Announcements

• Use Discussion Board to look for partner
• Reminder:
  – presentation time limit 20-25 minutes
• Condor paper reading assignment will be changed
• Book chapters covered so far 1-3, 12

Recap

<table>
<thead>
<tr>
<th></th>
<th>Temporal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coupled</td>
<td>Direct</td>
</tr>
<tr>
<td>Uncoupled</td>
<td>Mailbox</td>
</tr>
<tr>
<td>Coupled</td>
<td>Meeting</td>
</tr>
<tr>
<td>Uncoupled</td>
<td>Generative communication</td>
</tr>
</tbody>
</table>

Outline for Today

• More on Tuples
  – JavaSpaces/JINI
• Publish/subscribe Systems
  • Threads

JavaSpaces

Example: Game Service

```java
import net.jini.core.entry.Entry;
public class Ticket implements Entry {
    public String gameName;
    public Game game;
    public Ticket() {
        this.gameName = gameName;
        this.game = game;
    }
    public Ticket(String gameName) {
        this.gameName = gameName;
    }
    public Ticket(String gameName, Game game) {
        this.gameName = gameName;
        this.game = game;
    }
}
```
Game Service: Player

```java
public class Player {
    protected String gameName;
    GameServiceInterface gameInterface;
    public void mainloop() {
        while (true) {
            Game game = gameInterface.joinGame(gameName);
            try {
                System.out.println("Playing game " + gameName);
                game.play(myName);
            } catch (RemoteException e) {
                e.printStackTrace();
            }
            gameInterface.leaveGame();
            /* sleep */
        }
    }
}
```

Game Service: GameInterface (I)

```java
class GameServiceProxy implements GameServiceInterface {
    private String name; // keep track of current game
    private Game game;
    public Game joinGame(String name) {
        this.name = name;
        Ticket ticketTemplate = new Ticket(name);
        Ticket ticket;
        try {
            ticket = (Ticket)space.take(ticketTemplate, null, Long.MAX_VALUE);
        } catch (Exception e) { e.printStackTrace();   }
        return game;
    }
}
```

Game Service: GameInterface (II)

```java
class GameServiceProxy implements GameServiceInterface {
    public void leaveGame() {
        if (game == null) {
            System.out.println("Not participating in a game!");
        } else {
            Ticket ticketEntry = new Ticket(name, game);
            try {
                space.write(ticketEntry, null, Lease.FOREVER);
            } catch (Exception e) { e.printStackTrace();  }
            System.out.println("Left the game " + name);
            name = null;
            game = null;
        }
    }
}
```

Side note: Leases

- First analyzed by Gray/Cheriton (1989) in context of distributed filesystems:
  - Server provides data + lease to client
  - Client “owns” data for duration of the lease
  - Server delays writes until lease expires or can be revoked
- Disadvantage: requires physical clock synchronization
- What effect does duration of lease have?

Leases in JavaSpaces

- `JavaSpace.write()` returns Lease that says how long used implementation is willing to store Entry
- In general: promise access to a resource for a limited time

Implementing TupleSpaces

- Locally
  - Use Subspaces
  - Hashing
- Distributed
  - To replicate or not to replicate
Replicated JavaSpaces

Unreplicated JavaSpaces

Partial replication

• Logical Grid

Event-based Communication

Delivery Models

• Pull model
• Push model
• Hybrid
  – pulled from producer, pushed to subscriber
  – push from producer, pulled by subscriber

Example: TIB/Rendezvous
Summary

• Referentially uncoupled (or content-based) communication
  – Generative communication: Tuple Spaces
  – Meeting-oriented: Publish/Subscribe

• Thursday: Two Scheduling Papers