Group 6: Team Slackers

Design Document for Fault Tolerant Park'n Park

The fault Tolerance design consists of

1. 1 Primary and 2 replicas.
2. Replication manager, Fault Manager and Naming service as one Machine.
3. Database as one machine.
4. Various clients on different machines.

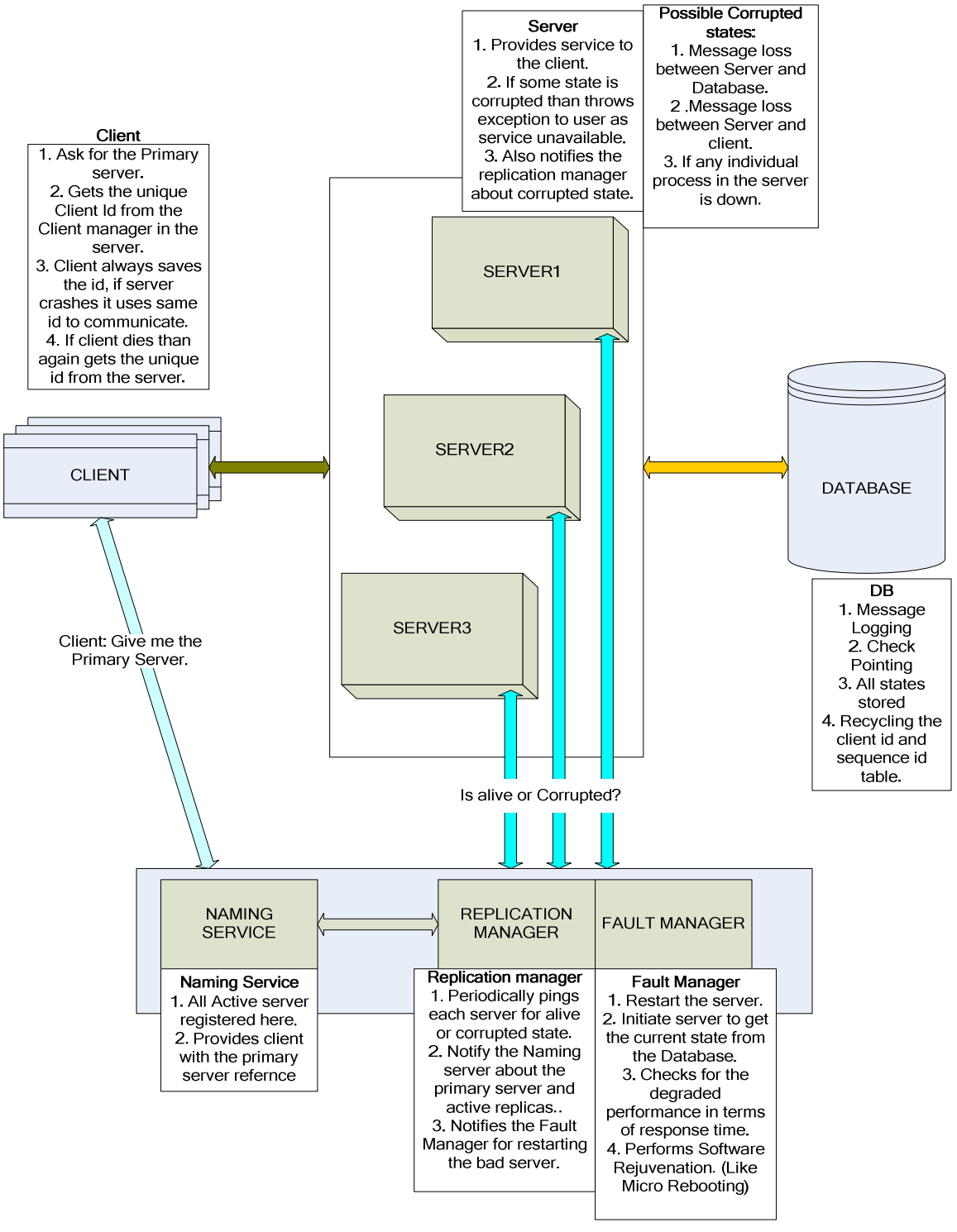
The faults being handled

1. Hardware and OS crashes for servers as well as client.
2. Message Loss between Server-Database connection
3. Message Loss Between Server-Client connection
4. Any individual process crash with in the server.

Statistics of fault handling

1. Two Simultaneous Faults can be tolerated.
2. Crash Crash Recovery Crash Recovery Recovery Crash Crash Model can work.
3. No more than 2 faults until the Recovery time (need to be figured out) of the Server.

Architecture is as follows (next page)


Server
1. Provides service to the client.
2. If some state is corrupted than throws exception to user as service unavailable.
3. Also notifies the replication manager about corrupted state.
Possible Corrupted states:
1. Message loss between Server and Database.
2 .Message loss between Server and client.
3. If any individual process in the server is down.
Client
1. Ask for the Primary server.
2. Gets the unique Client Id from the Client manager in the server.
3. Client always saves the id, if server crashes it uses same id to communicate.
4. If client dies than again gets the unique id from the server.
SERVER1
SERVER2
SERVER3
CLIENT
DATABASE
DB
1. Message Logging
2. Check Pointing
3. All states stored
4. Recycling the client id and sequence id table.
Client: Give me the Primary Server.
Is alive or Corrupted?
NAMING SERVICE
REPLICATION MANAGER
FAULT MANAGER
Naming Service
1. All Active server registered here.
2. Provides client with the primary server refernce
Replication manager
1. Periodically pings each server for alive or corrupted state.
2. Notify the Naming server about the primary server and active replicas..
3. Notifies the Fault Manager for restarting the bad server.
Fault Manager
1. Restart the server.
2. Initiate server to get the current state from the Database.
3. Checks for the degraded performance in terms of response time.
4. Performs Software Rejuvenation. (Like Micro Rebooting)

Working

Hardware faults and OS crashes (Server)

1. Replication manager pings each server to ask Is Alive? or corrupted?
2. If doesn't get a response back with in timeout limit (~1sec) than switches over to next replica.
3. Replication manager notifies the fault manager to restart the Server and bring it back.
4. Replication manager notifies the Naming service about the next primary.
5. Client gets the information about the crash via time out (~2-3sec) and than queries naming server for new primary.
6. Client uses its Id which it got from the first primary server to send the message and which is checked by the second primary (through database) in order to service it.
7. Consistency and determinism is maintained as everything is logged in the Database and the replica can get the current state from there.

Hardware faults and OS crashes (Client)

1. Client brings up its own (Our system doesn't handle restarting it).
2. But when it again communicates it is assigned a new client id and the normal operation resumes.

Message Loss between Server and Database

1. Server makes 3 tries to get connected to database if couldn't get connected than
    a. Notifies client that service not available.
    b. Notifies the replication manager that something is corrupted (It can be specific information also like couldn't get connected to database).

2. Replication manager assigns new primary and notifies naming service.
3. As client gets service unavailable it queries naming service to get the new Primary.
4. Fault manager restarts the first primary (Or can ping the database server to check whether it is alive or not)
5. If the problem is in database server than certain action can be taken (TBD).

Message Loss between Server and Client

1. If the server is down than the CORBA throws exception and client can switch over to other server.
2. If there is message loss than client sends the request again.
3. If three attempts fail than it switch over to next replica.

Role of Replication Manager

1. Periodically (every 1 sec) pings each server querying IsAlive & Corrupted or not?
2. If the response of any of these is negative than
    a. Activates next replica.
    b. Registers the replica as primary.
    c. Restarts the corrupted or dead server.
    d. Brings back in running mode and registers it again as replica.

Fault Manager (Can be implemented)

1. Checks the degrading performance of the system by monitoring the response time.
2. Or just to get the best out of the system can restart the process periodically (interval of a 1-2 hours demo purpose).
3. This is basically Software Rejuvenation.

Check pointing

1. Sever can put a check point in database that these requests have been serviced so that recycling script or new Server comes up doesn't have to look at the old entries.
2. All message Logging is done in the database so Server is like stateless.