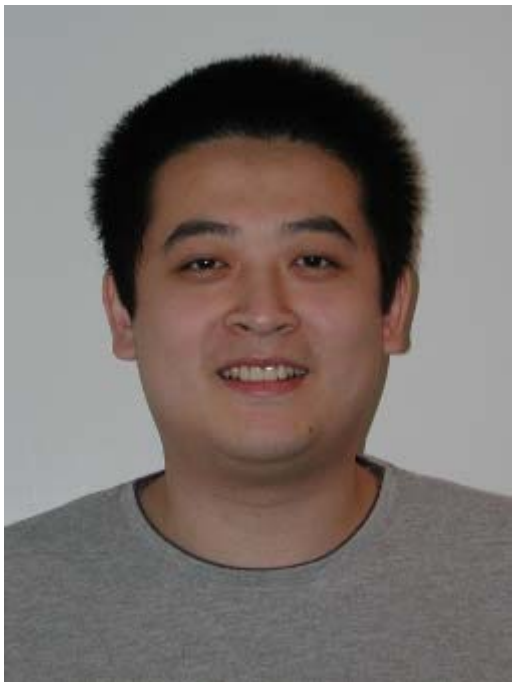
Distributed Bingo

17-654: Analysis of Software Artifacts
18-841: Dependability Analysis of Middleware

Team 5:
Jack Yao
Bubba Beasley
Kai Liao
Alex Berendeyev

http://www.ece.cmu.edu/~ece846/team5

Team Members



Kai Liao

Bubba Beasley

Alex Berendeyev

Real-World Demonstration

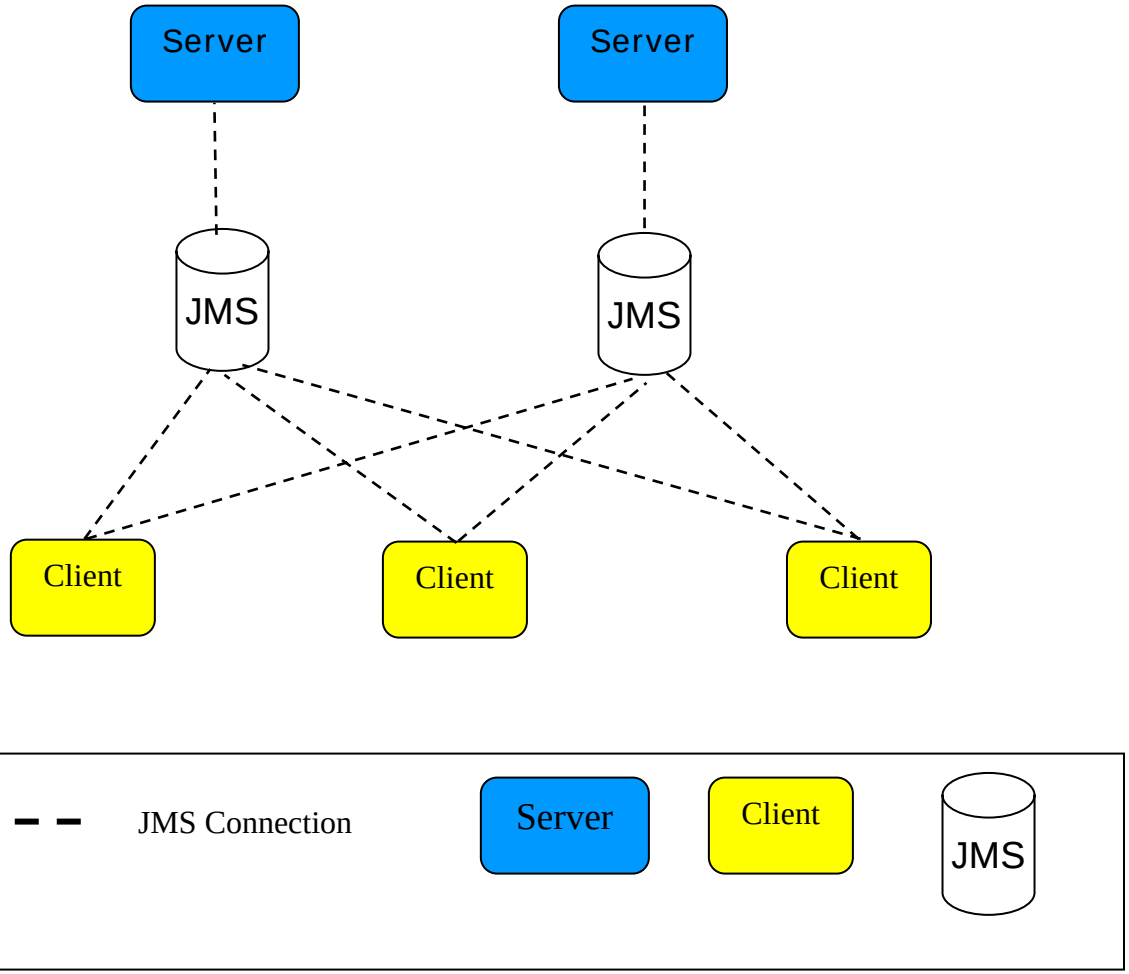
Real-World Characteristics

- Each player gets a Bingo card to start
- A player joining mid-game can catch up with knowledge of previous draws
- The host announces each draw
- The winning player announces Bingo
- The host verifies the win
- The host announces the win

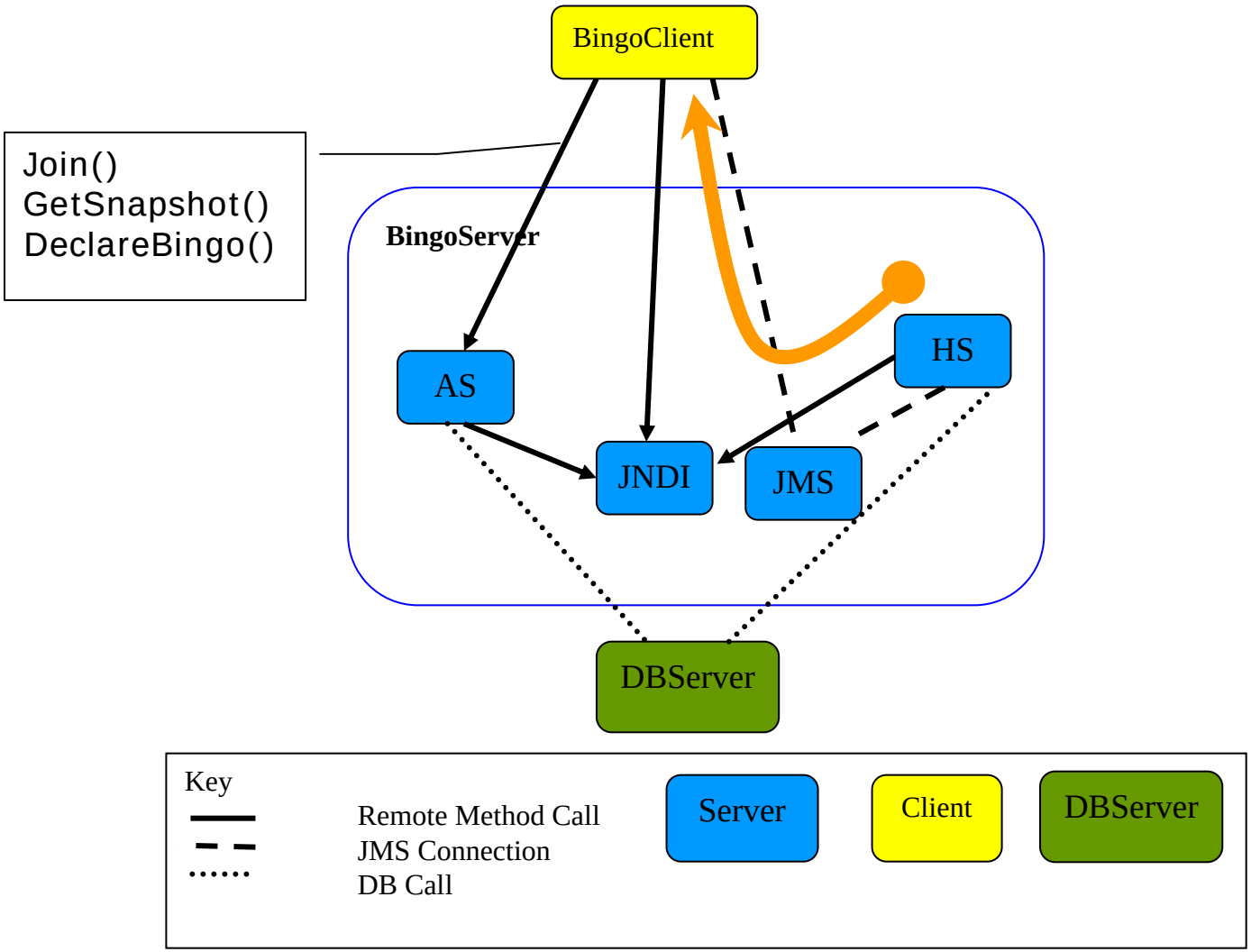
Baseline Application

- A distributed, online version of Bingo
- The clients *pull* data from the servers and the servers *push* data to the clients
- DB: SQL Server 2000, Windows XP, MSE Cave machine
  Servers: JBoss (JNDI, JMS) on Linux, ECE Game cluster
  Clients: Windows, Linux (theoretically anywhere)
    - Automated command-line client
    - Interactive GUI-based client

Publish/Subscribe with JMS



Baseline Architecture



From the Real to the App

- Each player gets a Bingo card to start
- A player joining mid-game can catch up with knowledge of previous draws

The Client asks the AnsweringServer to join and receives a bingo card and all previous draws.

- The host announces each draw

At regular intervals (5 seconds), the HostServer broadcasts the draws via the JMS.

From the Real to the App

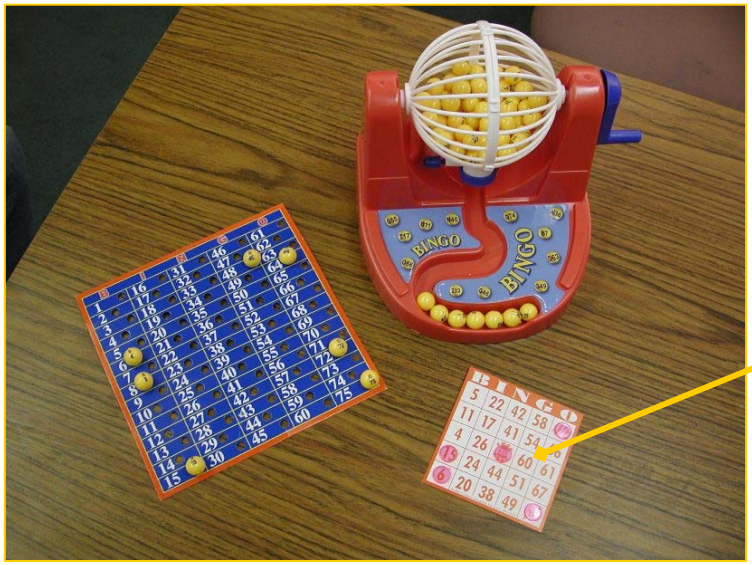
- The winning player announces Bingo
- The host verifies the win
- The host announces the win

The Client asks the AnsweringServer to verify Bingo

The AnsweringServer verifies and stores the winner's ID in the database

The HostServer checks for a winner in the DB and broadcasts that there is a win

GUI Interface



Java GUI on top of a Java command-line interface

Miscellany

- Each game: 100 draws, rather than 75
- Approximately 1.4 x 10^{30} card combinations
  (1.4 billion trillion trillion cards)
- Duplicate cards are not a problem, so theoretically no limit on the number of players in a single game
- No guarantee of fairness in declaring a winner

Fault Tolerance Goals (1)

Server Faults

- JBoss Process crashes
- Machine crashes

Network Faults

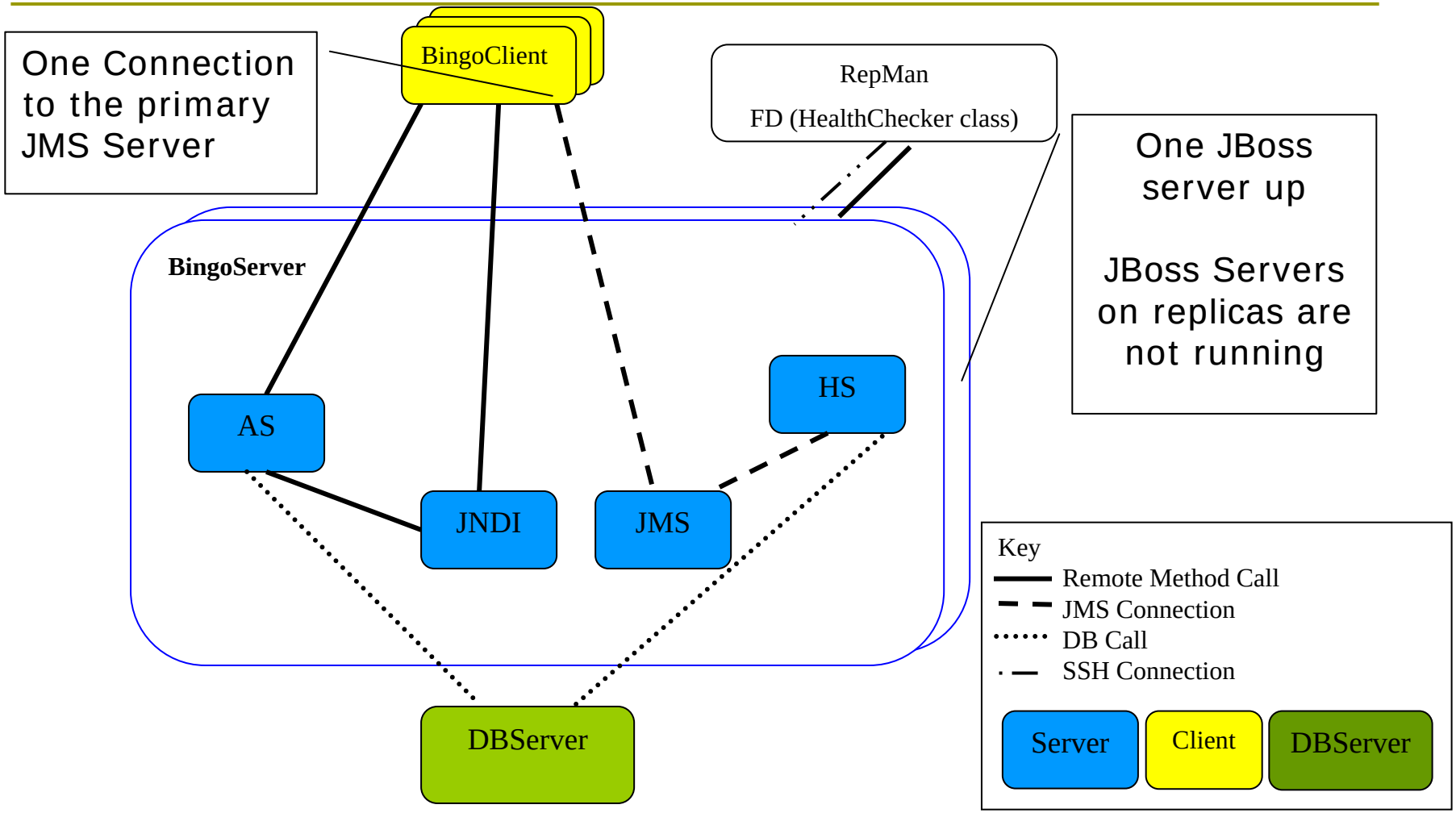
“Sacred Machine” Assumptions

- Replication Manager
- Global Fault Detector
- DB Server

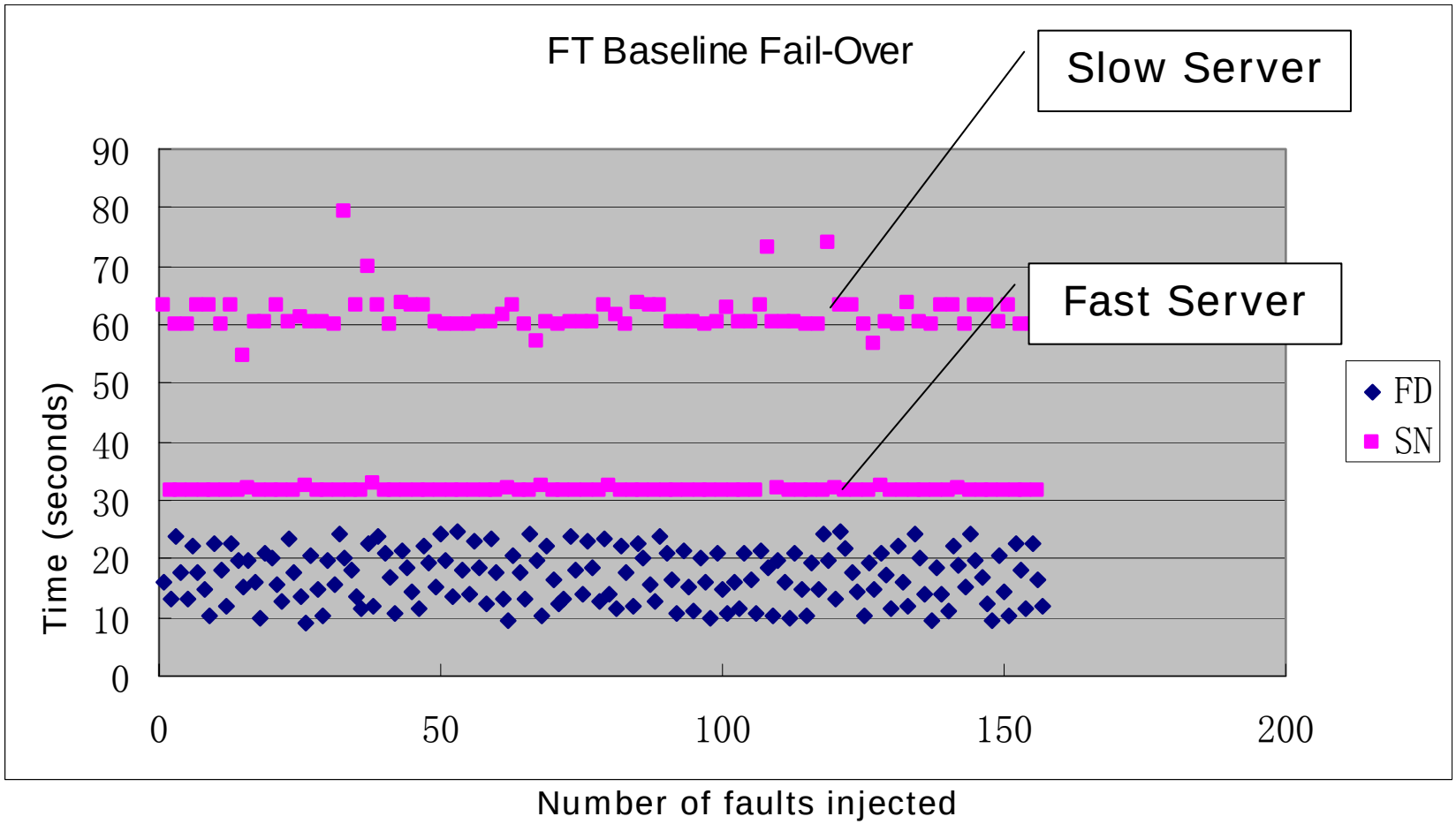
Replicas

N replicas
Tested with 3 replicas
Round-robin recovery of JBoss servers

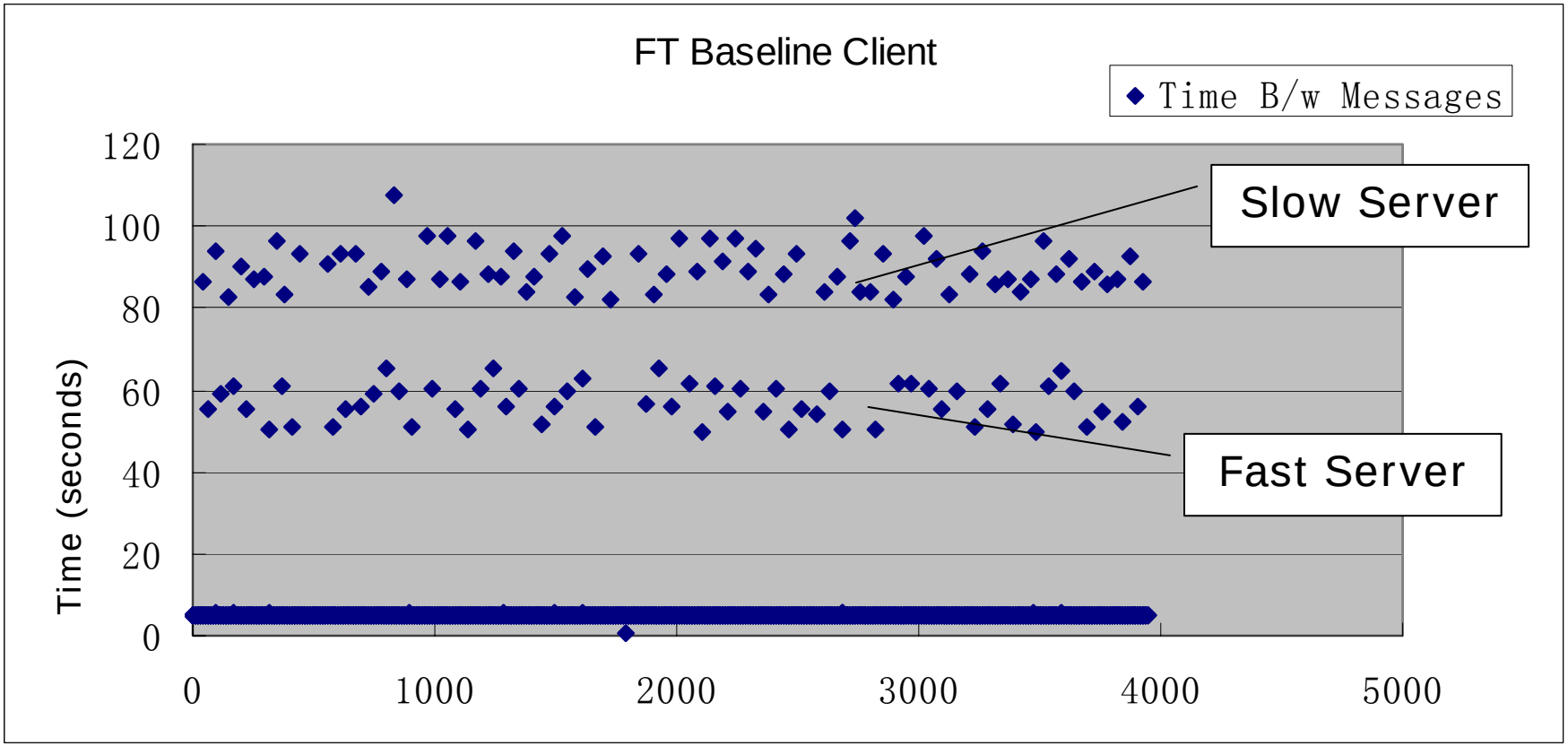
FT Baseline Architecture



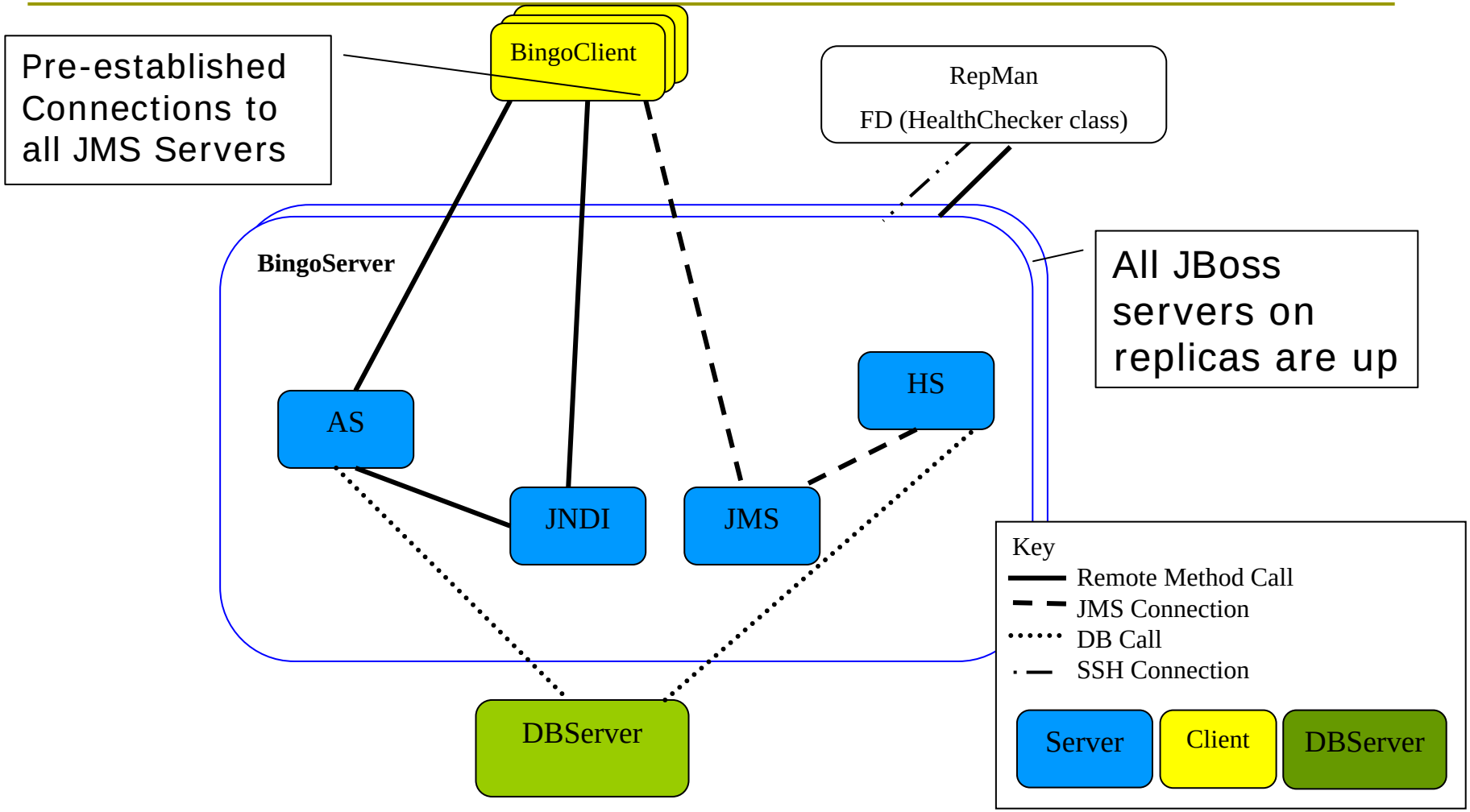
Baseline Fail-Over Measurements



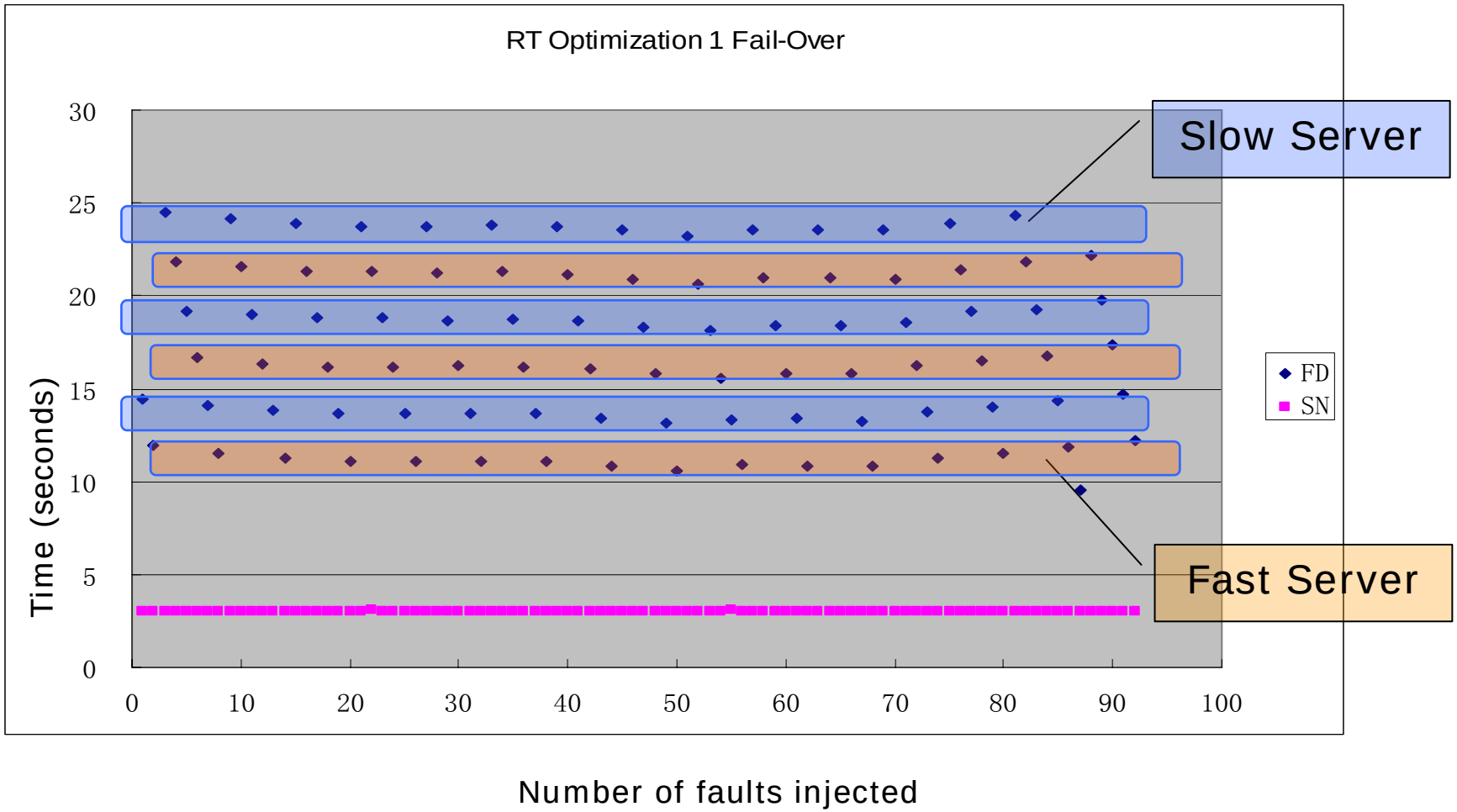
Baseline Fail-Over Measurements



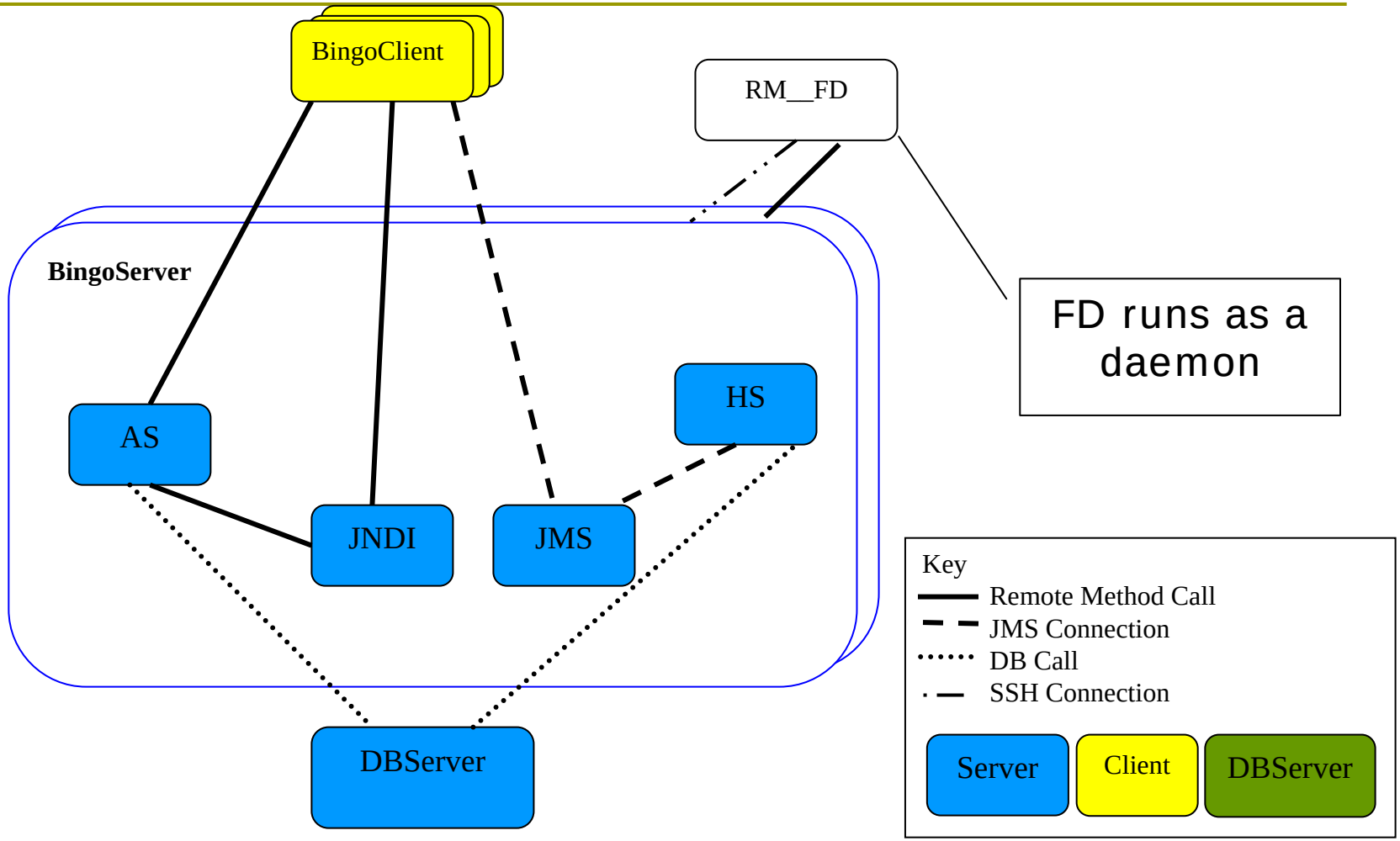
RT-FT Optimized 1 Architecture



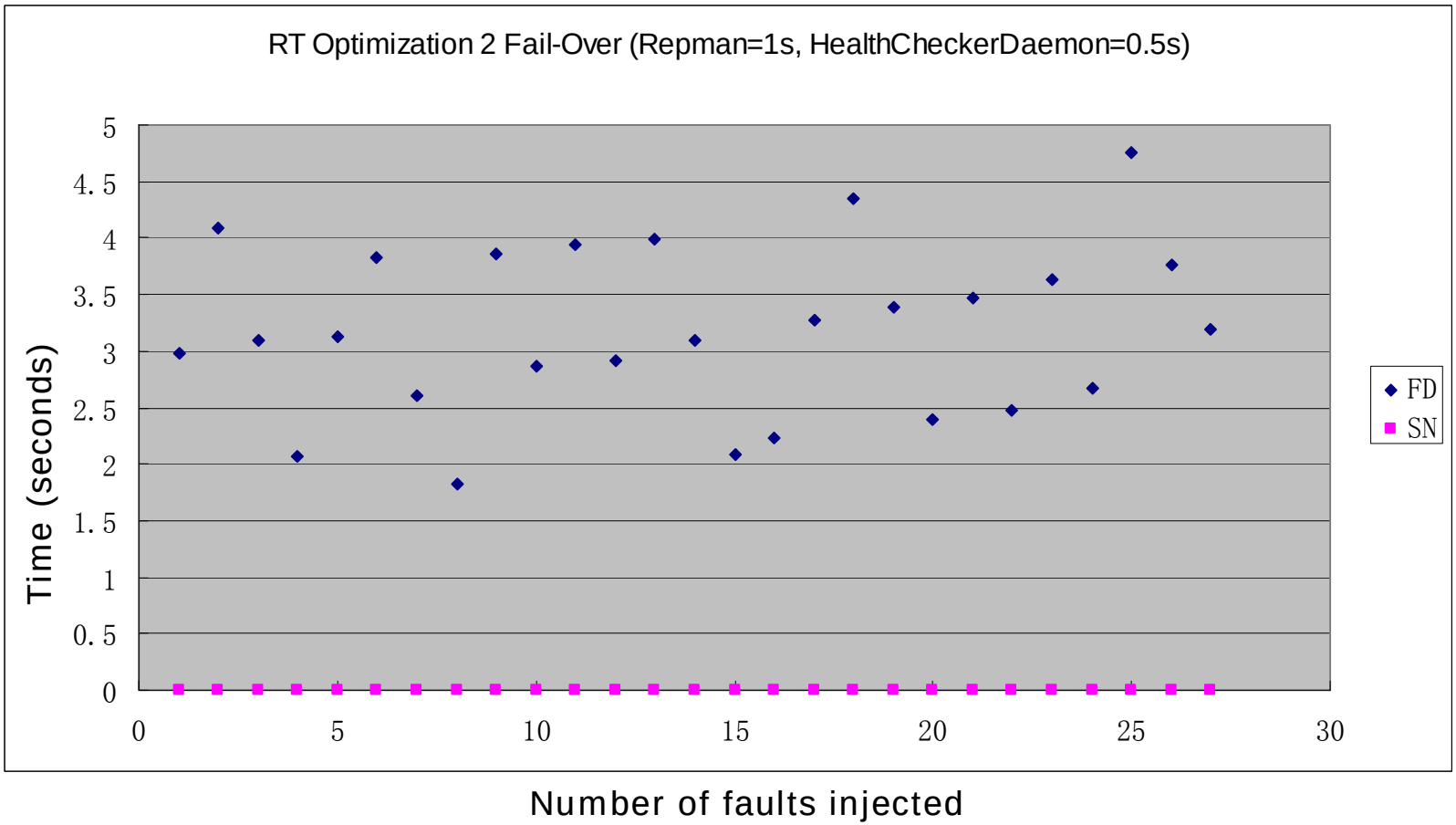
RT Optimized 1 Fail-Over Measurements



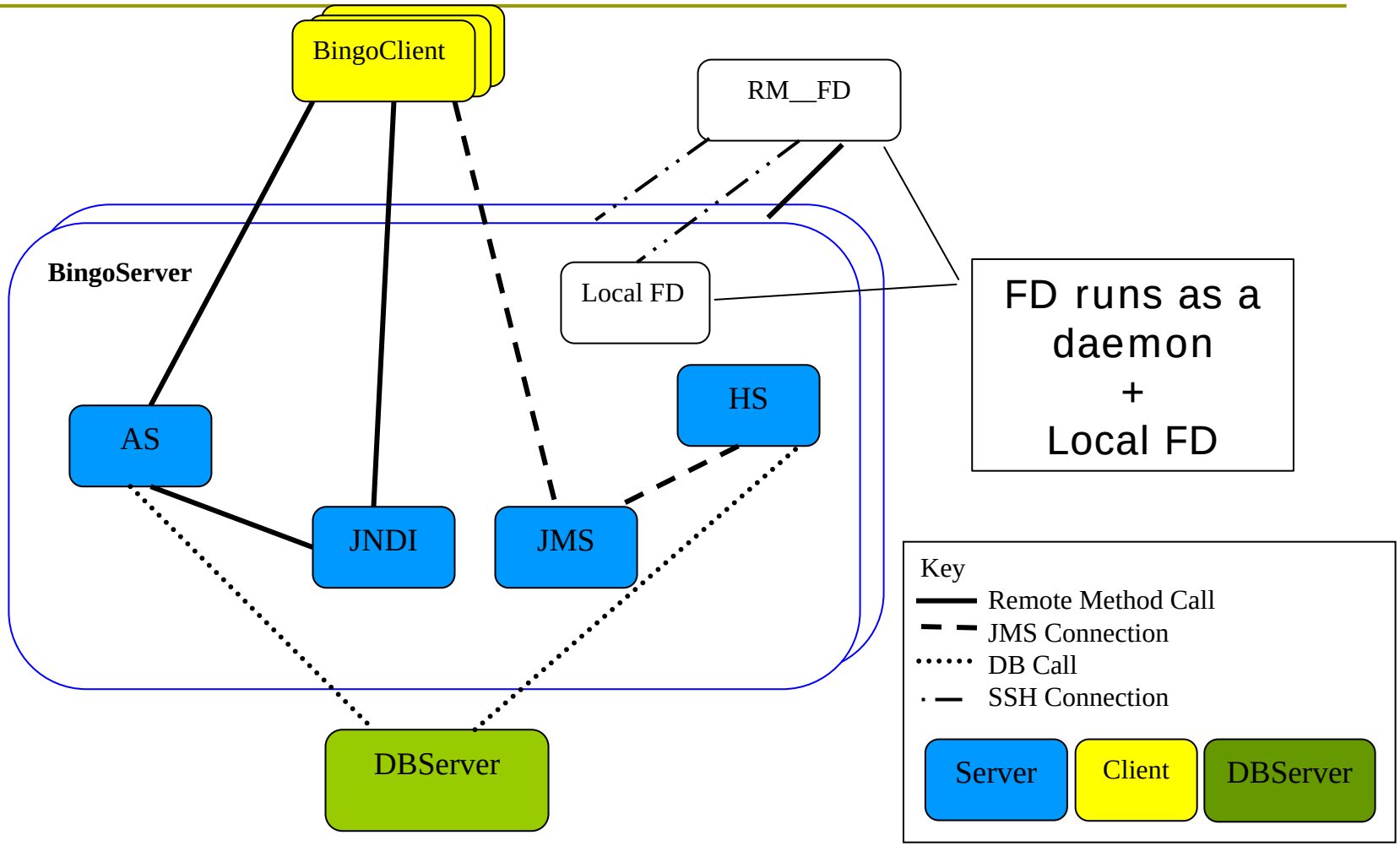
RT-FT Optimized 2 Architecture



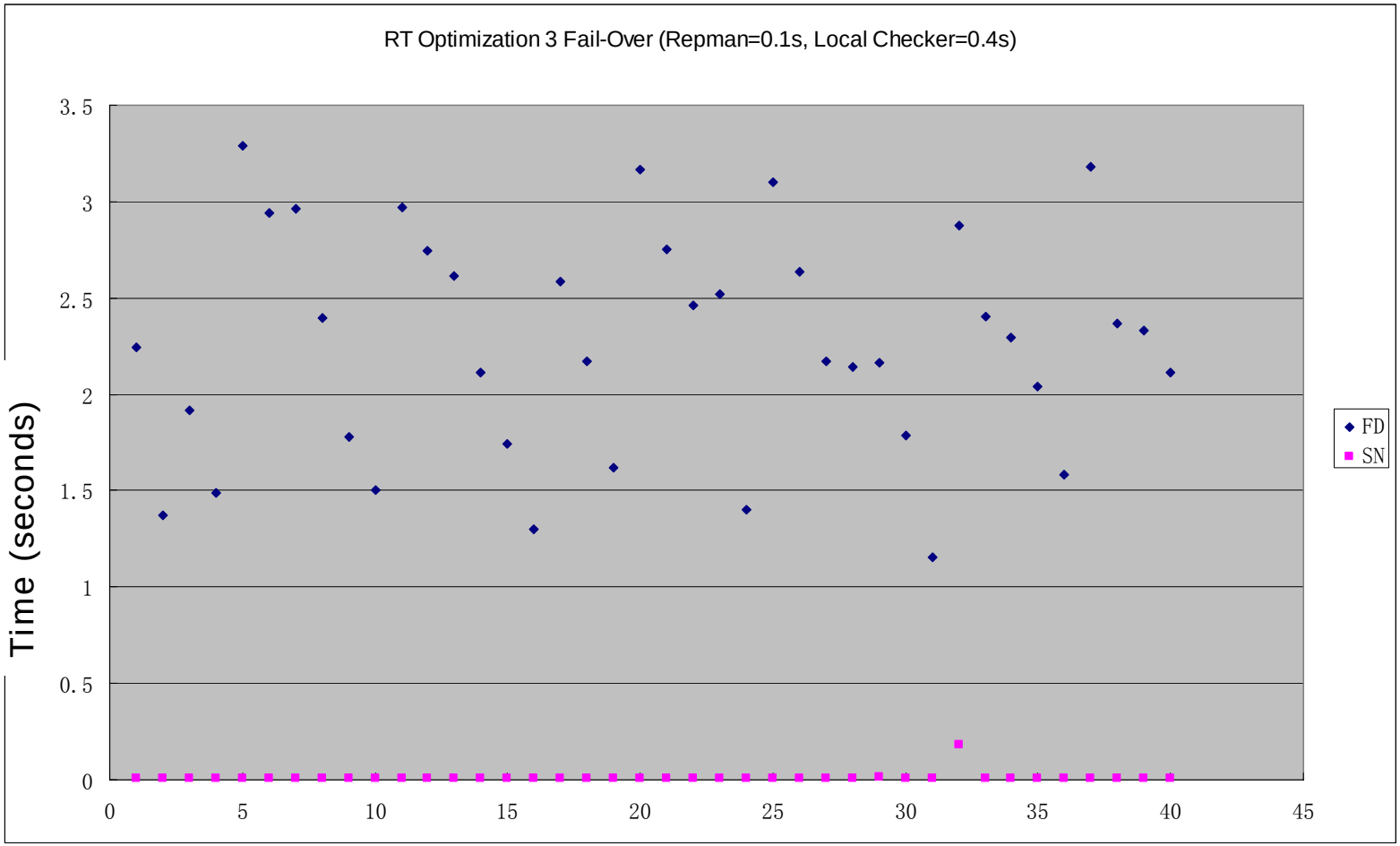
RT Optimized 2 Fail-Over Measurements



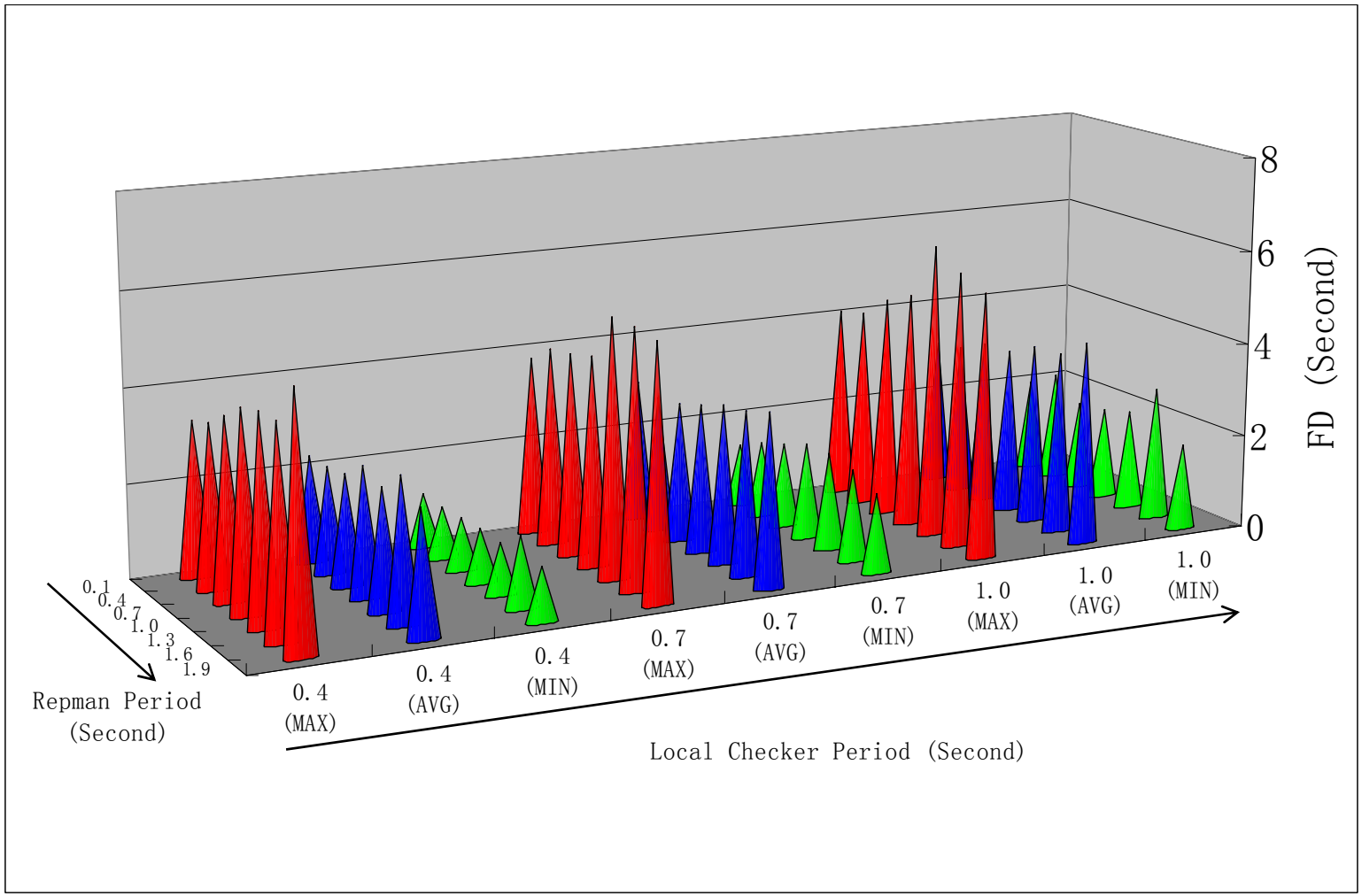
RT-FT Optimized 3 Architecture



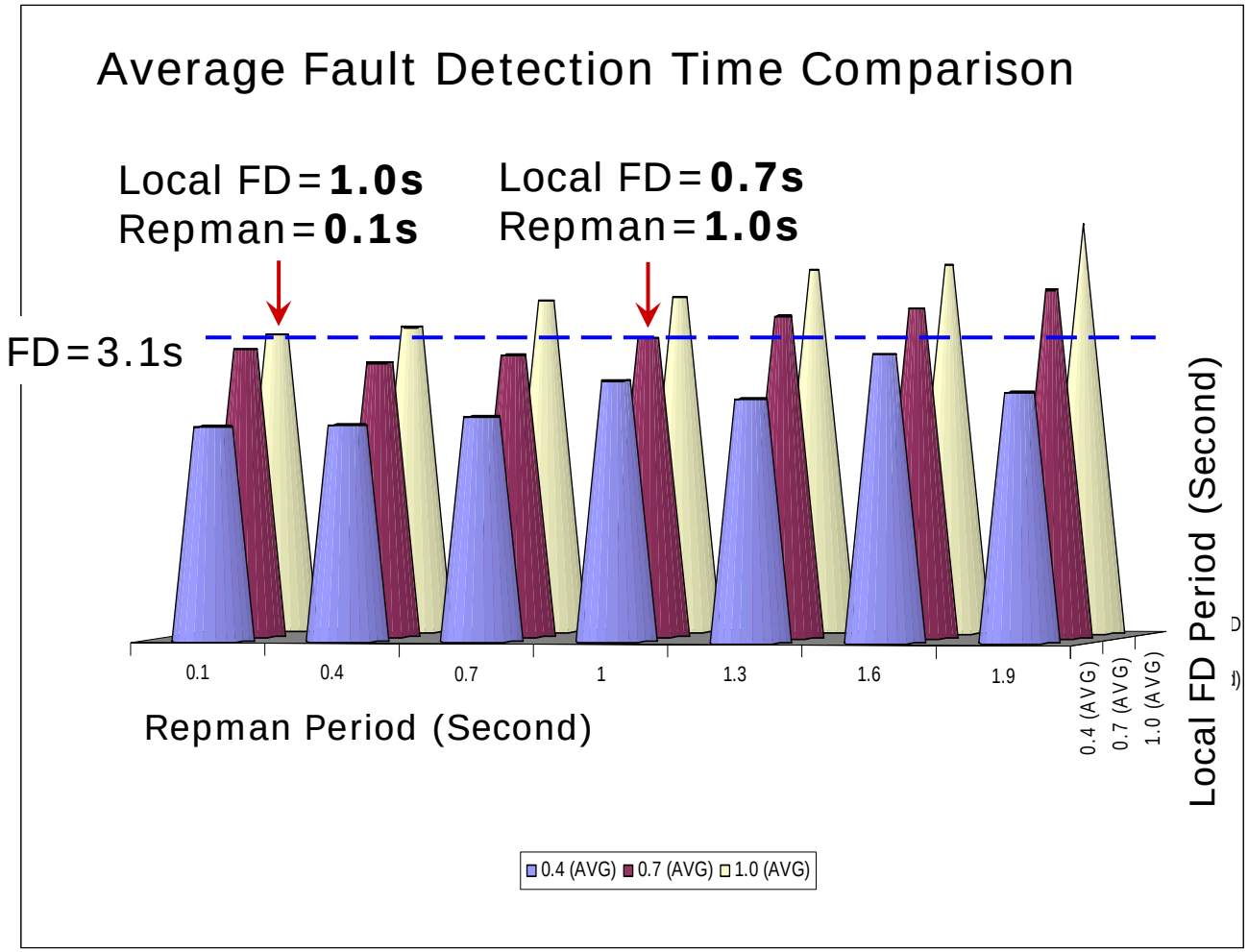
RT Optimized 3 Fail-Over Measurements



Measurements Insights (1)



Measurements Insights (2)

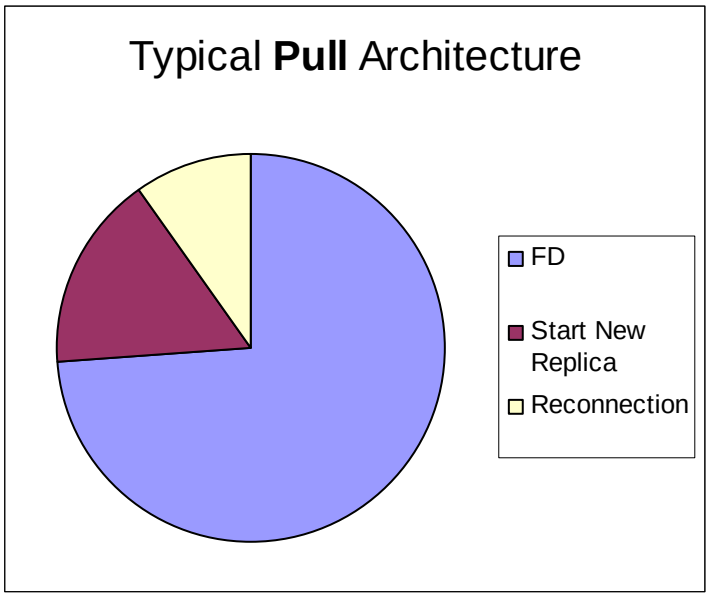


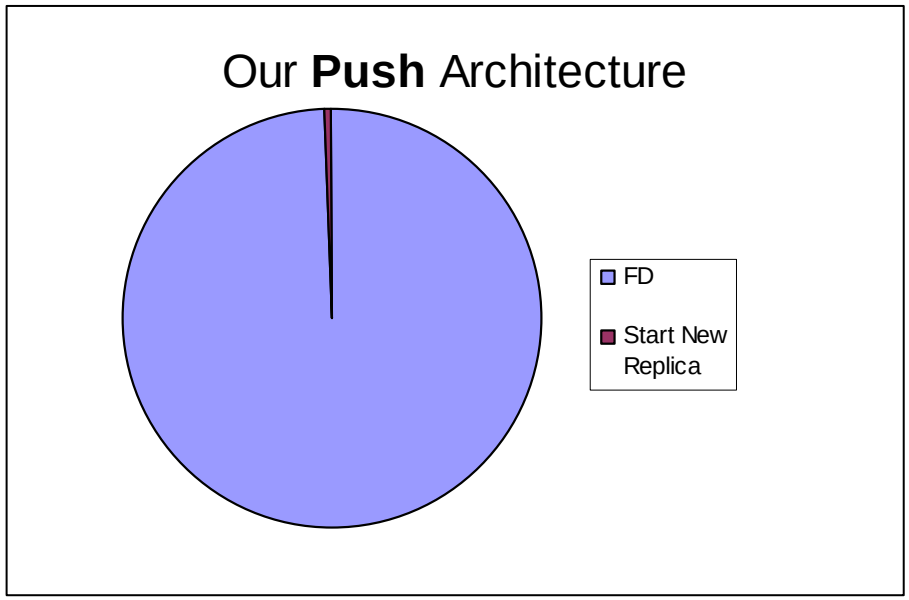
Real Time Tuning:

- Repman period
- Local FD period
- Broadcast period

Lessons Learned

- Potentially Publish/Subscribe (JMS) can hide server errors from clients





- JBoss Server should be run in the minimal configuration. (default configurations are not suited for RT)

Questions?