Design and implementation of an intentional naming system

William Adjie-Winoto    Elliot Schwartz
Hari Balakrishnan    Jeremy Lilley

MIT Laboratory for Computer Science
http://wind.lcs.mit.edu/

SOSP 17, Kiawah Island Resort
December 14, 1999
Environment

- Heterogeneous network with devices, sensors and computers
- Dynamism
  - Mobility
  - Performance variability
  - Services “come and go”
  - Services may be composed of groups of nodes
- Example applications
  - Location-dependent mobile apps
  - Network of mobile cameras
- Problem: resource discovery
Design goals and principles

Expressiveness
Names are intentional; apps know what, not where

Responsiveness
Integrate name resolution and message routing (late binding)

Robustness
Decentralized, cooperating resolvers with soft-state protocol

Easy configuration
Name resolvers self-configure into overlay network
Naming and service discovery

- Wide-area naming
  - DNS, Global Name Service, Grapevine
- Attribute-based systems
  - X.500, Information Bus, Discover query routing
- Service location
  - IETF SLP, Berkeley service discovery service
- Device discovery
  - Jini, Universal plug-and-play
- Intentional Naming System (INS)
  - Mobility & dynamism via late binding
  - Decentralized, serverless operation
  - Easy configuration
INS architecture

Client

Name resolver

Overlay network of resolvers

Service

Intentional anycast

Intentional multicast

Late binding
Name with message

Message routing using intentional names
Name-specifiers

- Expressive name language (like XML)
- Resolver architecture decoupled from language
- Providers announce descriptive names
- Clients make queries
  - Attribute-value matches
  - Wildcard matches
  - Ranges

```
[vspace=lcs.mit.edu/camera]
[building=ne43
    [room=510]]
[resolution=800x600]]
[access=public]
[status=ready]
```

```
[vspace=mit.edu/thermometer]
[building=ne43
    [floor5=
        [room=*]]
[temperature<60°F]]
```
Name lookups

- **Lookup**
  - Tree-matching algorithm
  - AND operations among orthogonal attributes

- **Polynomial-time in number of attributes**
  - $O(n^d)$ where $n$ is number of attributes and $d$ is the depth
Resolver network

- Resolvers exchange routing information about names
- Multicast messages forwarded via resolvers
- Decentralized construction and maintenance
- Implemented as an “overlay” network over UDP tunnels
  - Not every node needs to be a resolver
  - Too many neighbors causes overload, but need a connected graph
  - Overlay link metric should reflect performance
  - Current implementation builds a spanning tree
Spanning tree algorithm

- Loop-free connectivity
- Construct initial tree; evolve towards optimality
  - Select a destination and send a discover_bottleneck message along current path

UDP tunnel

A
new

B

max
Late binding

• Mapping from name to location can change rapidly
• Overlay routing protocol uses triggered updates
• Resolver performs lookup-and-forward
  – lookup(name) is a route; forward along route
• Two styles of message delivery
  – Anycast
  – Multicast
Intentional anycast

- lookup(name) yields all matches
- Resolver selects location based on advertised service-controlled metric
  - E.g., server load
- Tunnels message to selected node
- Application-level vs. IP-level anycast
  - Service-advertised metric is meaningful to the application
Intentional multicast

- Use intentional name as group handle
- Each resolver maintains list of neighbors for a name
- Data forwarded along a spanning tree of the overlay network
  - Shared tree, rather than per-source trees
- Enables more than just receiver-initiated group communication
Robustness

• Decentralized name resolution and routing in “serverless” fashion
• Names are weakly consistent, like network-layer routes
  – Routing protocol with periodic & triggered updates to exchange names
• Routing state is soft
  – Expires if not updated
  – Robust against service/client failure
  – No need for explicit de-registration
Performance and scalability

- Lookup performance

- Spawn INR on a new node to shed load
Routing Protocol Scalability

- vspace = Set of names with common attributes
- Virtual-space partitioning: each resolver now handles subset of all vspaces

Routing updates for all names

Delegate this to another INR

Name-tree at resolver

vspace=camera

vspace=5th-floor
INR Implementation

- Overlay Manager
- Network Monitor
- Route Manager
- Client Manager
- vspace neighbors
- Forwarder
- NameTreeSet
- Communicator
- Mobility Sockets
- TCP/UDP

Incoming message

Intentional anycast, multicast

lookup
Applications

• Location-dependent mobile applications
  – Floorplan: An map-based navigation tool
  – Camera: A mobile image/video service
  – Load-balancing printer
  – TV & jukebox service

• Sensor computing

• Network-independent “instant messaging”

• Clients encapsulate state in late-binding applications
Status

• Java implementation of INS & applications
  – Several thousand names on single Pentium PC; discovery time linear in hops
  – Integration with Jini, XML/RDF descriptions in progress

• Scalability
  – Wide-area implementation in progress

• Deployment
  – Hook in wide-area architecture to DNS
  – Standardize virtual space names (like MIME for devices/services)
Conclusion

• INS is a resource discovery system for dynamic, mobile networks
• Expressiveness: names that convey intent
• Responsiveness: late binding by integrating resolution and routing
• Robustness: soft-state name dissemination with periodic_refreshes
• Configuration: resolvers self-configure into an overlay network