Replication Models

1 Replication

- Provide multiple copies of the same data or functionalities (services) in a distributed system
- Improve system capabilities in terms of performance, availability and load distribution

2 Types and Requirements

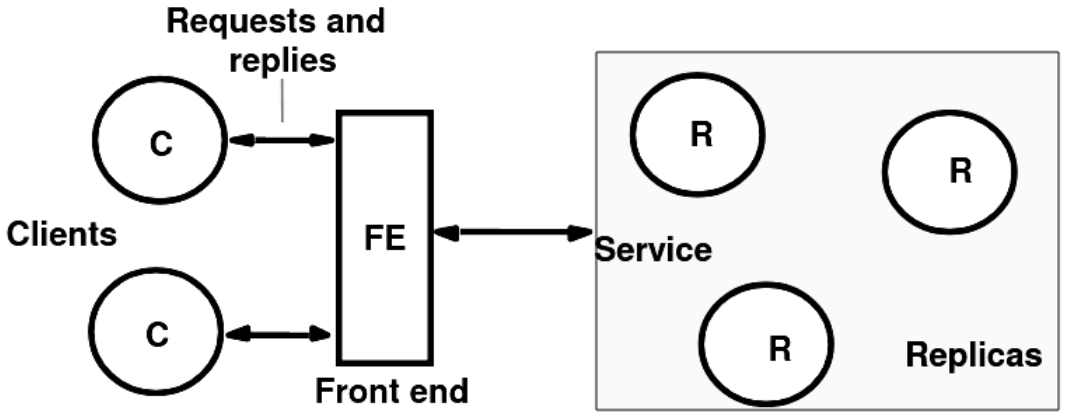
Types:

- Computation (function/service) replication: multiple instances of the same functional processes are executed (behaviour) - may run on different hardware and implement different algorithms
- Data replication: same piece of information is being stored in multiple devices (data integrity)

Requirements:

- Replication transparency: A user sees one logical service, but not its physical copies
- Data consistency: The same request will receive the same result even if it is processed by a different copy of the same service

3 System Model - Replication



Replicas:

- Maintain copies of the same data or functions - can be implemented by different technologies
- Replicas are not necessarily consistent all the time (some may have received updates, not yet conveyed to others)

4 System Components

Replicas (R)

- Maintain replicas (data/functions) on servers
- Process requests or store results (may propagate to other servers)
- Dynamic/static: set of Rs is fixed or variable (scalability issues)

Clients (C) requests

- Those without updates are called read-only requests, the others are called update requests (they may include reads)
- Read only: handle by one replica

- Update: may involve data propagation/synchronisation, and concurrency control

Front End (FE)

- Make replication transparent
- Monitor and maintain replica availability
- Perform request distribution, and collate responses
- Load balancing

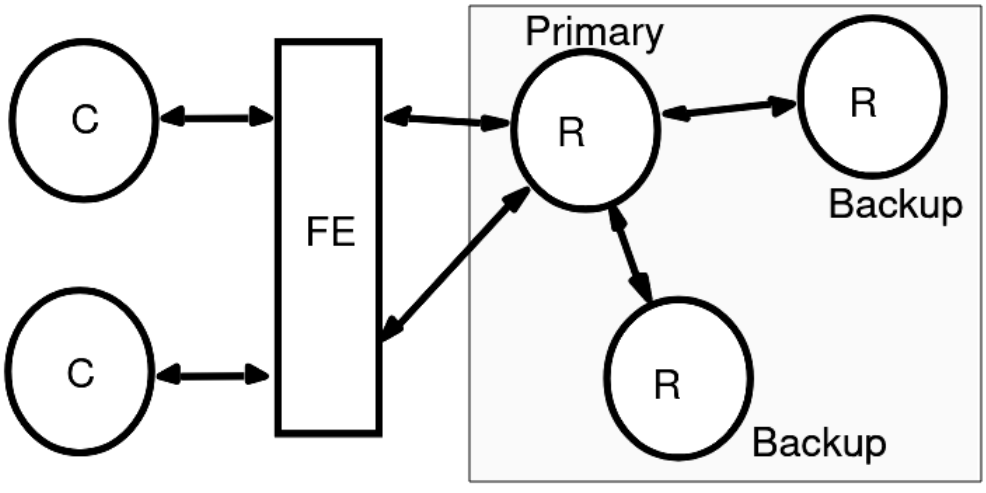
5 System workflow

1. Incoming request
   - Receive by the FE, and the FE will forward to the R(s)
2. Coordination
   - R(s) accepts a request
   - Decide the ordering request relative to other requests
3. Execution
   - R(s) process the request
4. Agreement
   - R(s) reach consensus on the effect of the requests
5. Response
   - One or more Rs reply to the FE
   - FE may process the response before returning it to the client

6 Fault-Tolerance Services

- Provide a correct service despite up to f process failures
- Each replica is assumed to behave according to the specification of the distributed system, when they have not crashed
- A service based on replication is correct if:
  - It keeps responding despite failures
  - Clients can't tell the difference between the service they obtain from an implementation with replicated data and one provided by a single correct replica manager

6.1 Passive (Primary-Backup) model for fault tolerance



- There is at any time a single primary R and one or more secondary (backup, slave) Rs
- FEs communicate with the primary which executes the operation and sends copies of the updates data to the result to backups
- If the primary fails, one of the backups is promoted to act as the primary

6.1.1 Workflow

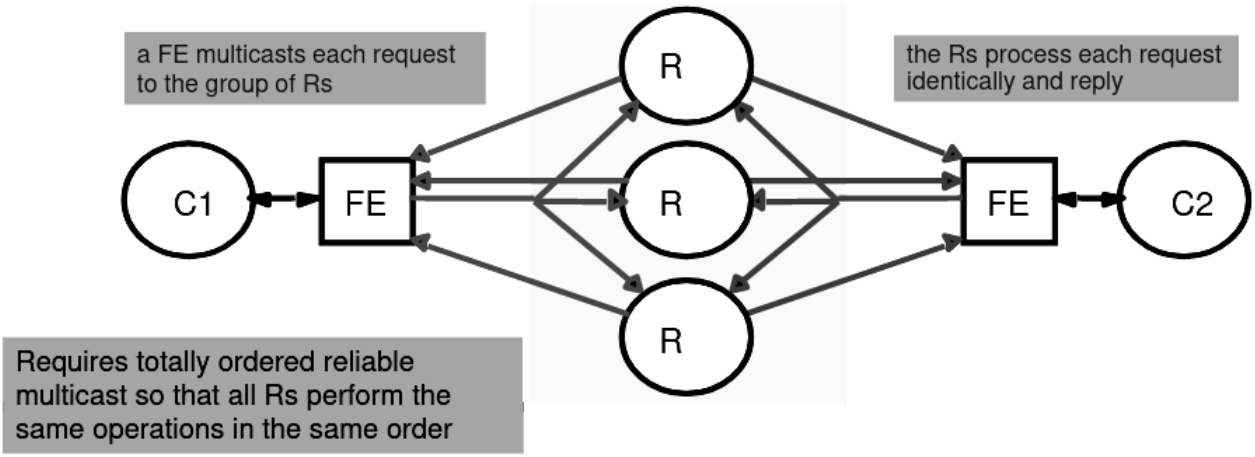
1. Request
   - An FE issues the request, containing a unique identifier, to the primary R
2. Coordination
   - The primary processes each request, in the order in which it received it relative to other requests (message ordering)
   - It checks the unique id; if it has already done the request, it re-sends the response (Handle Message loss)
3. Execution
   - The primary executes the request and stores the response
4. Agreement
   - If the request is an update the primary sends the updated state, the response and the unique identifier to all the backups. The backups send an acknowledgement (result propagation)
5. Response
   - The primary responds to the FE, which hands the response back to the client

6.1.2 Discussion of Passive Replication

- Non-deterministic behaviour at primary replica
  - e.g. due to multi-threading
  - No fatal problem: as other replicas (backups) only slavishly record states determined by the primary's actions
- Replica crashes
  - Survive up to f replica crash, when the system comprises $f + 1$ replicas
- Front-end functionality
  - Requires little functionality: only need to lookup a new primary replica when the current one isn't available
- System overhead
  - Relatively large due to data propagation

6.2 Active replication

- The Rs are state machines all playing the same role and organised as a group - all start in the same state and perform the same in the same order so that their state remains identical (synchronisation)
- If an R crashes it has no effect on performance of the service because the others continue as normal



Steps for client request

1. Request - uses a totally ordered reliable multicast - use TCP with multicasting to make reliable
2. Coordination - message ordering
3. Execution - every R executed the request
4. Agreement - none required as all Rs execute the same operations in the same order
5. Response - FEs collect responses from Rs

7 Byzantine Fault

Definition: Byzantine Fault

An arbitrary fault that occurs during the execution of an algorithm by a distributed system

This encompasses both

- Omission failure - crashes, failing to receive/send a request etc
- Commission failure - Incorrect processing

Definition: Byzantine failure-tolerant algorithms

Characterised by their resilience f, the number of faulty processes with which an algorithm can cope

Can mask up to f Byzantine faults if the system incorporates at least $2f+1$ replicas (just taking the mode of the results)