SuDuelKu
Fault Tolerant Distributed Systems

Team 1
Saul Jaspan
Yudi Nagata
Lucía de Lascurain
Chris Nelson
Luis Rios
Agenda

- Application Configuration
- Fault Tolerant Architecture
- Questions and Assumptions
- Lessons learned
- What’s next
Application Configuration

- EJB
- Linux
- Java
- Using:
  - JBoss
  - MySQL
  - Lomboz
  - Eclipse
Fault Tolerant Architecture - 1

- Client 1
- Client N
- Name Server
- Server 1
- Server 2 (backup)
- Database
- Fault Injector
- Recovery Manager
Fault Tolerant Architecture - 2

- **SuDuelKuClient**
  - **EJBClient**
    - **FaultInjecting EJBClient**
      - void setPacketDropRate(...)
      - void setCommandRate(...)
  - **FaultInjector**
    - void startNewClient(...)
    - void setPacketDropRate(...)
    - void setLoad(...)
    - void setErrorTypes(...)

- **SuDuelKu Server**
  - **Recovery Manager**
    - void startNewServer(...)
  - **FaultTolerant SuDuelKu Server**
    - void checkPoint(...)
    - bool isAlive(...)
    - void setPacketDropRate(...)
    - void setCheckpointFrequency(...)
    - void setAsPrimary(...)
    - void addBackup(...)
    - void addCommand(...)

- **CommandQueue**
  - void addCommand(...)
  - void clearQueue(...)
  - void getCommands(...)

Diagram showing relationships and methods related to fault tolerance in SuDuelKu.
Questions and Assumptions

- **Questions:**
  - Are we really 100% stateless?
  - Do we need checkpointing if we are stateless?
  - Should we have a separate JNDI?

- **Assumptions:**
  - Our naming server will not be killed.
  - Our message IDs will be committed to the database with the transactions atomically.
Lessons Learned

- EJB learning curve steep but eases development once summited
- Calling back from the server to the client is unnatural with middleware
- Fault Recovery is MUCH easier when the middle tier is stateless
What’s next

- Develop fault tolerant system.
- Develop fault injector.
- Obtain system metrics.
Questions