Naming and Service Discovery in Peer-to-Peer Networks

ECE1770 Expert Topic

Eli Fidler
Vinod Muthusamy

February 13, 2003

Outline

- Traditional Distributed Naming Systems
- Distributed Naming Paradigms
- P2P Naming
  - Existing Systems
  - Emerging Systems

Traditional Naming Systems

- TCP/IP Host Naming
  - Static
    - hosts files
    - No central authority
  - Hierarchical
    - Domain Name System (RFC1034/5)
    - Authority for domains is delegated, but top level is centralized
    - Caching is vital for acceptable performance

Distributed Naming Paradigms

- Host IDs (CORBA Naming Service)
  - Each host is given a globally unique ID
  - Hosts are organized into hierarchical namespaces
- Service IDs (CORBA Trader Service, Jini)
  - Services are *register*ed with broker, discovered using *lookup*
- Distributes Object IDs (file sharing networks)
  - Each object has a unique ID, but may not exist in any single place

Node Discovery Techniques

- Static/Neighbours
  - Each host has a static list of known nodes/neighbours
- Centralized Repository
  - Each host knows the address of a repository
- Local Broadcast
  - A host searches for nodes using broadcast
- "Buddy List"
  - A host connects to favourite/previously seen hosts

P2P Naming

- Static
- Centralized
- Neighbour Discovery
- “Smart” Discovery
- Emerging Naming Systems

Static P2P Naming

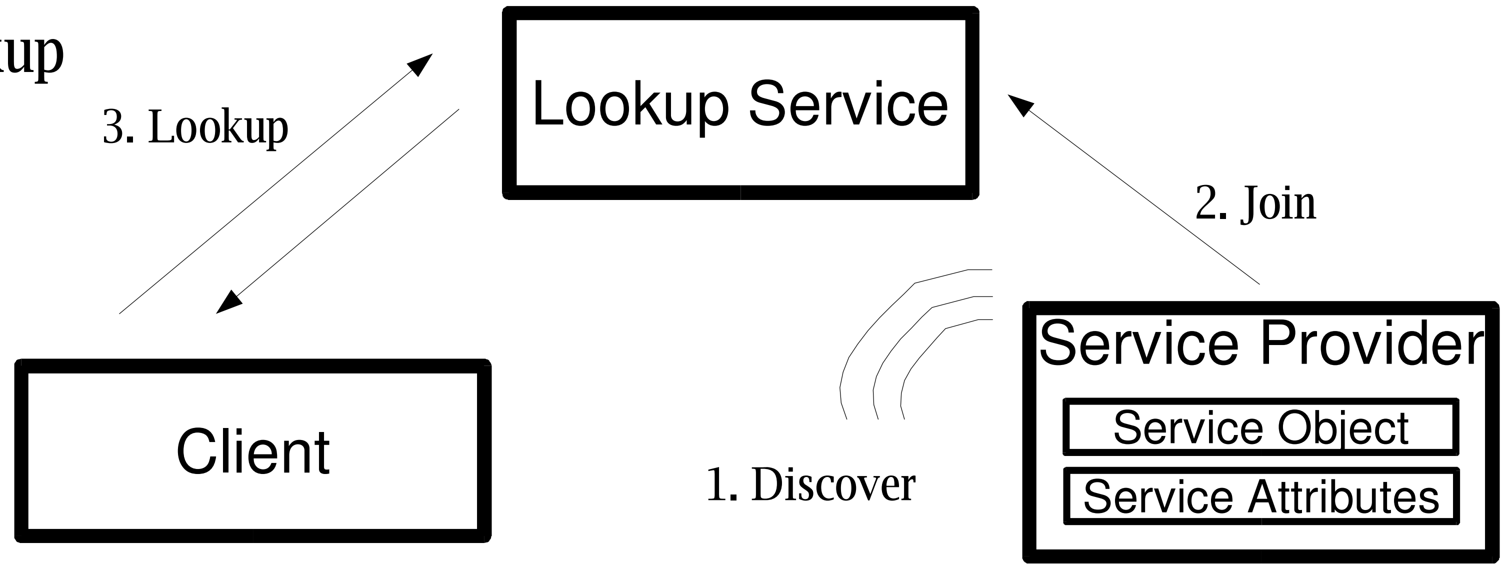
- Each host knows a static, explicit configuration of other nodes
- The P2P network is static
- c.f. hosts files

Centralized P2P Naming

- There is a single host responsible for each service (or one host for all services)
- Nodes connect to P2P network, then contact host for desired service
- ex. Napster, Jini

Jini

- Hierarchy of centralized lookup services
- Advertisement = { interface name, attributes }
- Lookup = { interface name, [attributes] }
- Object moves from Provider to Lookup Service to Client
- Must renew leases
- Peer lookup



Neighbour Discovery P2P Naming

- Once connected to P2P network, hosts use P2P neighbours to connect to services
- Searches/commands propagate in waves
- ex. Gnutella/Limewire

“Smart” Discovery P2P Naming

- Once connected to P2P network, hosts use P2P neighbours to connect to services
- Searches/commands propagate along “best” path of neighbour-neighbour links
- ex. Freenet

Freenet

- The requests get routed to the appropriate host by incremental discovery

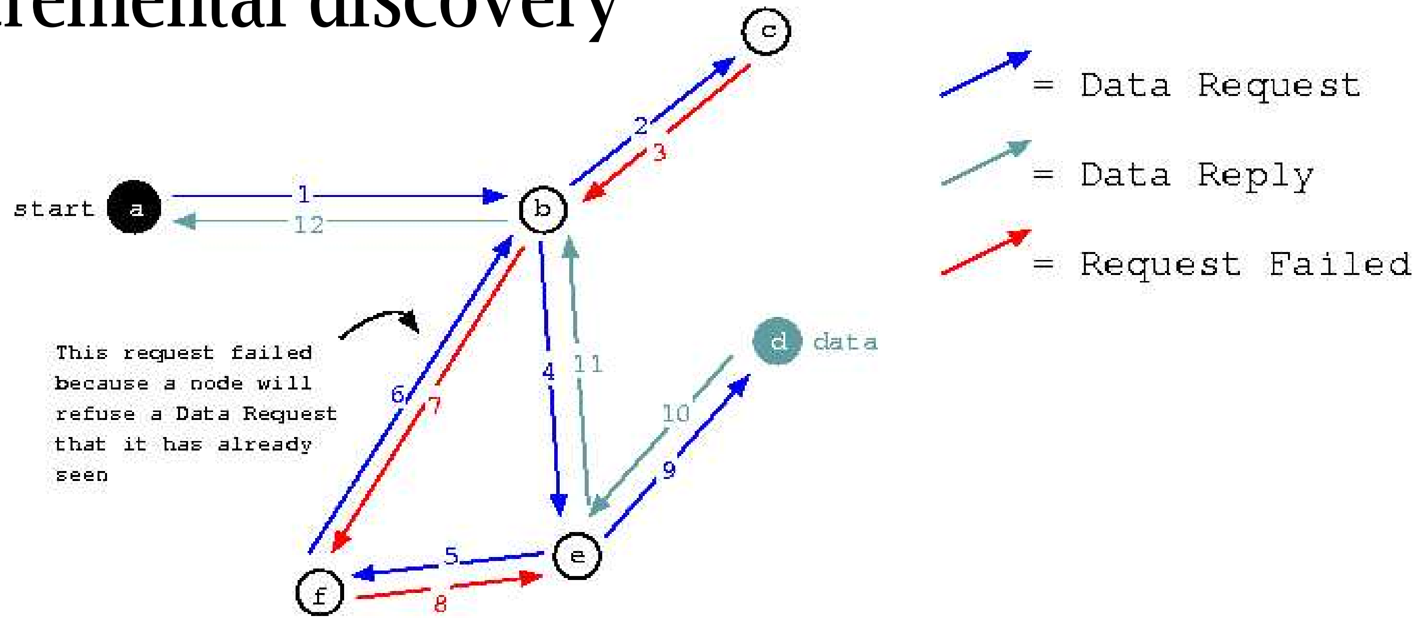

start
a
b
c
d
data
e
f
1
12
2
3
4
11
5
6
7
8
9
10
This request failed because a node will refuse a Data Request that it has already seen
= Data Request
= Data Reply
= Request Failed

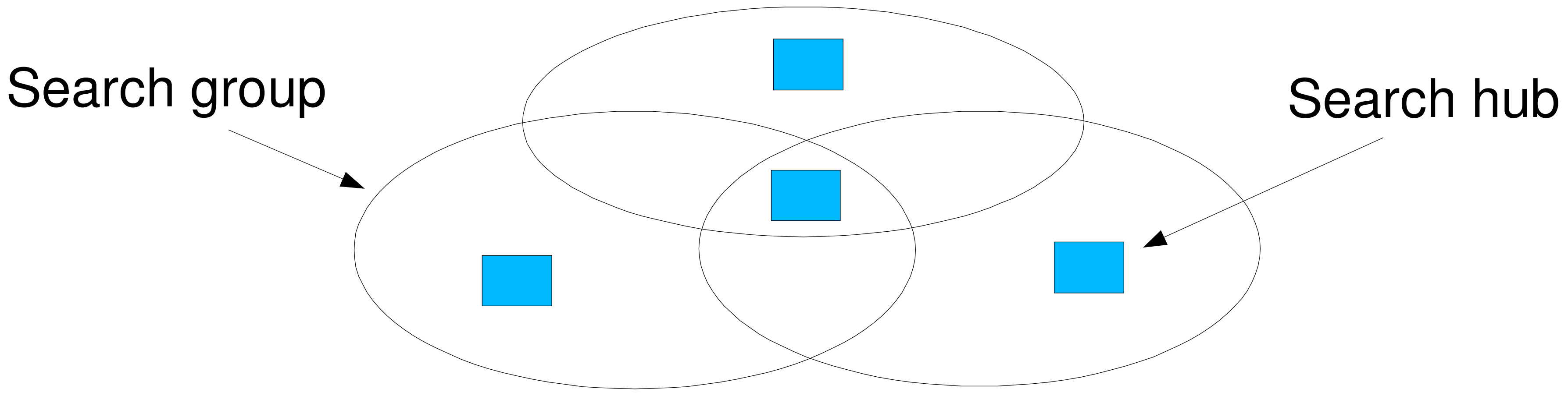
Fig. 1. A typical request sequence.

Emerging Naming Systems

- Technologies
  - JXTA
  - Intentional Naming System (INS)
  - Active Names
- Attributes
  - Naming expressiveness
  - Architecture

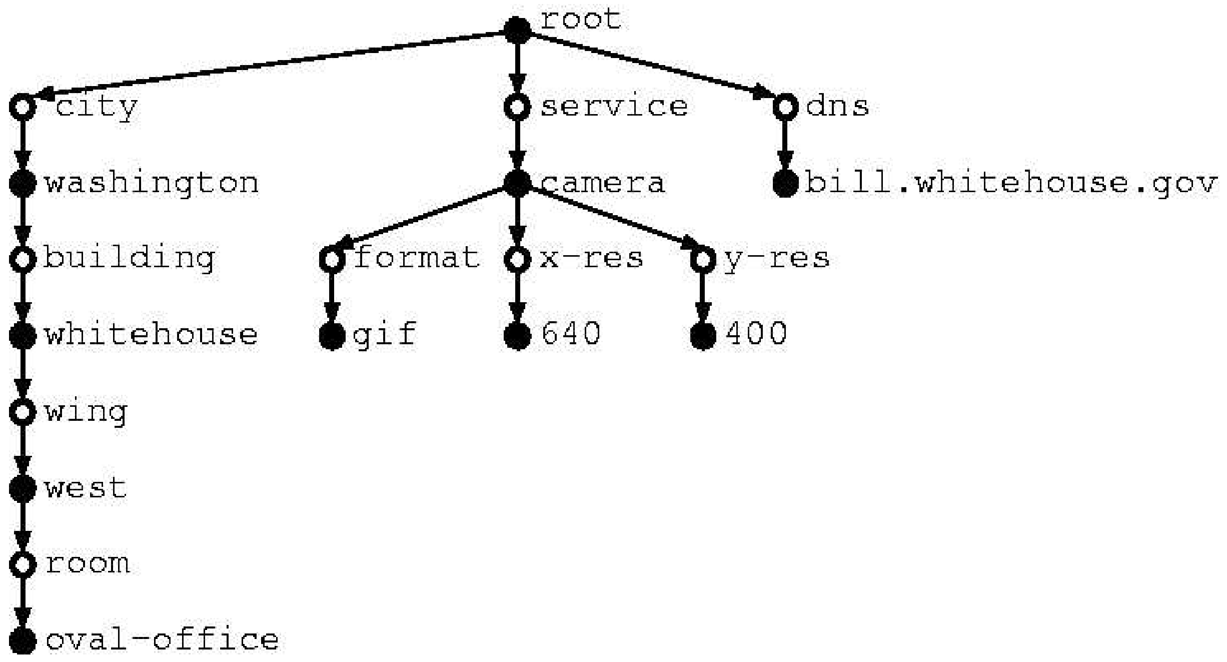
JXTA

- Super peers: distributed search hubs
- Advertisement = { query space, predicates, address }
- Query = { query space, predicates }
- Groups of hubs
  - Each group is responsible for some query space(s)
  - Each group has a member from every other group
  - Each hub has a summary of adverts in every other hub in its group



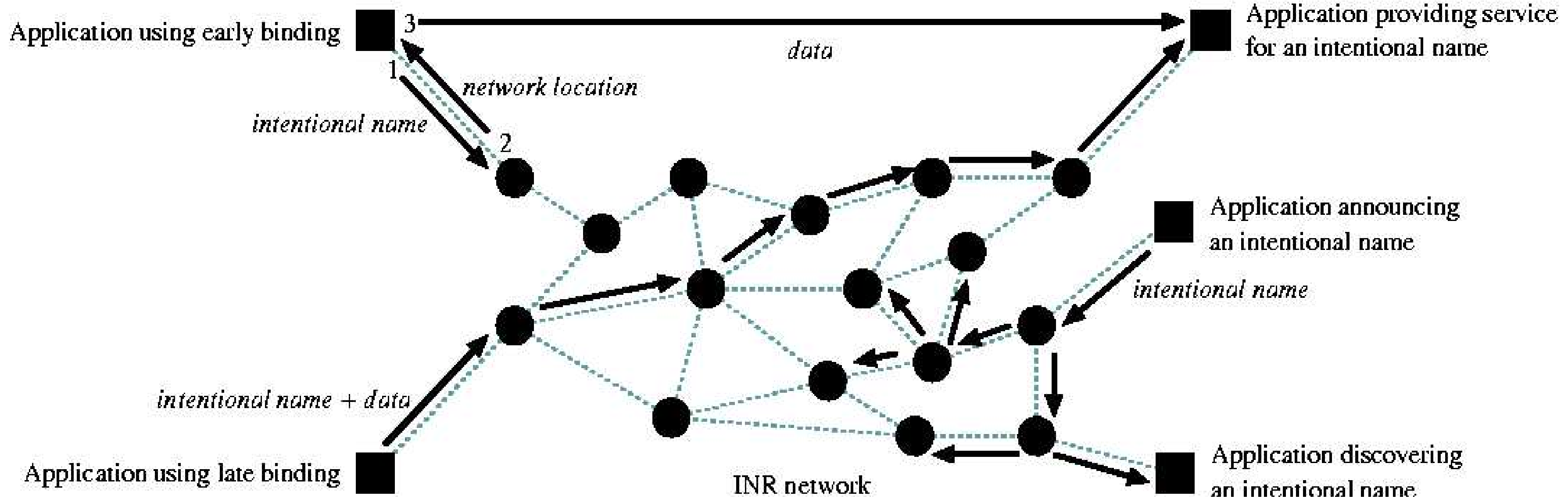
INS – Naming

- Name specifier ={ A hierarchy of attribute-value pairs }
- Name record = { Name specifier, metric, address }



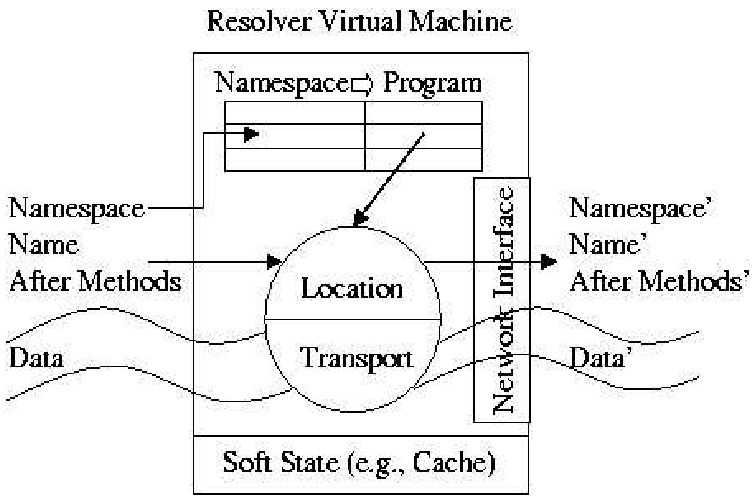
INS – Architecture

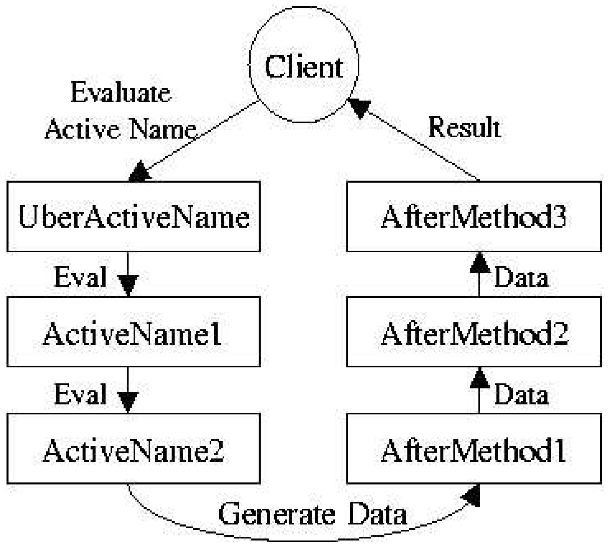
- INRs form spanning tree
- Late binding handles service/node mobility
- Name can refer to groups
- Scalability, load balancing


Application using early binding
3
1
2
network location
intentional name
data
Application providing service for an intentional name
Application announcing an intentional name
intentional name
intentional name + data
Application using late binding
INR network
Application discovering an intentional name

Active Names

- Hierarchical namespace delegation
  - Active Name = { name to resolve, namespace program }
  - Namespace program = { Active Name }
- Service composition using after methods
- Location independent execution of namespace program





Summary

- Decentralized administration
  - Well addressed
- Network failures, robustness
  - Addressed by periodic advertisements
  - Automatic resolver spawning in INS
- Lookup
  - Typically need (distributed) servers (INS, Freenet, etc.)
  - Flooding (Gnutella) is inefficient

Summary (Cont'd)

- Query expressiveness
  - Primarily still hierarchical (INS, AN, Jini, etc.)
- Node/service mobility
  - Addressed by periodic advertisements
  - Late binding in INS
- Scalability
  - Many are not scalable to Internet (INS, JXTA, Jini)
  - Rely on lookup service hierarchy for scalability