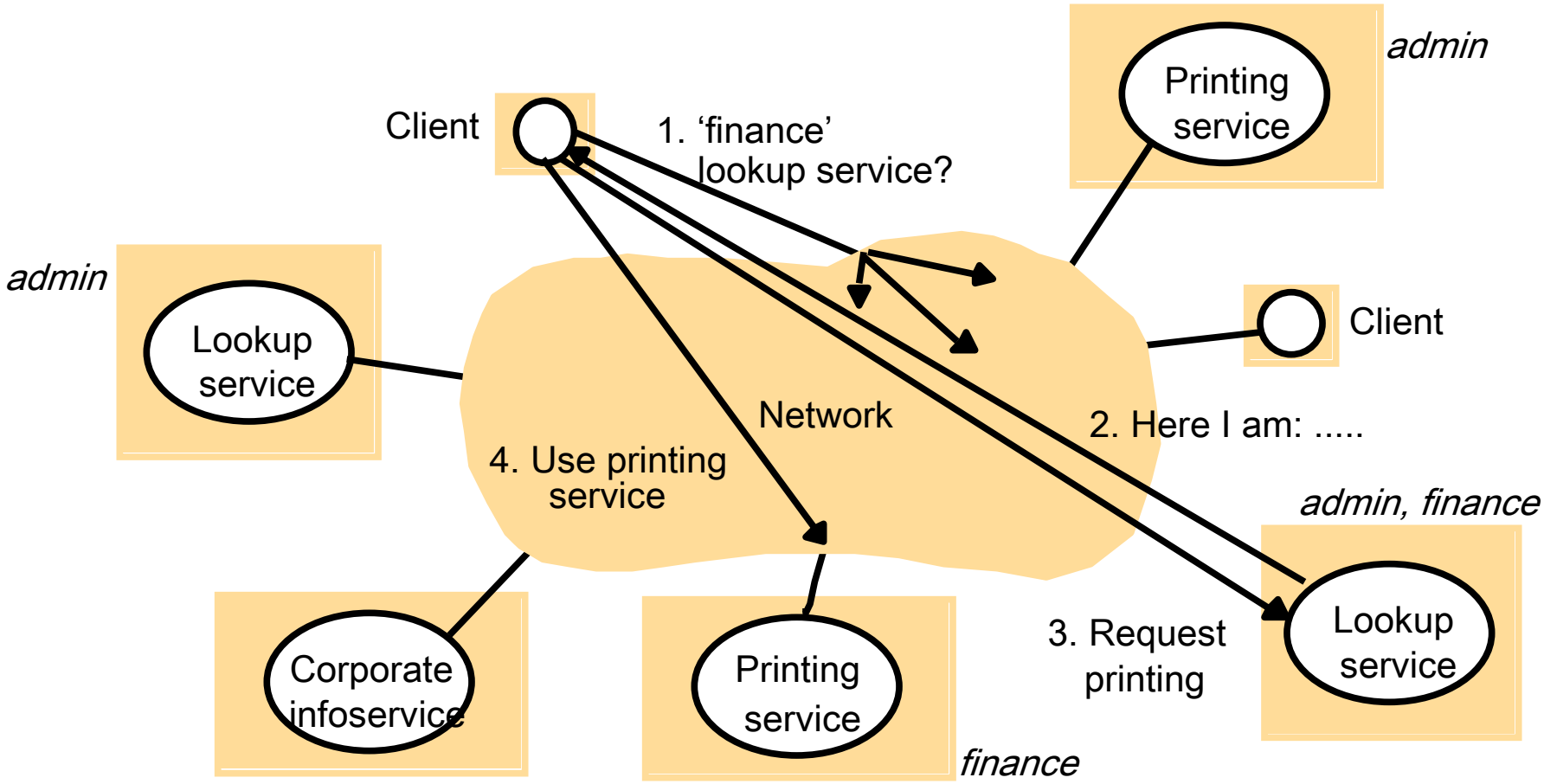
- X.500 Directory Service





Directory and Discovery Services (cont.)

- Service discovery in Jini

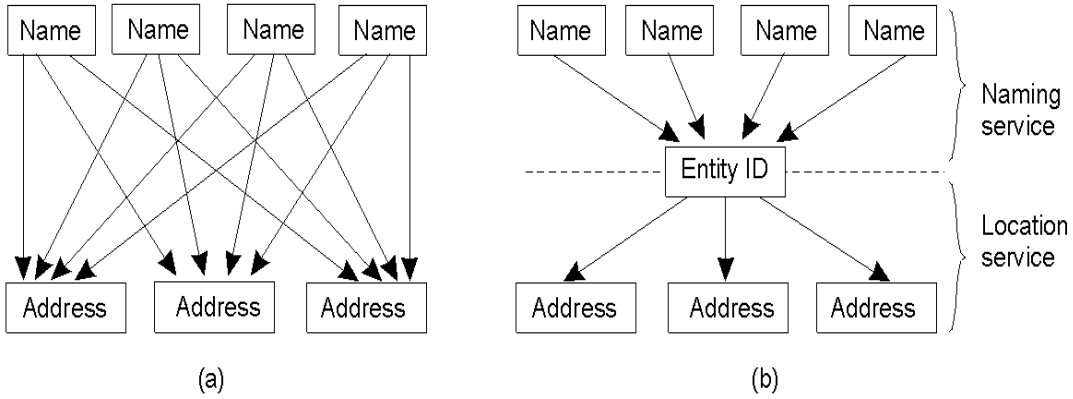


Directory and Discovery Services (cont.)

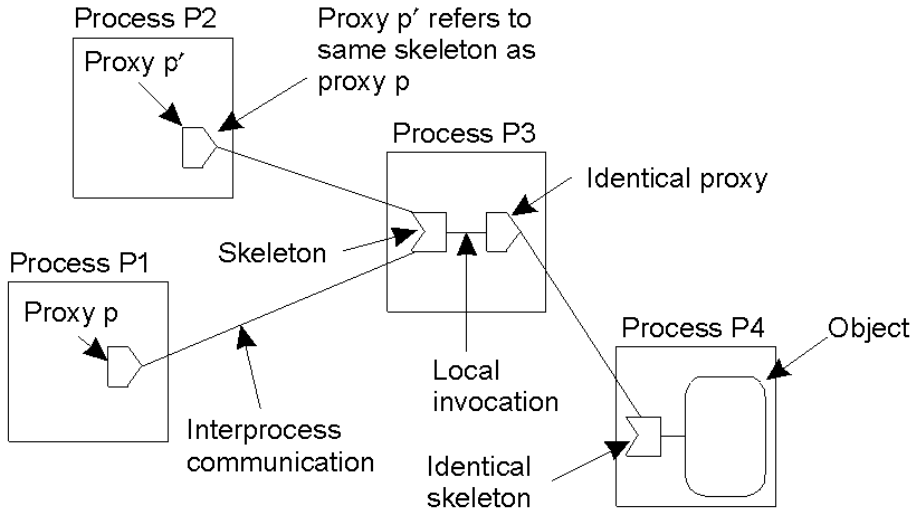
- Service discovery in ad-hoc environments
  - No central directory service
    - Jini, UPnP assume fairly stable network
    - Flooding? => designed for small network
  - Limited use of network/device resources
  - Distributed
    - Node acts as client + server
  - Service announcement
    - Flooding: How to reduce flooding traffic?
      - Periodically, delta announcement (Konark)
      - Slotted (DEAPSpace uses slotted+periodic)
    - Cache advertisements
    - TTL

Locating Mobile Entities

- Location service
  - separation between naming and locating entities via identifiers

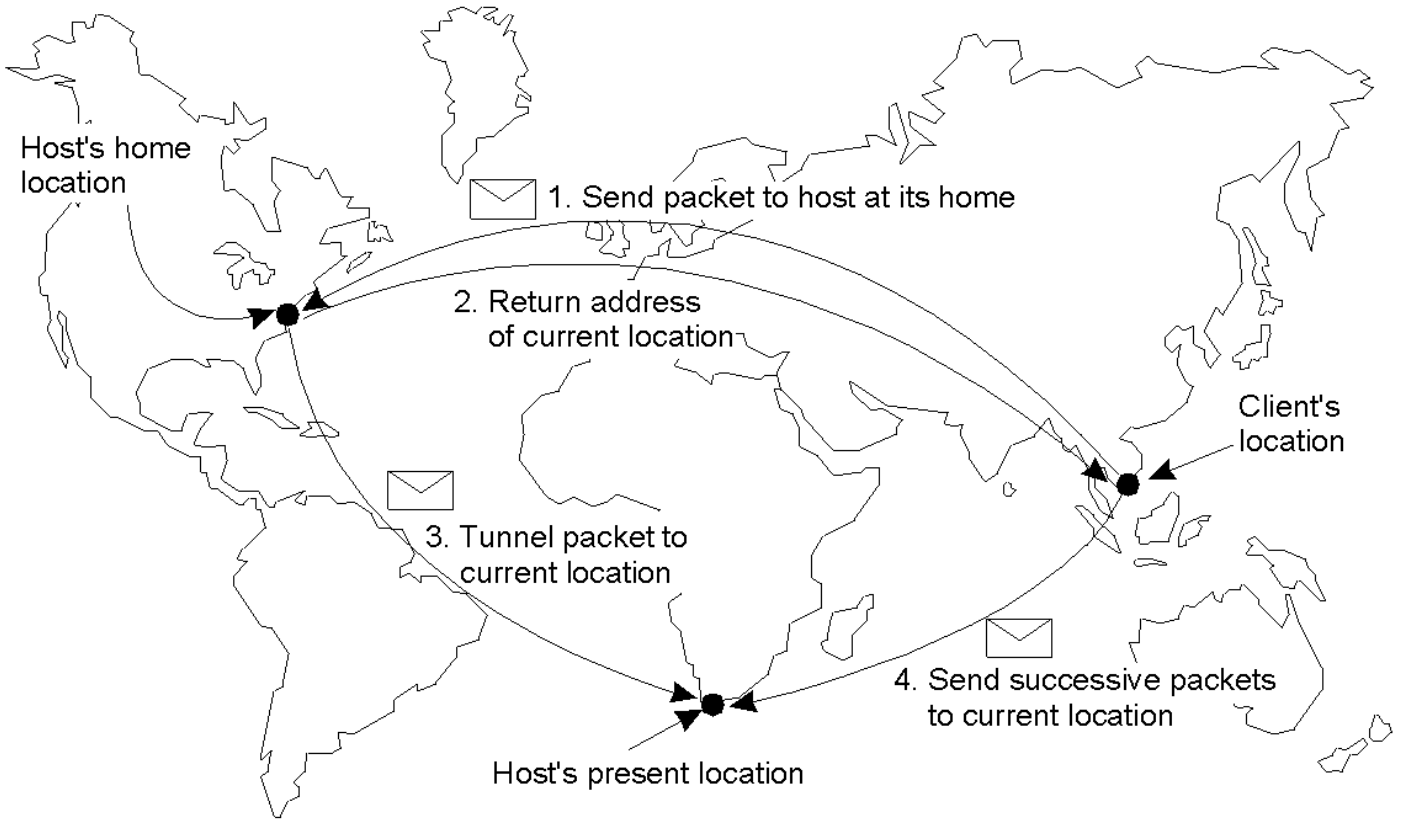


- Simple location services
  - multicasting/broadcasting
  - forwarding pointers



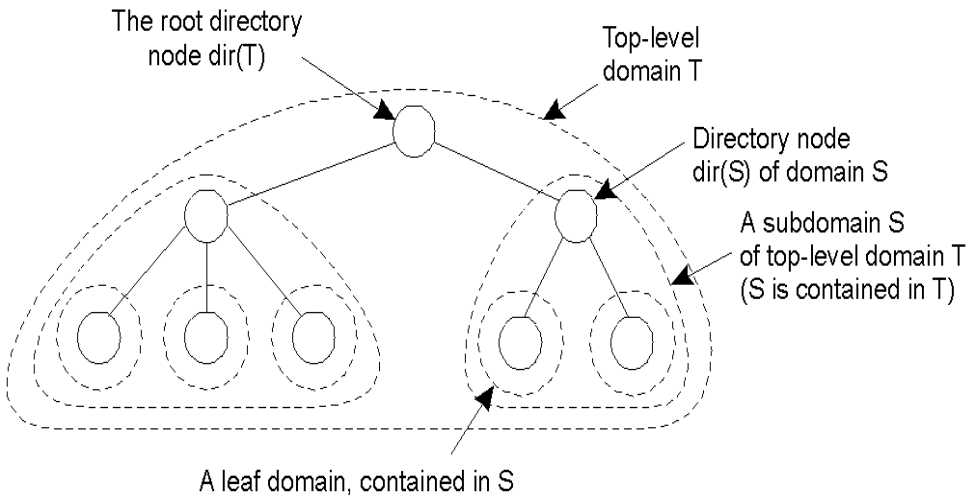
Locating Mobile Entities (cont.)

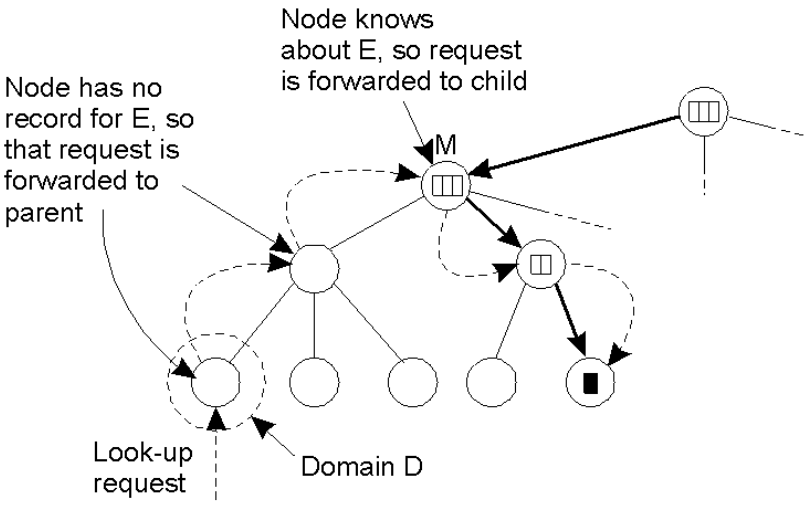
- Scalable solutions
  - home-based approach – similar to mobile IP



Locating Mobile Entities (cont.)

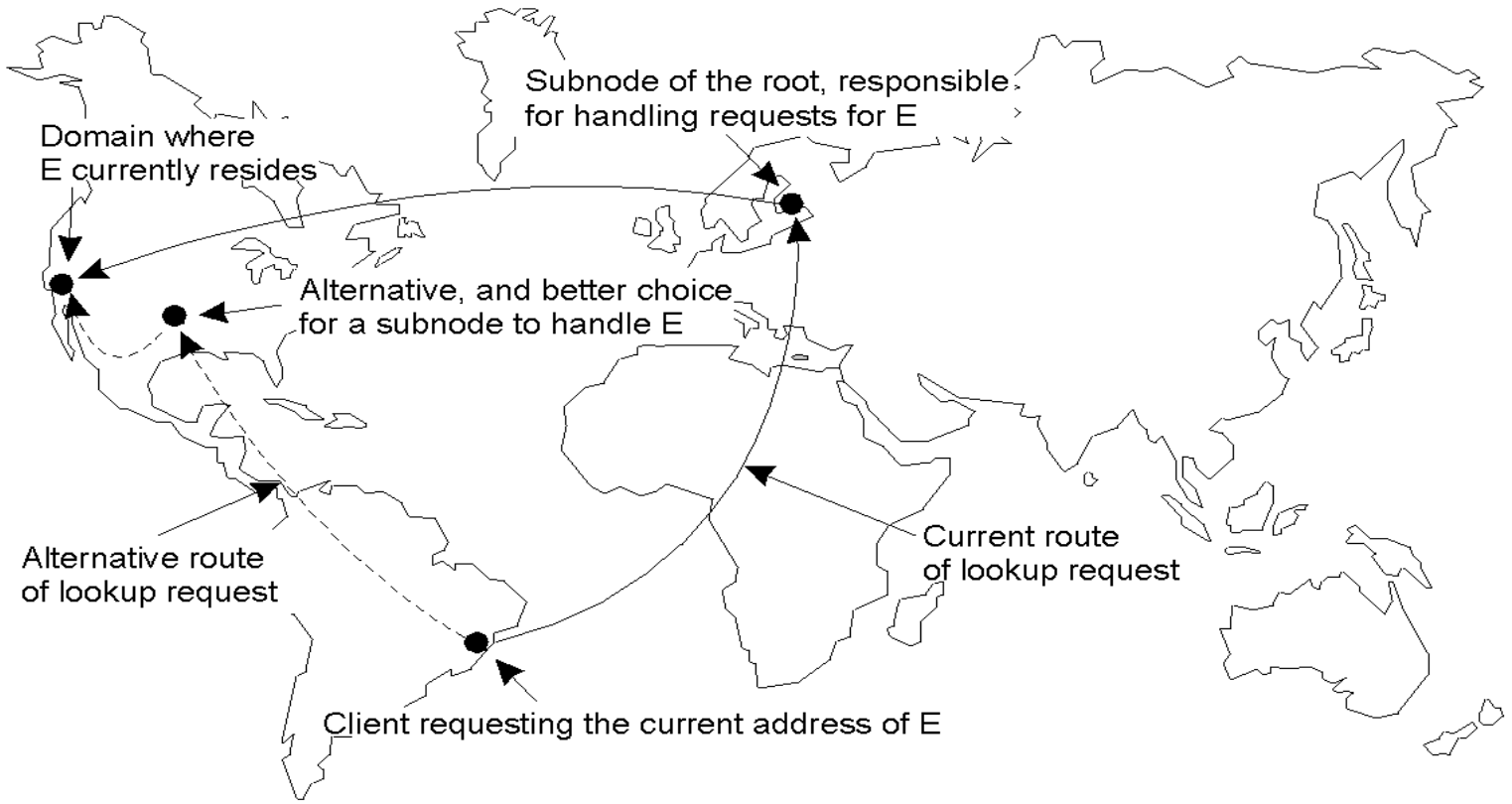
- Scalable solutions
  - hierarchical approach
    - hierarchical organization of a location service into domains, each having an associated directory node
    - pointer cache - caching a reference to a directory node of the lowest-level domain in which an entity will reside most of the time





Locating Mobile Entities (cont.)

- The scalability issues are related to uniformly placing the sub-nodes of a partitioned root node across the network covered by a location service.



Case Study: DNS

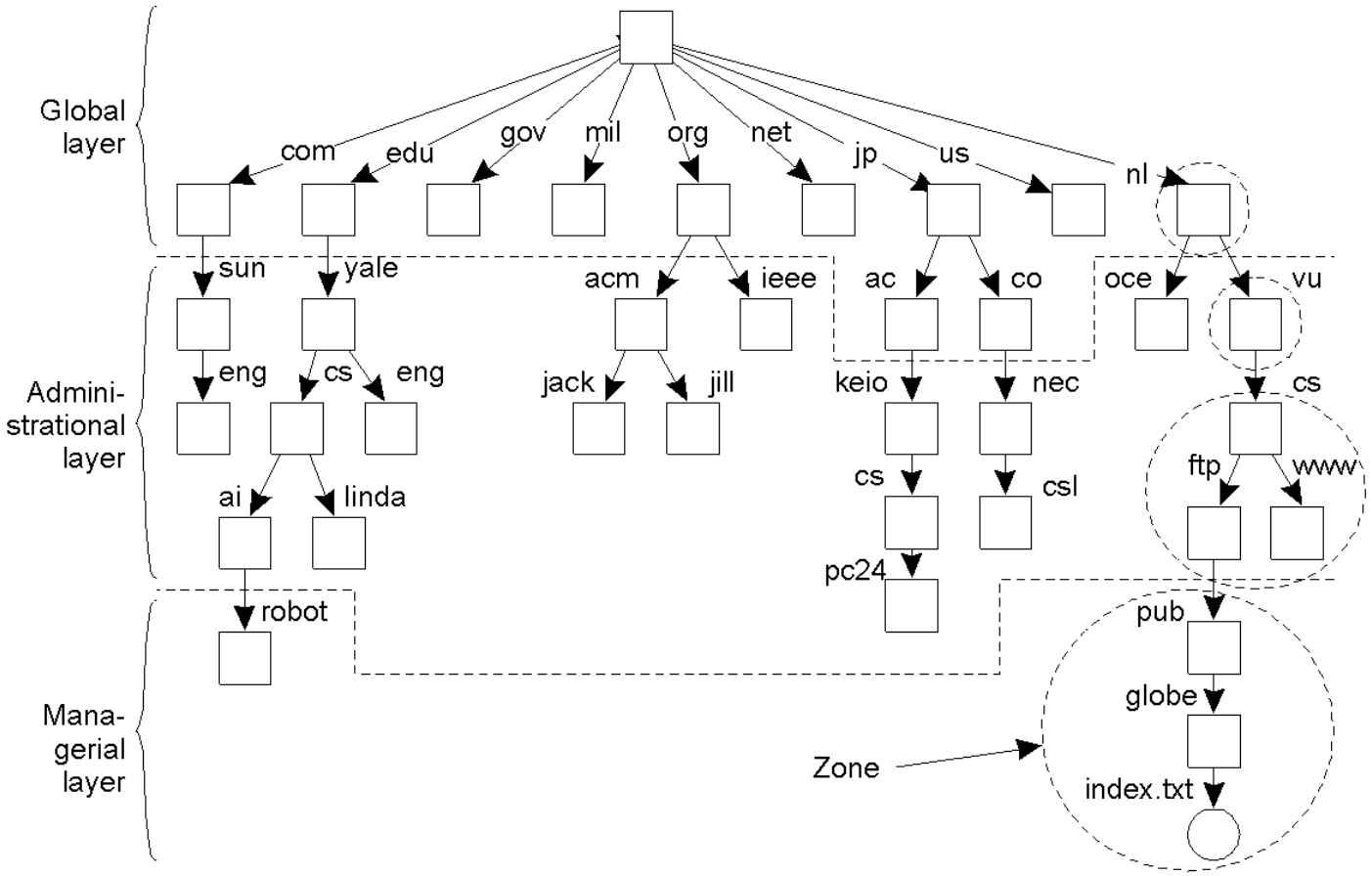
- Design goal
  - primarily used for host name resolution and for looking up mail hosts
- Characteristics
  - designed to provide short average response times for lookups
    - simple request and response protocol based on UDP
    - data are managed by a combination of partition, replication, and cache
  - temporal inconsistency is allowed
  - central authority of data consistency
  - common denominator for Internet application-level naming spaces
  - not designed for federation

Case Study: DNS (cont.)

- Name space [RFC 1034]
  - structure
    - hierarchical with a single root
    - each node has an exactly one incoming edge except the root
    - partitioned by organizations and locations
  - naming convention
    - each node is labeled with case-insensitive alpha-numeric characters
      - each label: 63 characters
      - complete path: 255 characters
    - labels are separated by “.”
      - eg: vega.kaist.ac.kr
    - read left (leaf node) to right (root or root of subtree)
  - domain name
    - list of labels on the path from node to root of sub-tree
    - relative or absolute

Case Study: DNS (cont.)

- Name space (cont.)



Case Study: DNS (cont.)

- Name service
  - DNS database is distributed across a logical network of servers
  - DNS naming data are divided into zones
    - attribute data for names in a domain
    - names and addresses of name servers
    - names of name servers for delegated sub-zones
    - zone management parameters
  - server may have data for zero or more zones
    - zone must be replicated authoritatively in at least two failure-independent servers
    - server may cache non-authoritative zone files
      - validity of cache is determined by time-to-live value

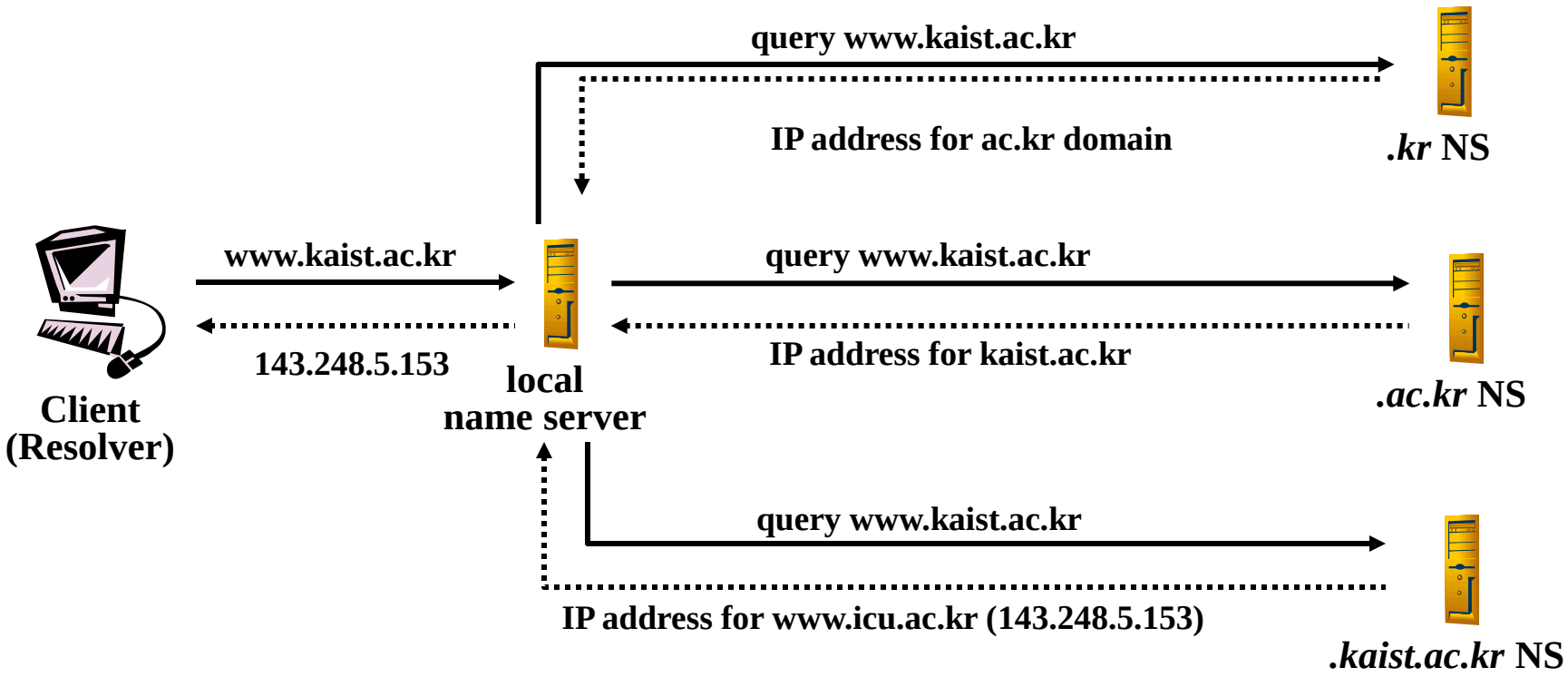
Case Study: DNS (cont.)

- Name service (cont.)
  - zone data are stored in one of several fixed types of resource records

Type of record	**Associated entity**	**Description**
SOA	Zone	Holds information on the represented zone
A	Host	Contains an IP address of the host this node represents
MX	Domain	Refers to a mail server to handle mail addressed to this node
SRV	Domain	Refers to a server handling a specific service
NS	Zone	Refers to a name server that implements the represented zone
CNAME	Node	Symbolic link with the primary name of the represented node
PTR	Host	Contains the canonical name of a host
HINFO	Host	Holds information on the host this node represents
TXT	Any kind	Contains any entity-specific information considered useful

Case Study: DNS (cont.)

- DNS name resolution [RFC 1034]
  - use DNS protocol based on UDP/IP
  - resolve chooses recursive or iterative navigation based on server's capability
  - multiple queries can be packed into a single request message



Case Study: Naming in ICN*

- ICN (Information Centric Networking)
  - Shift from host-oriented comm to content-centric
    - location-independent naming, in-network caching, and name-based routing for effective distribution of content over the network
  - general infrastructure that provides in-network caching so that content is distributed in a scalable, cost-efficient & secure manner
  - Receiver-driven model: subscribe/get objects of interest
  - Support for location transparency, mobility & intermittent connectivity
  - Needs also to be able to support interactivity (e.g. voice) and node oriented services (e.g. telnet)
- Why new naming?
  - P2P overlay and CDN are focused on how to find a local copy and do not exploit the underlying network topology

- M.F. Bari et al., "A Survey of Naming and Routing in Information-Centric Networks," IEEE Communications Magazine, Dec 2012;
- G. Pavlou, Information-Centric Networking: Introduction and Key Issues, 2015

Case Study: Naming in ICN

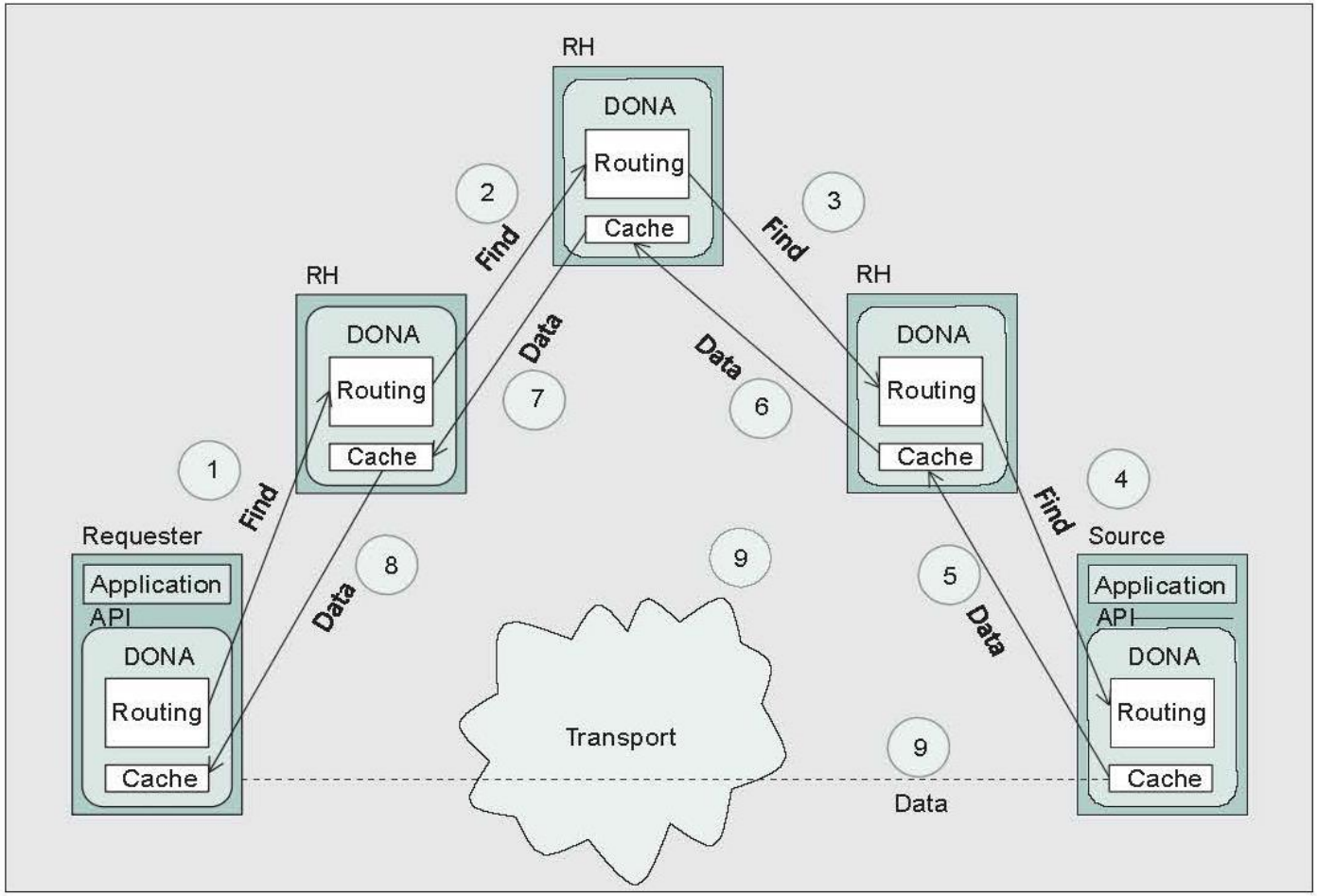
- Content Naming Issues
  - Information objects are identified by location-independent IDs, with all the object copies sharing a unique ID
  - Given that in ICN security applies to information, object IDs in many ICN architectures incorporate security
    - Non human-friendly IDs
    - Human-friendly names may additionally be associated with IDs
    - Search closely integrated with ICN given object metadata
  - Flat, hierarchical or combined ID schemes
  - Currently more than a trillion URLs (1012), many more IDs are expected
  - Scalability is a concern in particular for flat naming schemes
    - Sufficient aggregation required for hierarchical schemes

Case Study: Naming in ICN

- Name Resolution and Routing Issues
  - Two approaches: dependent on namespace/ID properties
    - Two-phase: name resolution with mapping of ID to locator, and routing to the source
    - One-phase : direct ID-based routing to source
  - The two-phase approach relies on name resolution servers
  - The locator is typically not visible to the application which uses a Get(ID) API abstraction – with Put(ID) for publishing content
  - Different characteristics of the two approaches:
    - The two-phase approach can be incrementally deployed on the current Internet given that locator-based routing is used
    - The one-phase ID-based routing is radical

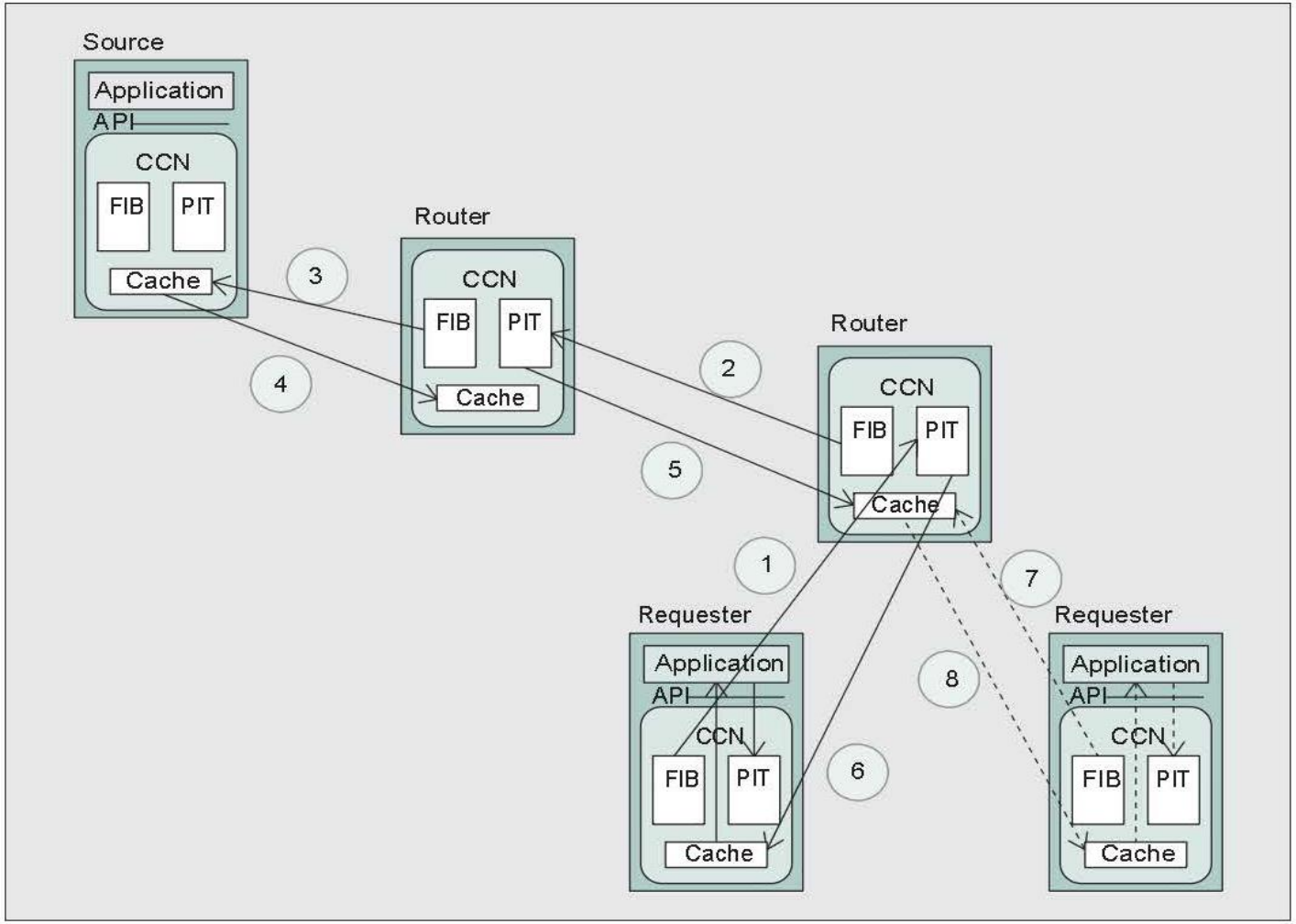
Case Study: Naming in ICN

- DONA - UCB



Case Study: Naming in ICN

- CCN – XEROX/UCLA

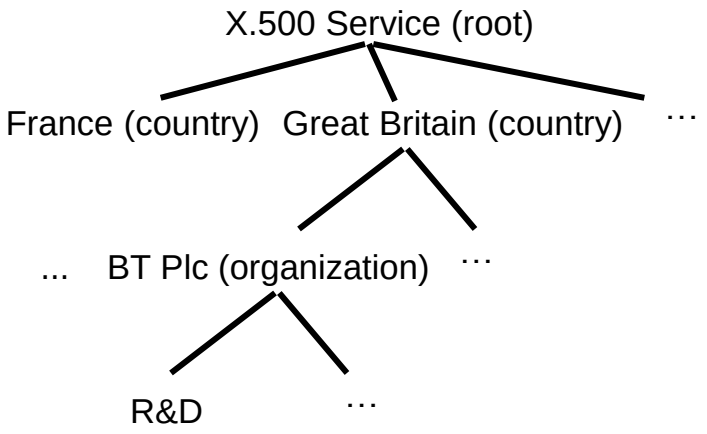


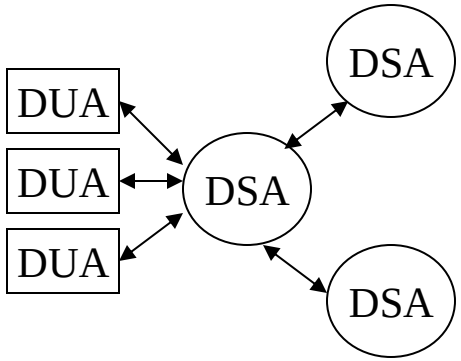
Case Study: Naming in ICN

- Name Resolution and Routing Issues
  - Two approaches: dependent on namespace/ID properties
    - Two-phase: name resolution with mapping of ID to locator, and routing to the source
    - One-phase : direct ID-based routing to source
  - The two-phase approach relies on name resolution servers
  - The locator is typically not visible to the application which uses a Get(ID) API abstraction – with Put(ID) for publishing content
  - Different characteristics of the two approaches:
    - The two-phase approach can be incrementally deployed on the current Internet given that locator-based routing is used
    - The one-phase ID-based routing is radical

Case Study: X.500

- Design goal
  - naming service supports both white page service and yellow page service
  - look up by name as well as by any required combination of attributes
- Name space
  - tree structure
  - DIT (Directory Information Tree)
    - name tree
  - DIB (Directory Information Base)
    - directory structure and attributes
    - entity: name, a set of attributes





Case Study: X.500 (cont.)

- DIB Entry

info

Alice Flintstone, Departmental Staff, Department of Computer Science,
University of Gormenghast, GB

commonName

Alice.L.Flintstone
Alice.Flintstone
Alice Flintstone
A. Flintstone

surname

Flintstone

telephoneNumber

+44 986 33 4604

uid

alf

mail

alf@dcs.gormenghast.ac.uk

Alice.Flintstone@dcs.gormenghast.ac.uk

roomNumber

Z42

userClass

Research Fellow

Case Study: X.500 (cont.)

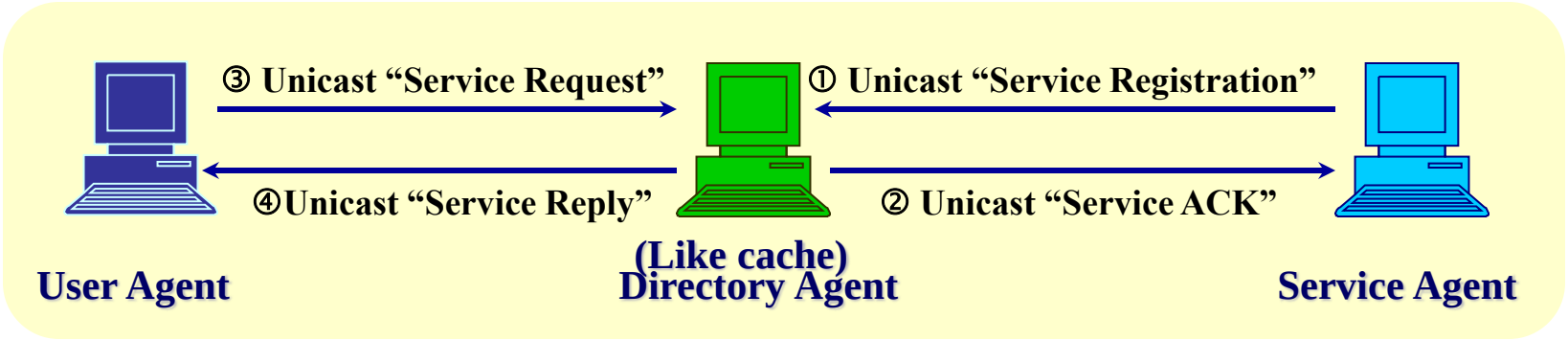
- Name service
  - read
    - DSA locates the named entry by navigating in the DIT for absolute or relative name with a list of attributed to be read and returns the required attributes
  - search
    - attribute-based access with base name and filter expression
    - return a list of names for all of entries below base node for which the filter evaluates to true
- LDAP
  - simplified interface to X.500 [RFC 2251]
  - not limited to X. 500
- Issues in X.500
  - requirement for a global directory service to exist is not clear
  - integration with Internet naming standards such as DNS
  - no clear definition for scope of information at national and international level

Case Study: SLP

- Overview
  - IETF standard protocol for service discovery and advertisement
  - Designed solely for IP-based networks
  - Doesn't define the protocol used between the client and server
- Components
  - User Agents perform service discovery on behalf of client software
  - Service Agents advertise the location and attributes of services
  - Directory Agents aggregate service information
- Service advertisement
  - Services are advertised using service URL : IP address, port number and depending on the service type, path
  - Service templates : description of the service attributes and service scheme associated with a particular service type

Case Study: SLP – Service Discovery

- Standard case



- No DA case

Case Study: Service Discovery in Ad hoc Environments

- Service discovery schemes have been extended
  - To support P2P/Virtual backbone based operation
  - To work efficiently with ad hoc routing
- Related research
  - P2P cache : E.g. [DEAPspace]
  - P2P cache + Group : E.g. [Allia]
  - Virtual backbone with selected nodes : E.g. [ULAS03]
  - Combined with routing : E.g. [M-ZRP]