Distributed Coordination-Based Systems

Adapted from: "Distributed Systems", Tanenbaum & van Steen, course slides

Need for Coordination

- so far we have mostly focused on request/reponse types of interactions between a client and a server
  - means the client and server are tightly coupled
  - client blocks until server delivers response
  - what if the one of the parties crashes?
  - what if more than two parties need to be involved?

Introduction to Coordination Models

- A taxonomy of coordination models (adapted from [cabri.g2000])

		Temporal	
		Coupled	Uncoupled
Referential	Coupled	Direct	Mailbox
	Uncoupled	Meeting oriented	Generative communication

Mailbox Model

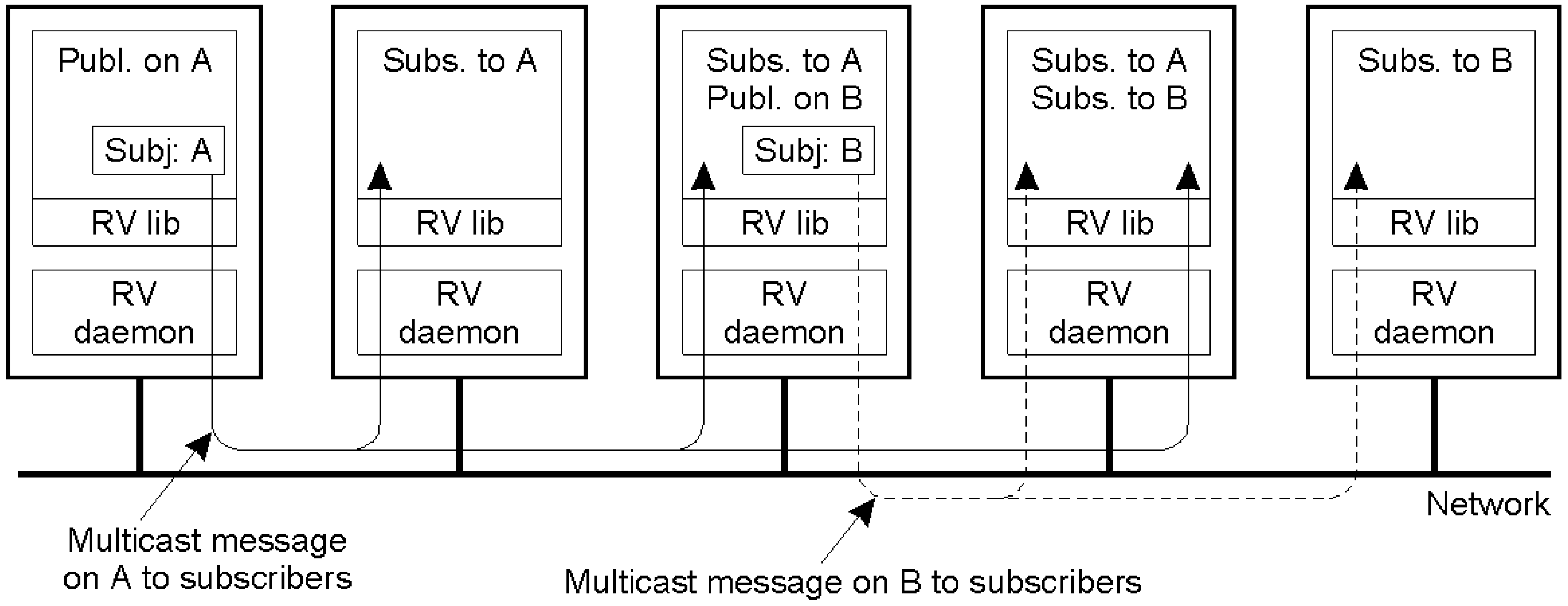
- it is in particular supported by the concept of "active objects" or "actors"
  - Actors run in their own thread of execution
  - they communicate via asynchronous method calls
  - some work has been done to implement distributed actors, but it is still mostly a research topic

Meeting Oriented

- To some extent, the Java event model allows for looser coupling between the client and the server
  - the server, as an event source, does not explicitly know who is listening
  - this is the basis of publish-subscribe
- Without such coupling, the client often has to keep polling the server

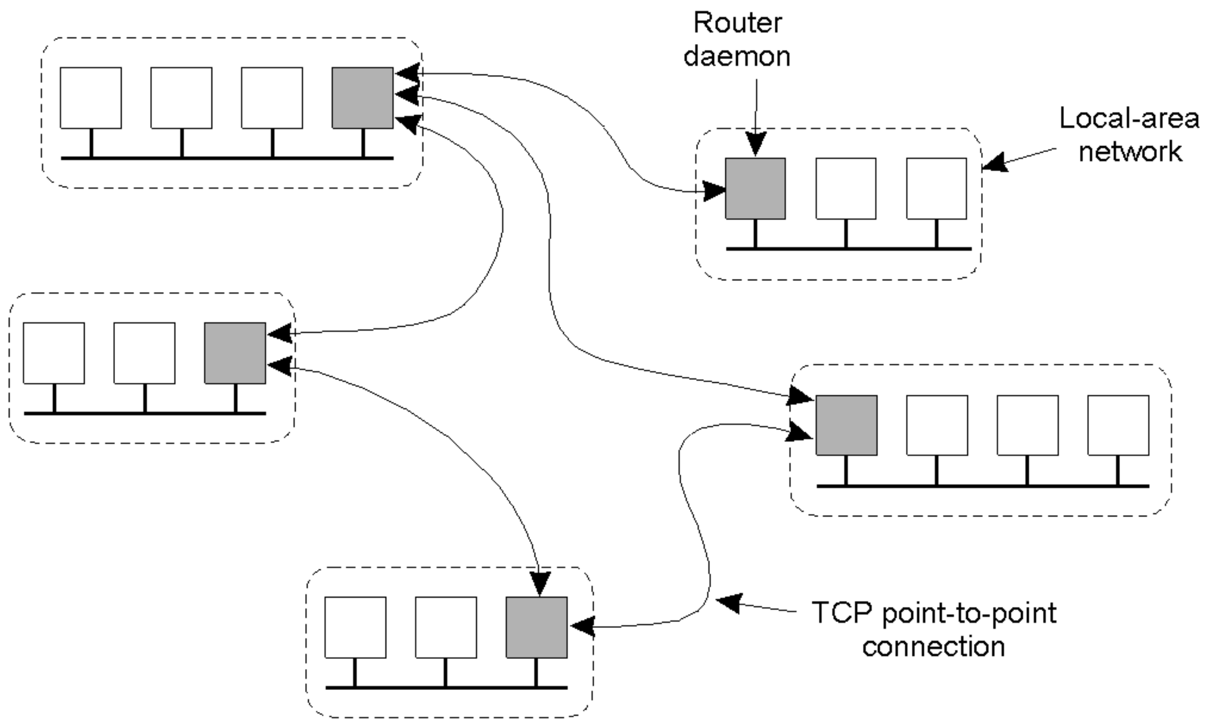
Meeting Oriented

- The principle of a publish/subscribe system as implemented in TIB/Rendezvous.



Coordination Model (2)

- The overall architecture of a wide-area TIB/Rendezvous system.



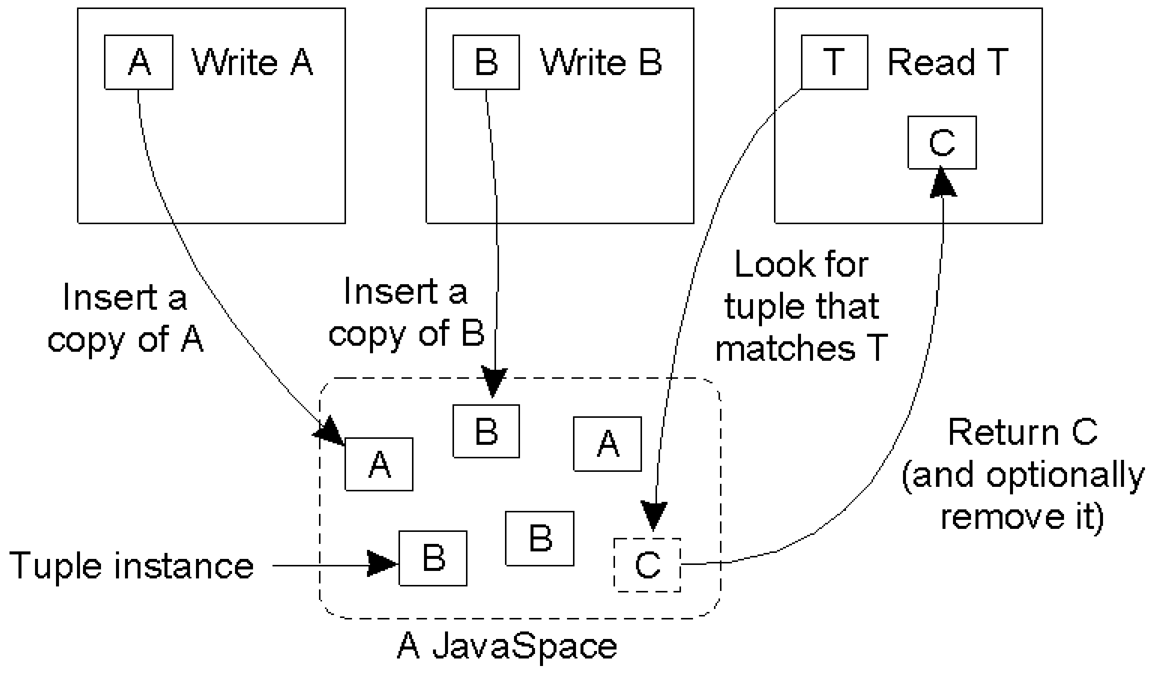
Basic Messaging

Attribute	Type	Description
Name	String	The name of the field, possibly NULL
ID	Integer	A message-unique field identifier
Size	Integer	The total size of the field (in bytes)
Count	Integer	The number of elements in the case of an array
Type	Constant	A constant indicating the type of data
Data	Any type	The actual data stored in a field

- Attributes of a TIB/Rendezvous message field.

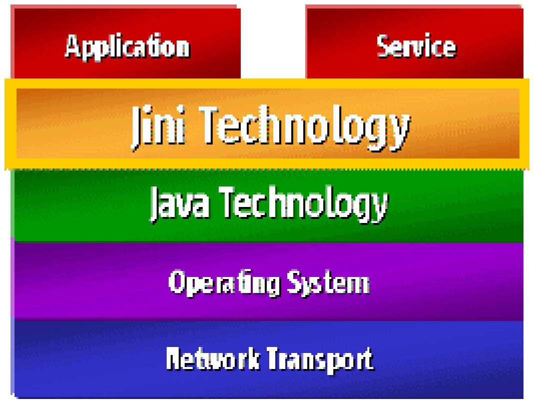
Generative Communications: Tuple-spaces

- JavaSpaces is an implementation of the concept of tuple-spaces, and is used by Jini



What is Jini?

- “makes computers and devices able to quickly form impromptu systems unified by a network”
- based on Java (JDK1.2)
- open source (SCSL license)



Jini

- Jini Services are represented and accessed through a Java Interface
- mobility of the implementation of the services is made possible using RMI

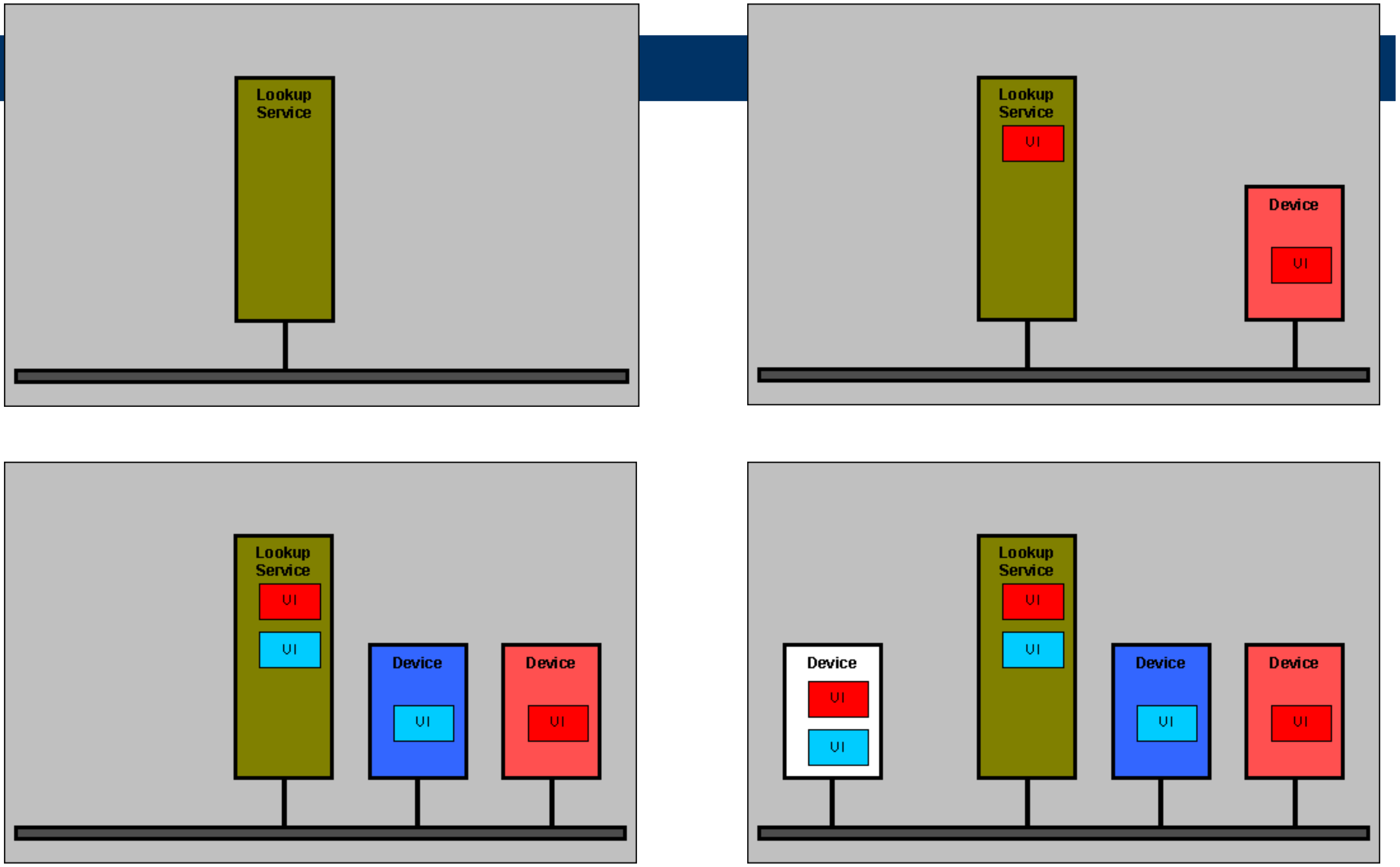
The Jini Architecture:
Key concepts and components

- Services
- Lookup Service
- RMI
- Security
- Leasing
- Transactions
- Events

Jini Lookup Service

- Repository of available services
- Stores services as Java objects
- Clients download services on demand

How it works



Lookup Service
Lookup Service
UI
Device
UI
Lookup Service
UI
UI
Device
UI
Device
UI
Device
UI
UI
Lookup Service
UI
UI
Device
UI
Device
UI

The Jini Lookup Service (1)

Field	**Description**
ServiceID	The identifier of the service associated with this item.
Service	A (possibly remote) reference to the object implementing the service.
AttributeSets	A set of tuples describing the service.

- The organization of a service item.

The Jini Lookup Service (2)

Tuple Type	Attributes
ServiceInfo	Name, manufacturer, vendor, version, model, serial number
Location	Floor, room, building
Address	Street, organization, organizational unit, locality, state or province, postal code, country

- Examples of predefined tuples for service items.

Discovery Protocol

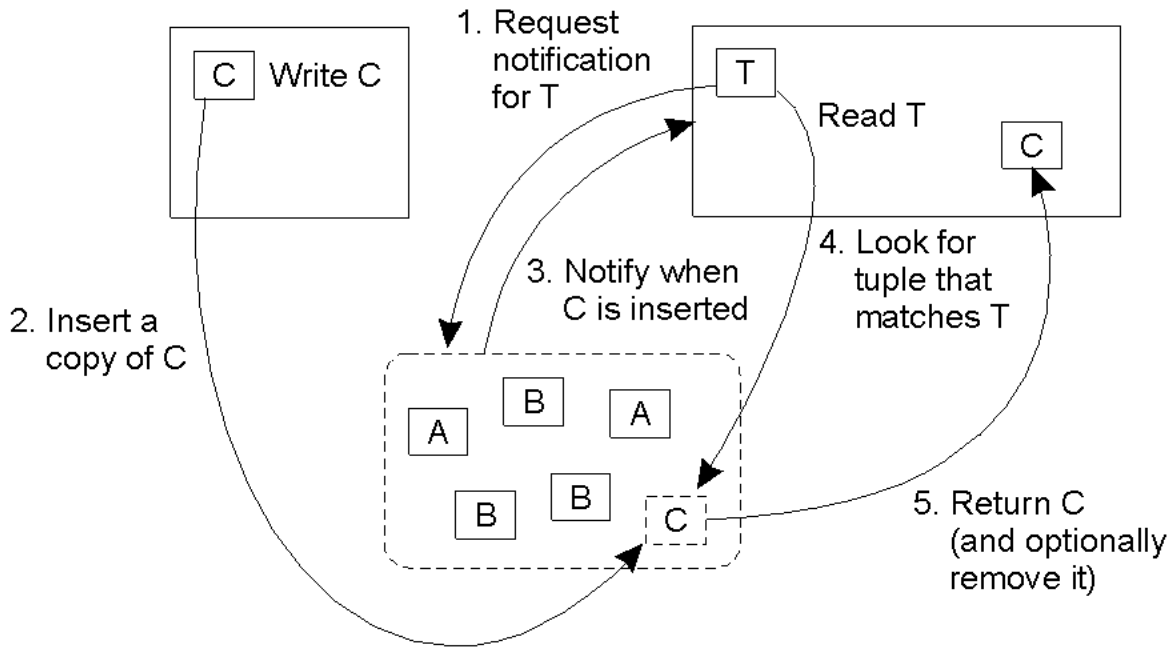
- a process of finding lookup services
- used by both Jini services and clients
- Discovery Model
  - Multicast discovery for LAN

Leasing

- For managing resources based on time duration
- leases are time-based grants of resources or services
  - loose contracts between grantor and holder
  - can be cancelled, renewed, allowed to expire...

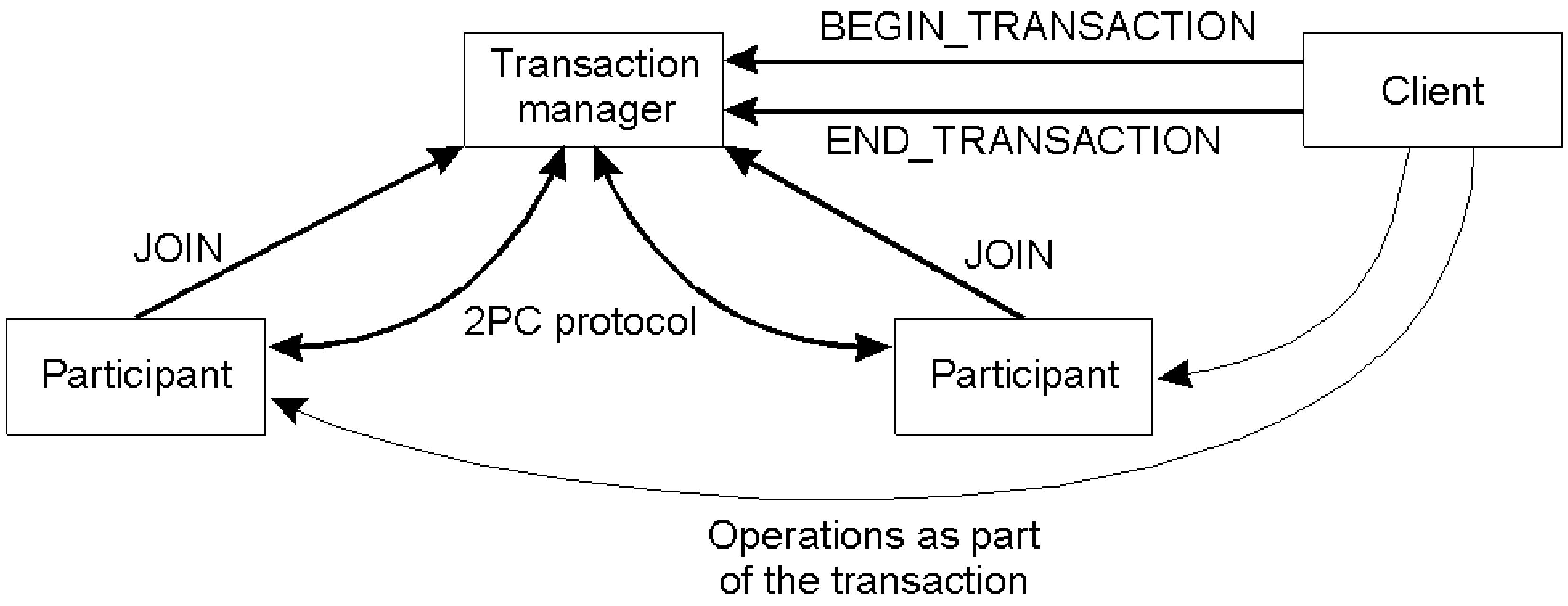
Communication Events

- Using events in combination with a JavaSpace


C
Write C
1. Request notification for T
T
Read T
C
2. Insert a copy of C
3. Notify when C is inserted
4. Look for tuple that matches T
A
B
A
B
B
C
5. Return C (and optionally remove it)

Synchronization of Transactions

- The general organization of a transaction in Jini. Thick lines show communication as required by Jini's transaction protocol



Set-up

- Start a Web server (to enable class downloading)
- Start the RMI daemon
- Start the lookup service (Reggie)
- Optional: run the Transaction Manager
- Optional: run the JavaSpaces Service
- Run the services

Comparison of TIB/Rendezvous and Jini

Issue	**TIB/Rendezvous**	**Jini**
Major design goal	Uncoupling of processes	Flexible integration
Coordination model	Publish/subscribe	Generative communication
Network communication	Multicasting	Java RMI
Messages	Self-describing	Process specific
Event mechanism	For incoming messages	As a callback service
Naming services	None	Lookup service
Transactions (operations)	Messages	Method invocations
Transactions (scope)	Single process (see text)	Multiple processes
Locking	No	As JavaSpace operations
Caching and replication	No	No
Reliable communication	Yes	Yes
Process groups	Yes	No
Recovery mechanisms	No explicit support	No explicit support
Security	Secure channels	Based entirely on Java