Programming models 2

Brian Nielsen bnielsen@cs.aau.dk

Registry = NameServer



Motivation

- Example scenarios
 - "Find all nearby color duplex printers"
 - "Start brewing coffee five minutes before my alarm clock goes off"
 - "Let my cell phone use the car speakers"
- Coordination framework
 - Simple, seamless, and scalable interoperability
 - Network "plug and play" with minimum admin
 - Networked software and hardware provide services
 - Any device can find and use existing services

One Way Scenarios Might Be Done Today



What Jini Proposes



Spontaneous networking

- Jini enables clients to automatically discover services at runtime
- Associative search
 - Not by name lookup (e.g. http://some.url:port)
 - Instead: find a service that does this or that
- Loose coupling
 - Services and clients can join and leave the system (Jini federation) at any time without causing system failure

Jini and Java

Jini Services	 JavaSpaces[™] Transaction Managers Printing, Storage, Databases
Jini Infrastructure	DiscoveryLookup Service
Jini Programming Model	 Leasing Distributed Events Transactions
Java 2 Platform	• Java RMI • Java VM

Service Interface

- Everything is a service on the network.
- Interfaces define what a service offers, not how it implements it
- Java interfaces gives strong typing
- The ServiceItem Java object represents a service registration:
 - A unique service ID
 - The service proxy object
 - A set of optional Java attribute objects
 - (e.g. location, status, vendor, etc)
- Services are found in the lookup service by template matching described by a ServiceTemplate object

Discovery, join, lookup protocols

• Join protocol

- Services register with lookup services
- Registration is a Java object

• Discovery protocol

- Clients and services find lookup services by multicasting requests at predefined IP multicast address or direct addressing (PULL)
- Lookup service may advertise its existence by periodic multicasting its presence (PUSH)

Lookup protocol

- Clients send a search request to lookup services
- Specify service type and properties as a Java object
- Lookup service finds and returns matching services
- Returned is a Java object (ServiceItem) that carries the service proxy

Service Registration





Alarm Clock service's Proxy Object



Find interface AlarmClock(radio=yes, vendor=*)



Can use any protocol to communicate to service (or proxy can contain service itself!)

The role of the proxy



- The proxy is a Java object downloaded from the service
- Hides communication between proxy and service
 - Costumizable, dedicated protocol (transparent)
 - Protocol independent (TCP/IP, HTTP, SOAP, etc.)
 - Replication
 - Updatable
 - protocol
 - Interact directly with physical devices

Leasing

- What happens if a service dies?
 - It must be de-registered at the lookup service
 - May crash, or "forget" to do so
- A lease is a time period during which the grantor of the lease promises that the holder of the lease will have access to some resource
 - Eg. The lookup service stores a service registration
- A client requests a lease from the provider (lease grantor) and must periodically renew it.
- Expired or non-renewed lease results in removing lease and freeing up resources

Event Based Systems

Event driven programming

- Programming principle based on the idea that objects should react on events (state change, either concretely or abstractly) happening in other objects.
- Objects *publishes* information about events that other objects might be interested in knowing about.
- Objects can *subscribe* to receive information about when an event occur.

Event Programming



Architecture for distributed event notification



- 1. OOI directly notifies subscriber
- 2. OOI nofies subscriber via observer
- 3. Observer queries (polls) OOI and notifies subscriber

Remote Events

- Remote events more expensive to send than local events
- Remote events propagate slowly
- Remote events may not arrive in order sent
- Remote events may not arrive at all (or at all subscribers)
- What does sender do if receiver has crashed -- keep trying? How long?

Jini Remote Events

Jini Remote Events



Jini Events

- *Event generators:* An object, that allows other objects to subscribe to its events and generates notification (via RMI).
- *Remote event listeners:* An object that can receive notifications.
- *Remote events:* An object that is passed by value to remote event listeners (a notification).
- *Third-party agents:* Objects that may be interposed between an object of interest and a subscriber *(observers).*
- Subscriptions are subject to leasing

RemoteEventListener interface

public interface RemoteEventListener extends Remote,

```
java.util.EventListener
```

```
void notify(RemoteEvent theEvent)
```

```
throws UnknownEventException,
RemoteException;
```

```
}
```

ł

RemoteEvent Class

```
public class RemoteEvent extends java.util.EventObject {

public RemoteEvent( Object source,

long event ID,

long SeqNum,

MarshalledObject handback) {...}
```

```
public Object getSource() {...}
    public long getID() {...}
    public long getSequenceJumber() {...}
    public MarshalledObject getRegitrationObject {...}
}
```

source+eventID+SeqNum uniquely identifies event occurrence:

 \Rightarrow Notify is IDEMPOTENT

Example *EventGenerator*

The EventRegistration Class

public class EventRegistration implements java.io.Serializable
{

```
public EventRegistration( long eventID,
Object eventSource,
Lease eventLease,
long seqNum) {...}
```

```
public long getID() {...}
public Object getSource() {...}
public Lease getLease() {...}
public long getSequenceNumber() {...}
```

3rd party objects (Observers)

- Store-and-forward agents
- Notification filters
- Notification mailboxes

Store and forward agent



- Delivery policies (eg retry policies)
- reliable Multicast, hardware multicast



Notification Demultiplexing



Notification mailbox



1) Mailbox stores events until registrant comes "online".

2) When registrant comes"online" the stored events are delivered.



Linda (and JavaSpaces)

David Gelernter, yale University.

Linda

- Program Concurrency by using uncoupled processes with shared data space
- Add concurrency into a sequential language by adding:
 - Simple operators
 - Runtime kernel (language-independent)
 - Preprocessor (or compiler)



- Have a shared memory space ("tuple space")
 - Processes can add, read, and take away values from this space
- Bag of processes, each looks for work it can do by matching values in the tuple space
- Implicit load balancing, synchronization, messaging, etc.

Tuples

	Conventional Memory	Linda/JavaSpaces
Unit	Bit	Logical Tuple (23, "test", false)
Access Using	Address (variable)	Selection of values
Operations	read, write	in, out, read JavaSpaces: read, write, take Immutable

Tuple Space Operations

- **out** (t) add tuple *t* to tuple space
- in (s) → t returns and removes tuple t matching template s
- read (s) \rightarrow t same as in, except doesn't remove t.
- Originally also **Eval**(f(42),g(45));
- Operations are atomic (even if space is distributed)

Meaning of in

in ("f", int n)	Tuple Space
in ("f", 23)	("f", 23)
in ("t", ? bool b, ?int n)	("t", 25) ("t", true)
in (?string s,? int n)	("t", false) ("f", 17)
in ("cookie")	(, , , , , , , , , , , , , , , , , , ,

Counting Semaphore

- Create (int n, String resource)
 for (i = 0; i < n; i++) out (resource);
- Down (String resource) in (resource)
- Up (String resource)
 out (resource)

Distributed Ebay

• Offer Item (String item, int min bid, int timeout):

out (item, minbid, "owner");

sleep (timeout);

in (item, ? bid, ? bidder);

if (bidder ≠ "owner") SOLD!

 Bid (String bidder, String item, int bid): in (item, ? highbid, ? highbidder); if (bid > highbid) out (item, bid, bidder) else out (item, highbid, highbidder)

Further Reading

• Jini Specification

http://www.sun.com/jini/specs

- Jini Developers <u>http://jini.org</u>
- Universal Plug and Play

http://www.upnp.org

Salutation

http://www.salutation.org

