Discovery

Robert Grimm
New York University
The Problem: Naming 
(Or, How to Start a Religious War)

- The Internet today
  - IP addresses
    - Strict location
  - DNS names
    - Fuzzy location
      - Think cluster-based services, content distribution networks (Akamai)
      - Hierarchical naming structure enables hierarchical implementation

- But, we would like to use more descriptive names
  - We need a layer of indirection (see last week)
Five case studies
- UDDI, Jini, SDS, INS, one.world

All five systems rely on a centralized directory
- Services register with directory
- Clients look up appropriate services in the directory

But, they differ significantly in focus, architecture, and capabilities
**Difference in Focus**

- **e.business** (notably B2B communications)
  - UDDI

- Locating services in ever more dynamic networks
  - Jini, SDS, INS, *one.world*

**What are the implications of difference in focus?**

- Scope of directory
- Expected rate of change of directory information
- Persistence of directory information
Differences in Architecture

- **UDDI**
  - Provides standardized descriptions for e-business
  - Dedicated, global servers hosted by Microsoft, IBM
  - Hard state

- **Jini, SDS, INS, one.world**
  - Provide mechanisms for discovery
  - Local servers
    - But…
  - Soft state
Hard State vs. Soft State

- **UDDI**
  - Directory is a "real" database at a well-known location

- **Jini, SDS, INS, one.world**
  - Directory is an ever changing database at an undetermined location
    - Need to locate directory
      - Multicast queries and announcements
      - Directory directory or meta-directory
    - Need to refresh directory contents
      - Periodic registrations
      - Leases
Differences in Capabilities

- Structure of descriptions
- Expressiveness of queries
- Supported communication patterns
  - Early vs. late binding
  - Anycast vs. multicast
  - Forward vs. reverse lookups
- Security
Diggin’ Deeper: UDDI
UDDI Data Model

- **businessEntity**
  - Basic information on business including name, contact
  - **identifierBag** to record business identifiers
  - **categoryBag** to record business categories

- **businessService**
  - Services provided by business

- **bindingTemplate**
  - Access information for service

- **tModel**
  - Pointer to external technical specification
    - D&B D-U-N-S numbers, UNSPSC, WSDL
UDDI API

- Builds on SOAP
- Identifies all records by UUIDs
- Includes set of methods to discover records
  - find_business, find_relatedBusiness, find_service, find_bindings, find_tModel
- Includes set of methods to retrieve detailed records
  - get_businessDetail, get_serviceDetail, get_bindingDetail, get_tModelDetail
- Performs left-2-right substring matching by default
  - Optionally supports exact name, case sensitive, and sound-alike queries
Finding Businesses and Services in More Detail

- What can we search on?
  - name
  - categoryBag
  - tModelBag

- For businesses only, also
  - identifierBag
  - discoveryURLs

- Is this enough???
Focus on Mechanism:
Jini, SDS, INS, one.world
Jini

- Set of services to help build dynamically configurable networks of services
  - Distributed leasing
  - Distributed events
  - Distributed transactions
  - JavaSpaces
    - Based on Linda’s tuple space model
  - Discovery
- Builds on Java RMI
  - With exception of multicast for discovery
Jini Discovery

- Manually configured directory server
  - Announces itself through multicast
  - Clients also query for server through multicast
- Service registrations are leased
- Descriptions are so-called *entries*
  - Public fields of objects implementing *Entry* interface
  - Fields must be objects, not primitive types
- Matching based on templates
  - Objects in template must equal objects in entry
  - Null value treated as wild card
- **Directory server**
  - Announces itself (and CA, CM) through multicast

- **Services**
  - Announce themselves through multicast as well

- **Clients**
  - Query directory server through authenticated RMI

- **Certificate authority (CA)**
  - Manages bindings between principals and keys

- **Capability manager (CM)**
  - Converts ACLs into capabilities, distributes them to clients
SDS Security

- All communications are secure
  - Authenticated RMI between servers and for lookups
    - Also encrypted for privacy
    - Handshake establishes symmetric key between endpoints
  - Authenticated directory server announcements
    - Signed by server, but not encrypted
  - Secure service description announcements
    - ID, Ciphered Secret, Payload
- Services are only visible to authorized clients
  - Service descriptions are associated with capabilities
  - Capability manager creates them based on ACLs
  - Clients present them during lookups
SDS Scalability
Hierarchy to the Rescue

- Basic idea: Maintain a hierarchy of directory servers
- Hierarchy management
  - Many hierarchies maintained in an unspecified way
- Description aggregation and query routing
  - Subset hashes collected in Bloom filter tables
    - Maintained for each child independently
    - Aggregated up the hierarchy (by or-ing childrens’ tables)
    - Periodically cleared (or per-bit counts)
- Queries routed…
  - Down the hierarchy on table match
  - Up the hierarchy on no match
Directory servers form overlay network

- Form spanning tree based on communication latencies
- Resolve lookups
- Route messages (late binding + anycast/multicast)
- Adjust to load by starting or shutting down servers
- Selected from pre-assigned set maintained by Domain Space Resolver (DSR)

Services periodically register with any server

- Include application-specific metric for anycast selection

Clients interact with any server
INS Name Trees

- Descriptions are nested attribute value pairs
- All descriptions are combined into a single name tree
  - Alternating levels for attributes and values
  - Leaves point to name records
    - Routes to next hop directory server
    - IP addresses for end-points
    - Metrics
    - Expiration time
- Lookups are based on set-based algorithm
  - Weed out non-reachable name records
- Retrieving descriptions requires backtracking
Some background

- All data are tuples (records of name/value pairs)
- All functionality implemented by event handlers
  - Events are data are tuples

Discovery

- Relies on per-network directory server
  - Automatically elected from local devices
- Fully integrated with point-to-point communications


- **export**
  - Make event handler remotely accessible
    - Descriptor may be `null`, `Name`, `Query`, or any other tuple
  - Resulting *binding* between name and handler is leased

- **lookup**
  - Find event handler(s) matching query (early binding)

- **send**
  - Send event through point-to-point communications
  - Send event through late binding discovery
    - Using anycast or multicast
Directory server announces itself through multicast
After two missed announcements, per-device election manager calls election
During fixed period, each device broadcasts a score
  - CPU speed, memory size, uptime, connectivity
Device with highest score wins election
  - Starts discovery server
How to tolerate inconsistencies?
  - Export all services to all visible directory servers
  - Only query one visible directory server
  - Directory server with lower score shuts itself down
Comparing Mechanisms: Jini, SDS, INS, one.world
Descriptions and Queries

- Public fields and equality matching
  - Jini
- XML and equality matching
  - SDS
- Nested attribute/value pairs and equality matching (extension for simple comparison in the works)
  - INS
- Tuples and arbitrary queries
  - one.world
- What are the trade-offs here?
Services

- Java objects
  - Jini
- Network address and protocol
  - SDS, INS
- Event handler
  - one.world
- What is the trade-off here?
  - Integration
  - Flexibility
<table>
<thead>
<tr>
<th>Which Device Can Be a Directory Server?</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ Manually configured device</td>
</tr>
<tr>
<td>▪ Jini</td>
</tr>
<tr>
<td>▪ Manually configured devices</td>
</tr>
<tr>
<td>▪ SNS</td>
</tr>
<tr>
<td>▪ Some devices out of a pre-configured pool</td>
</tr>
<tr>
<td>▪ INS</td>
</tr>
<tr>
<td>▪ Any device on the local network</td>
</tr>
<tr>
<td>▪ <em>one.world</em></td>
</tr>
<tr>
<td>▪ <em>What are the implications on manageability, trust?</em></td>
</tr>
</tbody>
</table>
To Multicast Or Not to Multicast

- Directory server
  - Jini, SDS, \textit{one.world}
- Clients
  - Jini
- Services
  - SDS
- No multicast
  - INS
    - Uses Domain Space Resolver instead (extension to DNS)
- \textbf{What are the trade-offs here?}
Communication Patterns

- Early binding and forward lookups
  - Jini, SDS, INS, one.world

- Late binding
  - INS, one.world

- Anycast and multicast
  - INS, one.world
    - INS includes numeric metric to select best match for anycast

- Reverse lookups
  - one.world

- What are the trade-offs here?
Underlying Network Protocol

- RMI (over TCP)
  - Jini, SDS
- UDP
  - INS
- UDP, TCP
  - one.world
- What are the implications on reliability?
Discussion

- Is discovery as a research topic dead?
- What is the right trade-off?
  - Expressiveness
  - Programmability
  - Adaptability (responsiveness, robustness, manageability)
  - Security
  - Scalability