Distributed Systems (5DV147)

Replication

Fall 2014

Replication

What is it?

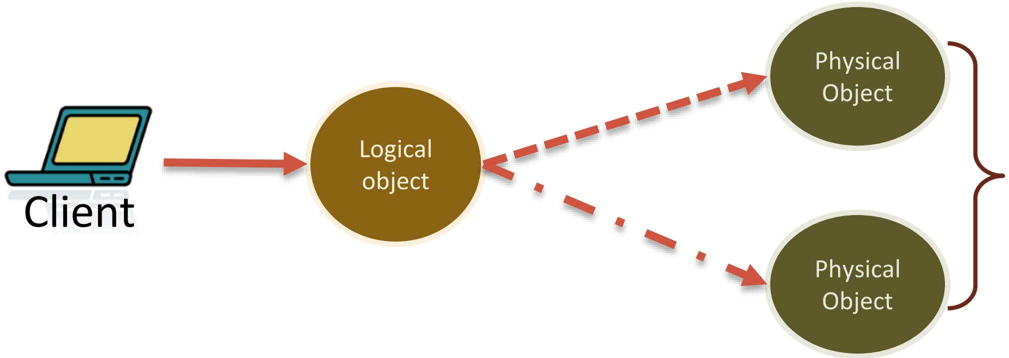
- Make multiple copies of a data object and ensure that all copies are identical
  - Immutable data – trivial
  - Often updated data – tricky, can be expensive to maintain copies identical
- Two types of access
  - Read
  - Write (update)

Why would we use replication?

- Reliability
  - Fault tolerance (redundancy, $f + 1$ servers crash)
  - Protection against corrupted data (majority)
  - Availability (server failures, network partitions)
- Performance
  - Scalability (divide the work)
  - Load balancing
  - Reducing access latency (data closer to process)
    - Caching

Replication requirements

- ❑ Transparency (illusion of a single copy)
  - ➢ Clients must be unaware of replication
- ❑ Consistency
  - ➢ Obtain identical results from different copies



Not always identical:

- Some have received updates

Problems that you may find

- Multiple clients access replicas
  - Concurrent access, rather than exclusive
  - Operations are *interleaved*
    - How do we ensure correctness?
- Replica placement
  - Placing servers
  - Placing content
- Overhead required to keep replicas up to date
  - Global synchronization (Atomic operations)
    - Relaxed atomicity constraint, but copies will not always be the same
      - Depends on access and update patterns of data

System Model

Basic architectural model

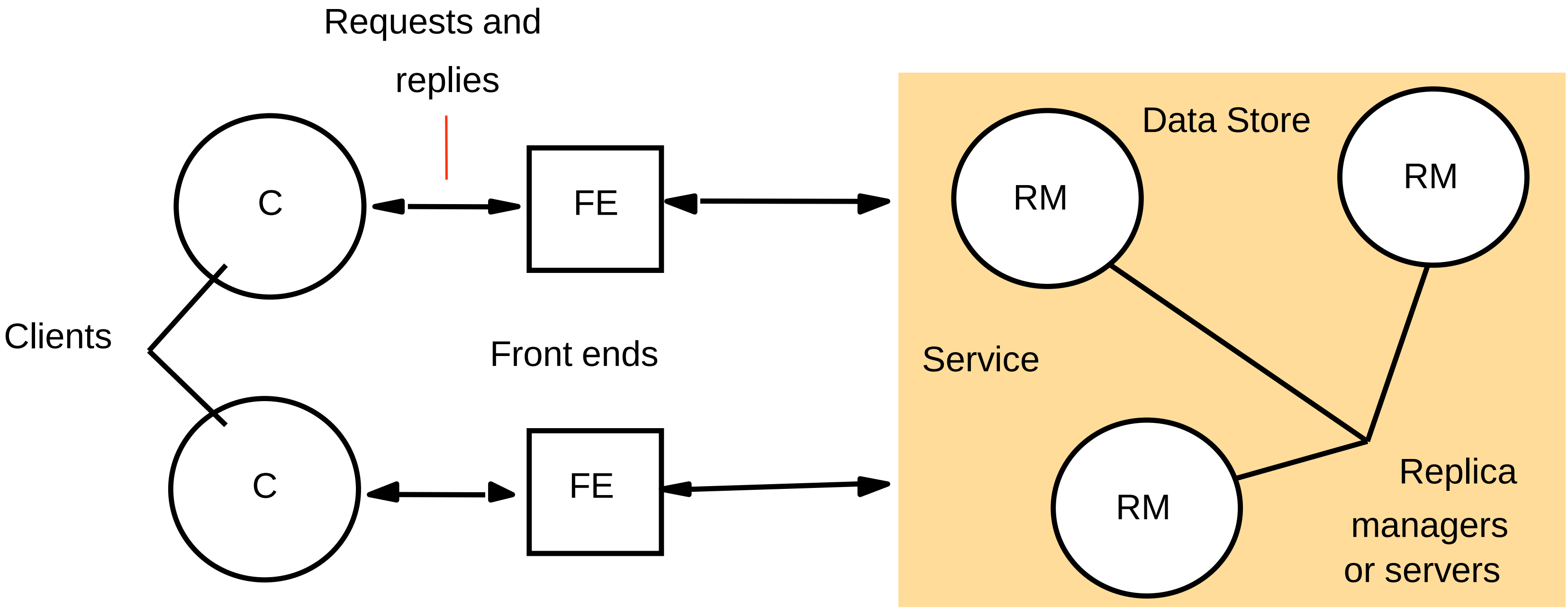


Figure adapted from Instructor's Guide for Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5 © Pearson Education 2012 – based on **Figure 18.2**

General replication phases

- Request – client makes request
- Coordination – replica managers decide on execution of request and on order of request relative to others (the usual orderings)
- Execution – request is executed
- Agreement – replica managers agree on result of execution
- Response – response is sent back to the client

Types of ordering adapted to replication

FIFO– if a client issues **r** and then **r’**, any correct RM that handles **r’** handles **r** before it

Causal– if the issuing of **r** happened-before issuing **r’**, then any correct RM that handles **r’** handles **r** before it

Total – if a correct RM handles **r** before **r’**, then any correct RM that handles **r’** handles **r** before it

… group communication

Recap: Group views...

- Contain the set of group members at a given point in time
  - Dynamic groups
  - Failed identified processes are not in the view (failure detection)
- Events occur *in views*
- Views are delivered when membership changes
  - *Primary-partition* and *partitionable* groups
- View-synchronous group communication
  - Based on view delivery, we know which messages must have been delivered within a view

… and view delivery requirements

- Order
  - View changes always occur in the same order at all processes
- Integrity
  - If a process delivers a view then that process is part of that view
- Non-triviality
  - Processes that have joined a group and communicate indefinitely are members of the same group
  - Membership service should eventually reflect network partitions

View-synchronous group communication

- Correct processes deliver the same set of messages in any given view (agreement)
- Messages are delivered at most once and the sender is always in the group view in which the message is delivered (integrity)
- Correct processes always deliver messages they send (validity)
  - If delivering to *q* fails, the next view excludes *q*

Brief intro to correctness

Intuition

- A replicated system is correct when:
  - It maintains execution despite failures
  - Clients can't tell the difference between the results from a system that uses replicated data from those obtained from a system with a single correct replica

Example

Client 1	Client2
$setBalance_B(x,1)$	
$setBalance_A(y,2)$	
	$readBalance_A(y) \rightarrow 2$
	$readBalance_A(x) \rightarrow 0$

- Local replica of Client 1 is B
- Local replica of Client 2 is A

Correctness of interleaving

- "Basic" correctness property
  - An interleaved sequence of operations must meet the specification of a single correct copy of the object(s)
- Sequential consistency property
  - Order of operations is consistent with the program order in which each individual process executed them
- Linearizability property
  - Order of operations is consistent with the real times at which the operations occurred during execution

Example of interleaved operations for 2 clients:

C1: A, B, C

C2: d, e, f

Real Order during execution: A, B, d, C, e, f

An interleaving with sequential consistency:

A, B, d, e, f, C

Interleaving with linearizability:

A, B, d, C, e, f

Models of replication

Passive replication

Passive (primary-backup) replication

- One *primary* replica manager, many backup replicas
  - If primary fails, backups can take its place (election!)
- Implements linearizability if:
  - A failing primary is replaced by a unique backup
  - Backups agree on which operations were performed before primary crashed
- View-synchronous group communication!

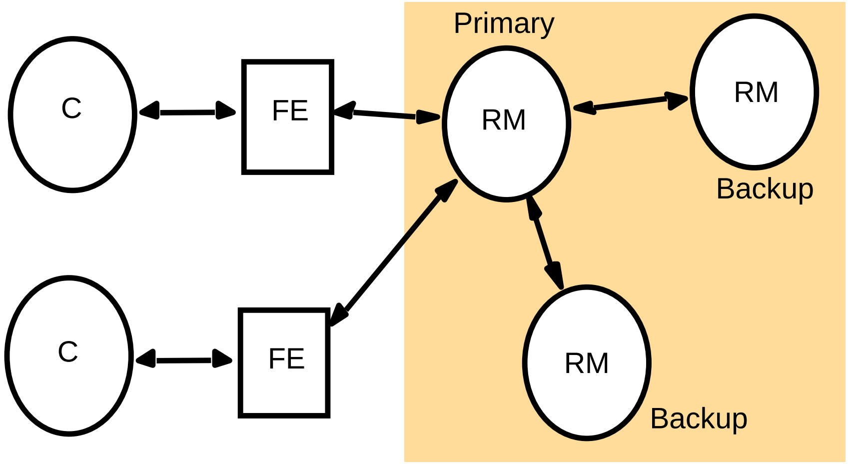
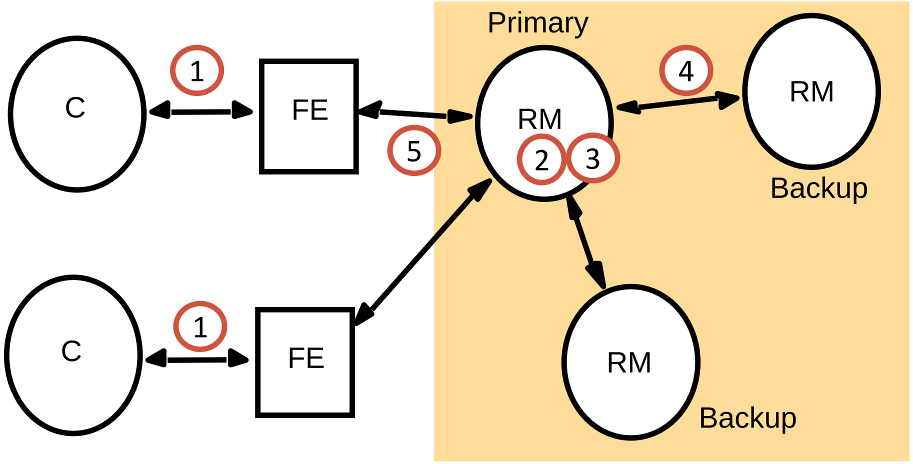


Figure adapted from Instructor's Guide for Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5 © Pearson Education 2012 – based on **Figure 18.3**

Steps of passive replication

1. Request
   – Front end issues request with unique ID
2. Coordination
   – Primary checks if request has been carried out, if so, returns cached response
3. Execution
   – Perform operation, cache results
4. Agreement
   – Primary sends updated state to backups, backups reply with Ack.
5. Response
   – Primary sends result to front end, which forwards to the client



What happens if the primary RM crashes?
- ❑ Before agreement
- ❑ After agreement

Active replication

Active replication

- RMs play equivalent roles
- All replica managers carry out all operations
- Front ends multicast one request at a time (FIFO)
- Requests are totally ordered
- Implements sequential consistency
- Tolerate Byzantine failures

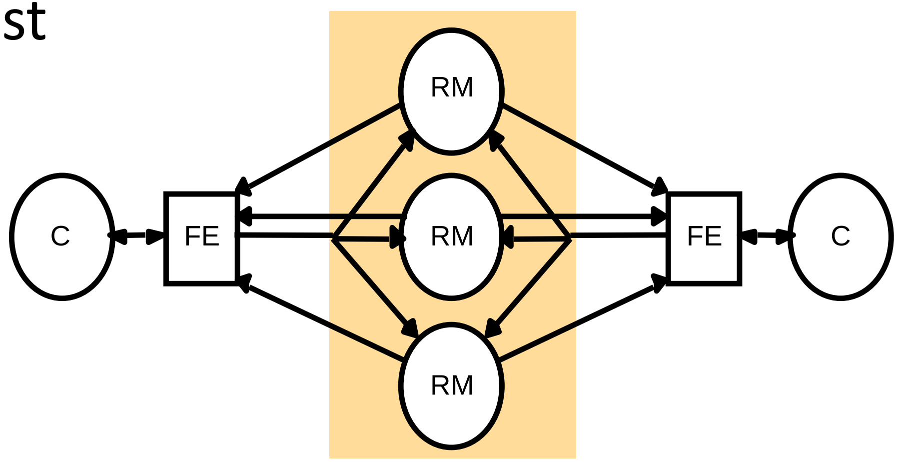
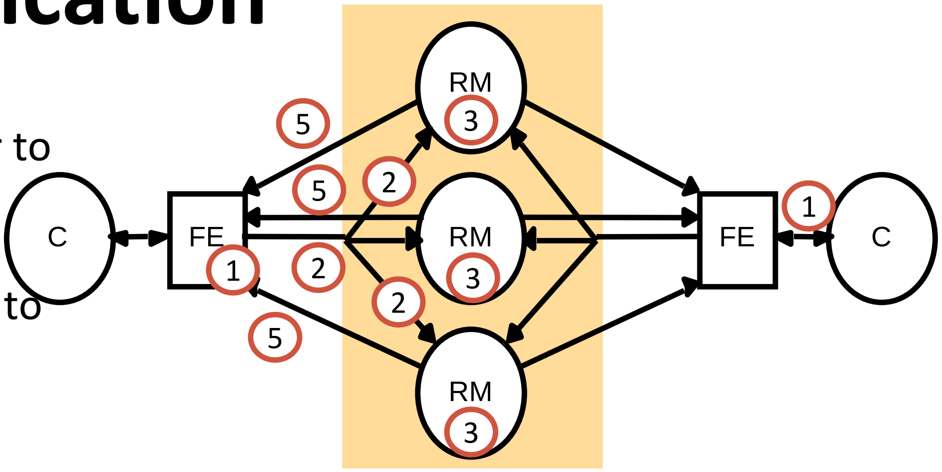


Figure adapted from Instructor's Guide for Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5 © Pearson Education 2012 – based on **Figure 18.4**

Steps of active replication

1. Request
   - ❑ Front end adds unique identifier to request, multicasts to RMs
2. Coordination
   - ❑ Totally ordered request delivery to RMs
3. Execution
   - ❑ Each RM executes request
4. Agreement
   - ❑ Not needed
5. Response
   - ❑ All RMs respond to front end, front end interprets response and forwards response to client



Comparing active and passive replication

- Both handle crash failures (but differently)
- Only active can handle arbitrary failures
- Passive may suffer from large overheads
- Optimizations?
  - Send “reads” to backups in passive

    Lose linearizability property!
  - Send “reads” to specific RM in active

    Lose fault tolerance
  - Exploit commutativity of requests to avoid ordering requests in active

Replica management

Placement

- Placement of replica servers
  - Find the best location for a server
  - Optimization: pick *k* sites among *N* potential sites ($k<N$) (NP-hard)
    - Tree-based algorithms, random assignments, hot spots, greedy algorithms, graph-based
- Placement of content
  - Finding the best server to hold certain content

Placement of content

- Permanent replicas
  - Initial replicas for distribution and mirroring replicas
- Server-initiated replicas
  - Initiated by the RM to cope with load and enhance performance (temporary replicas)
  - Access counts per object: delete and replication thresholds, migration of replicas
- Client-initiated replicas
  - Temporarily stored replica at the client (caches)

Update propagation

- Only propagate a notification of an update
  - Invalidation protocols
  - Good for low read-to-write ratios (r<<w)
- Transfer data from one copy to another
  - Good for high read-to-write-ratios (r>>w)
- Propagate the update operation to other copies
  - Active replication
- Push-based protocols
  - Updates propagation initiated by server (good for high read-to-write ratios)
- Pull
  - Client requests server to send updates (good for low read-to-write ratios)

Summary

- ❑ What is replication and why is it necessary
- ❑ System model
  - ➢ Basic architecture and replication phases
    - Request, coordination, execution, agreement, response
  - ➢ Types of ordering adapted to replication: FIFO, Causal, Total
  - ➢ Role of group communication
- ❑ Correctness of interleaving
  - ➢ Basic, sequential consistency, and linearizability

- Models of replication
    - Passive replication
        - A single primary replica manager and one or more backup replica managers
        - Implements linearizability
    - Active replication
        - Independent replica managers executing all operations
        - Implements sequential consistency
- Replica management
    - Placement of servers and of content
    - Updates propagation
        - Update notifications, data, or operations, and push vs. pull

Next Lecture

Consistency