Replication

Replication

- **Replication** = Maintenance of data copies at multiple computers
- **Replication** = Technique to increase the system efficiency:
  - Performance enhancement
  - Increased availability
  - Fault Tolerance

Replication / Performance enhancement

- Data caching at clients = avoid latency of fetching resources
- Data replication between several servers = distribution of workload
  - Proxy Server = Several servers have the same name
    - Ex : DNS returns different Server's IP for requests
    - Ex : using group communication
  - Proxy Server distributes the workload between the worker servers
- Protocol to ensure that clients receive up-to-date data.

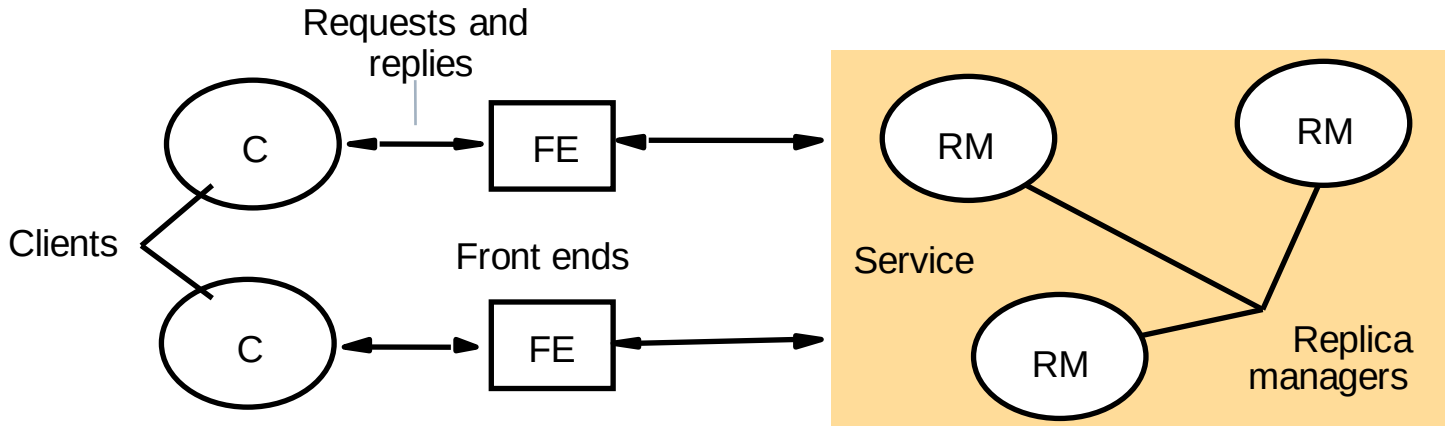
Replication / Increased availability

- Increased availability :
  - Replication = technique for maintaining data in case of:
    - Disconnected operation (mobile users)
    - Server failures & network partitions
  - $Av = 1 - P^n$
    - P = Probability of failing
    - N = Nb of servers

Replication / Fault Tolerance

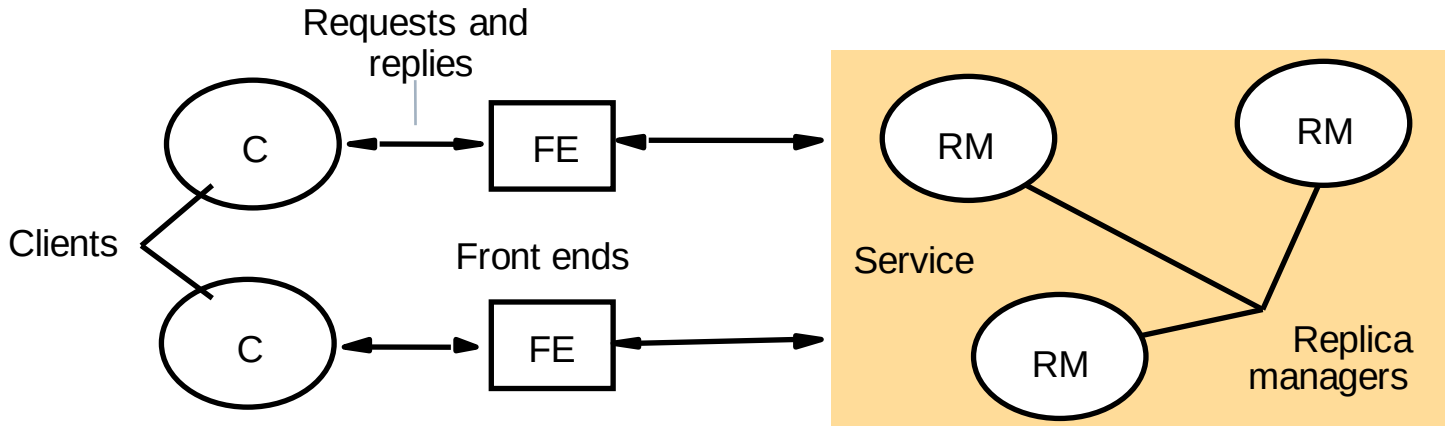
- Fault Tolerance :
  - Replication = used to guarantees *correct behavior* over server failures
    - Nb of total servers = 1+ 2 Failing servers
  - System state consistency must be maintained = coordination between the system components
  - Replication transparency = clients don't know of multiple copies (organized as **logical objects**)

Fault-Tolerant Components (by Replication)



- **FE** : process client requests
- (Replication Transparency) → **FE** isolates the client of **RM**

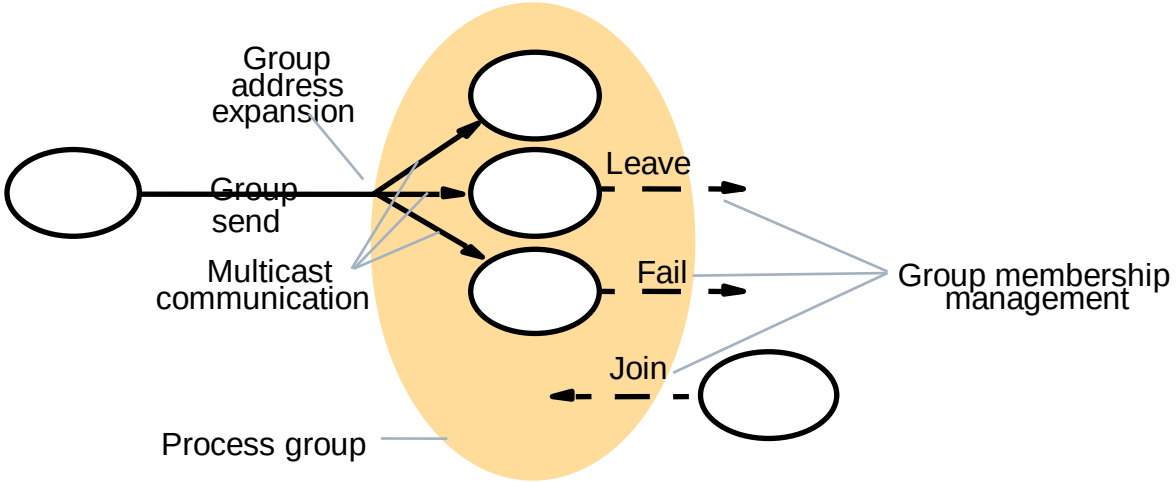
Replication based Fault-Tolerant Services



- **Request :** FEs give IDs to client requests
- **Coordination** : request ordering (FIFO/Causal/Total/…)
- **Execution** : tentatively (may be undo)
- **Agreement** : commit the request effects
- **Response** : by one or more replica managers

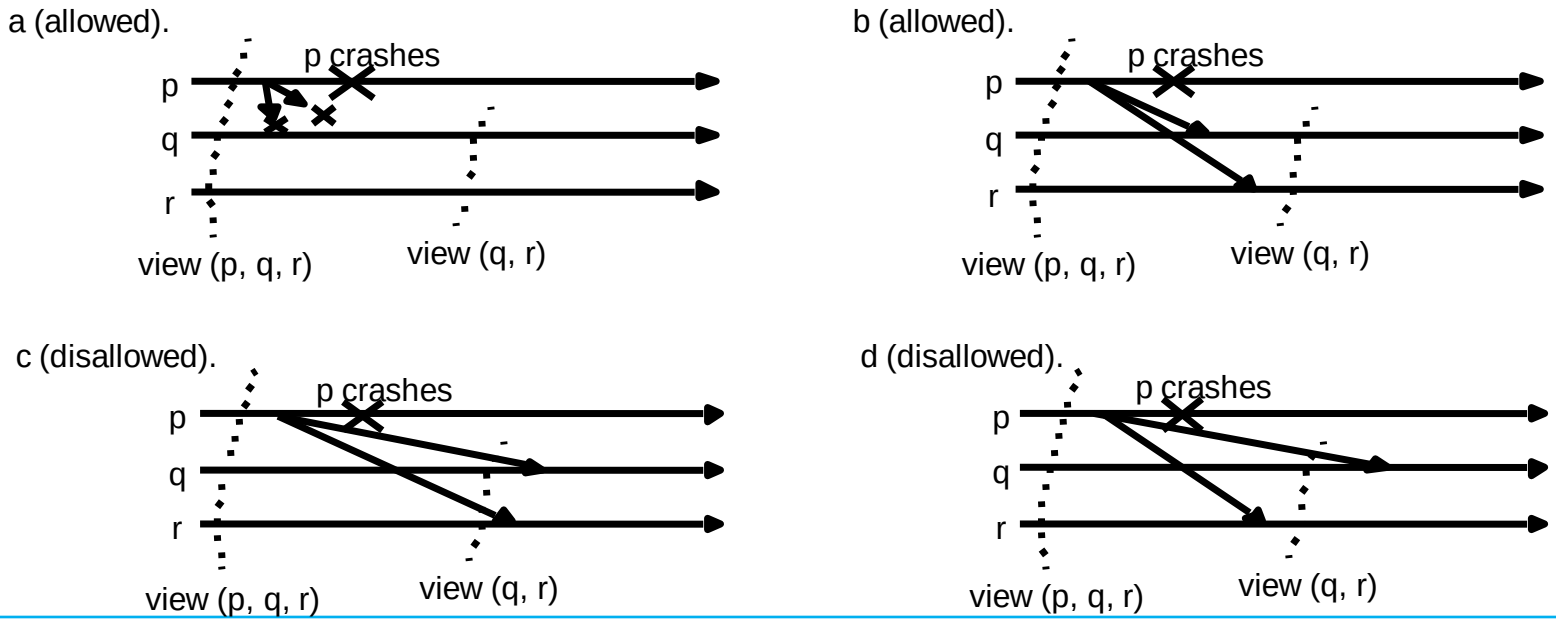
Replication based Fault-Tolerant Services / Group communications

- Group membership management = add / remove process
- Failure detector
- Notifying about Group member changes
- Group address = group id



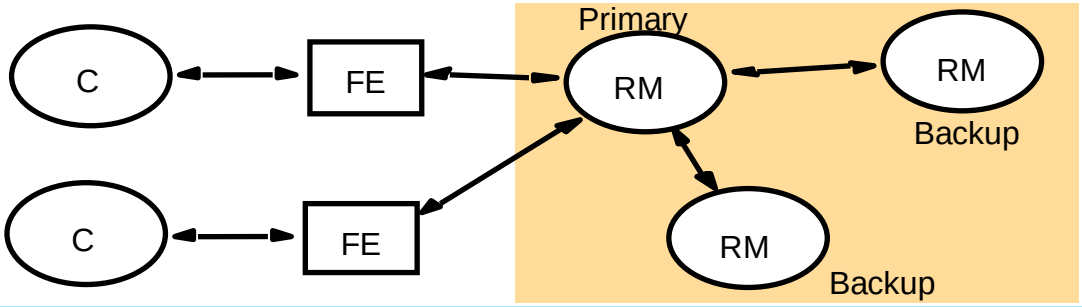
Replication Fault-Tolerant Services / View delivery & synchronization

- View delivery = multicast message that tell members about group membership change.
- Synchronization



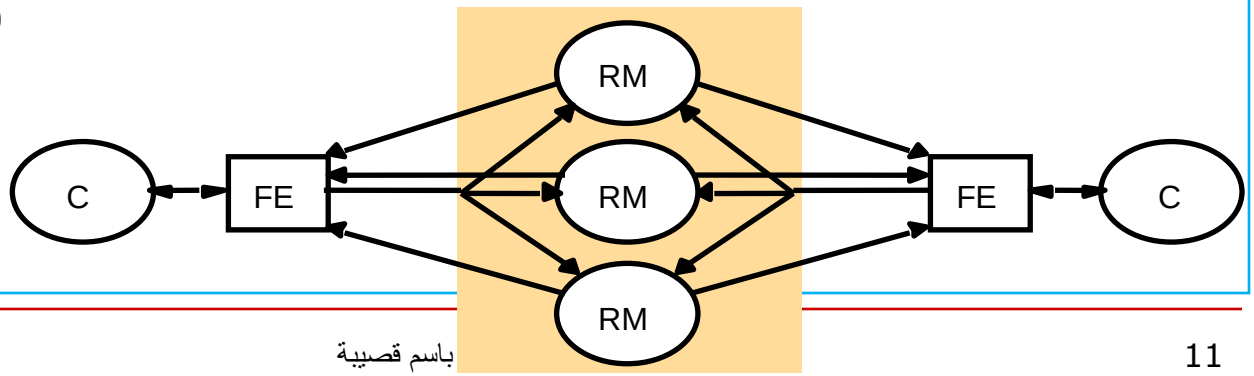
Fault-Tolerant Services
Passive Replication

- Primary RM and secondary RMs
  - FE send **request** (with ID)
  - **Coordination** : Primary RM checks the ID (execute request | re-send response)
  - Primary RM **executes** operations (in the receiving order)
  - sends updates & ID (**agreement**) to other RMs using view synchronization
  - sends a response to client when it gets all acknowledgments.
- Pri-RM crashes
  - Election
  - ID + View Syn



Fault-Tolerant Services
Active Replication

- All RMs play equivalent roles
  - FE multicast requests + ID to RMs (reliable & Total order multicast) and it waits the response before the next request.
    - In case of multi-threaded clients ➔ we use Causal Total order
  - Coordination : group communication delivers requests in total order
  - RMs execute request
  - No agreement
  - RMs send response to FE (FE returns the first or 1+ N/2 equivalent responses)

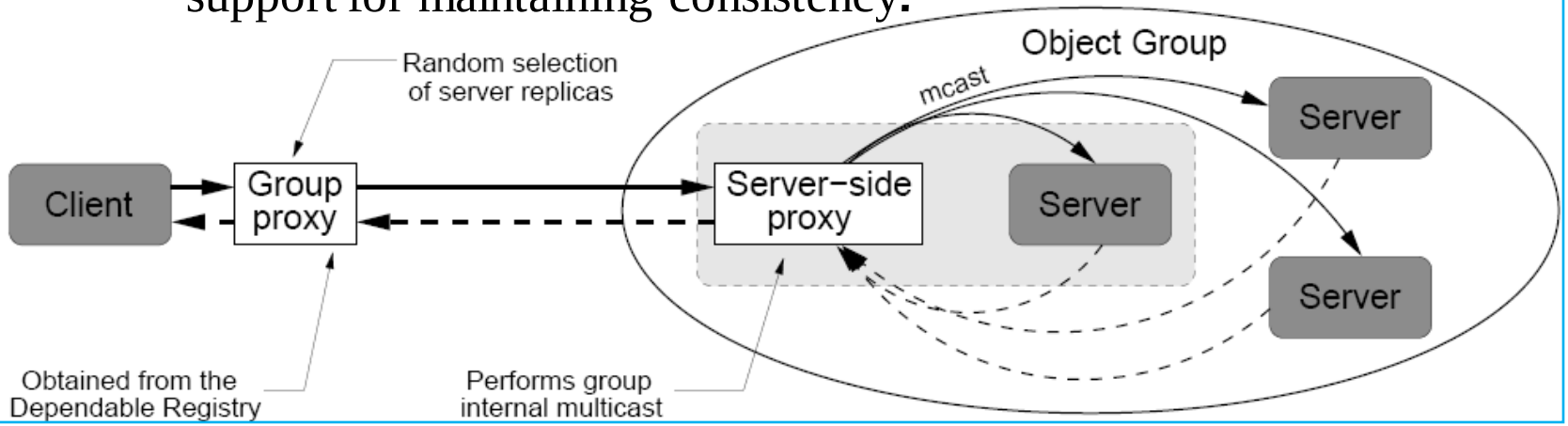


Case Study : JGroup

- Several object servers = object group (logical address)
- (**GMI**) Group Method Invocation = clients invoke methods on the object group
  - Method is executed by one or more of the group objects
- Clients interact with object group by EGMI (External Group Method Invocation)
- Servers interact with other servers by IGMI (Internal Group Method Invocation)
  - Goal = maintain the group state consistency
- State merging service
  - Maintain the consistency of all server states

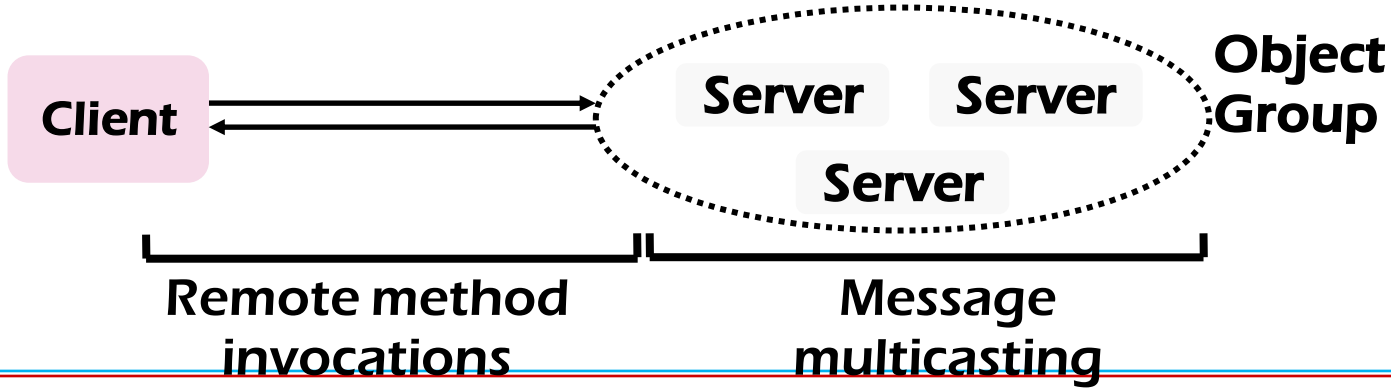
JGroup

- **JGroup** provides adequate support for developing reliable and high-available applications.
- **JGroup provides** reliable "one-to-many" interaction primitives
  - From the client's point of view:
    transparent access to replicated servers.
  - From the server's point of view:
    support for maintaining consistency.

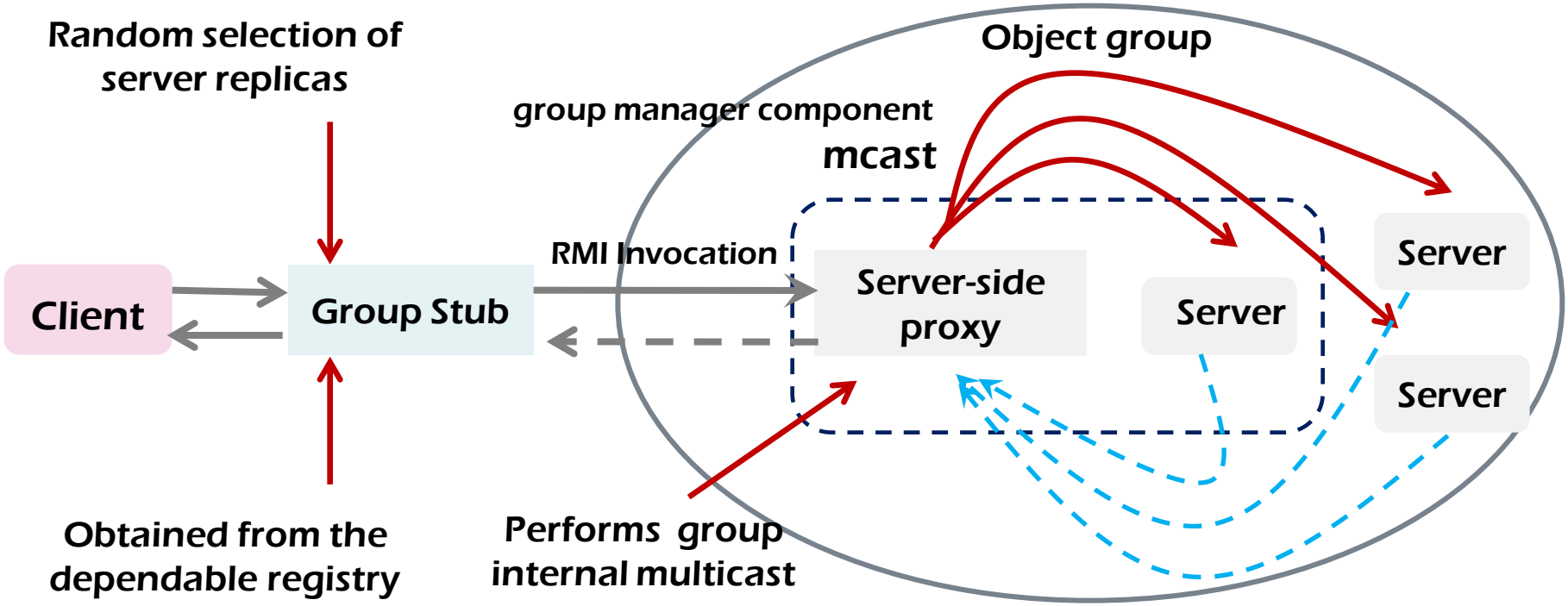


JGroup / Object Group Paradigm

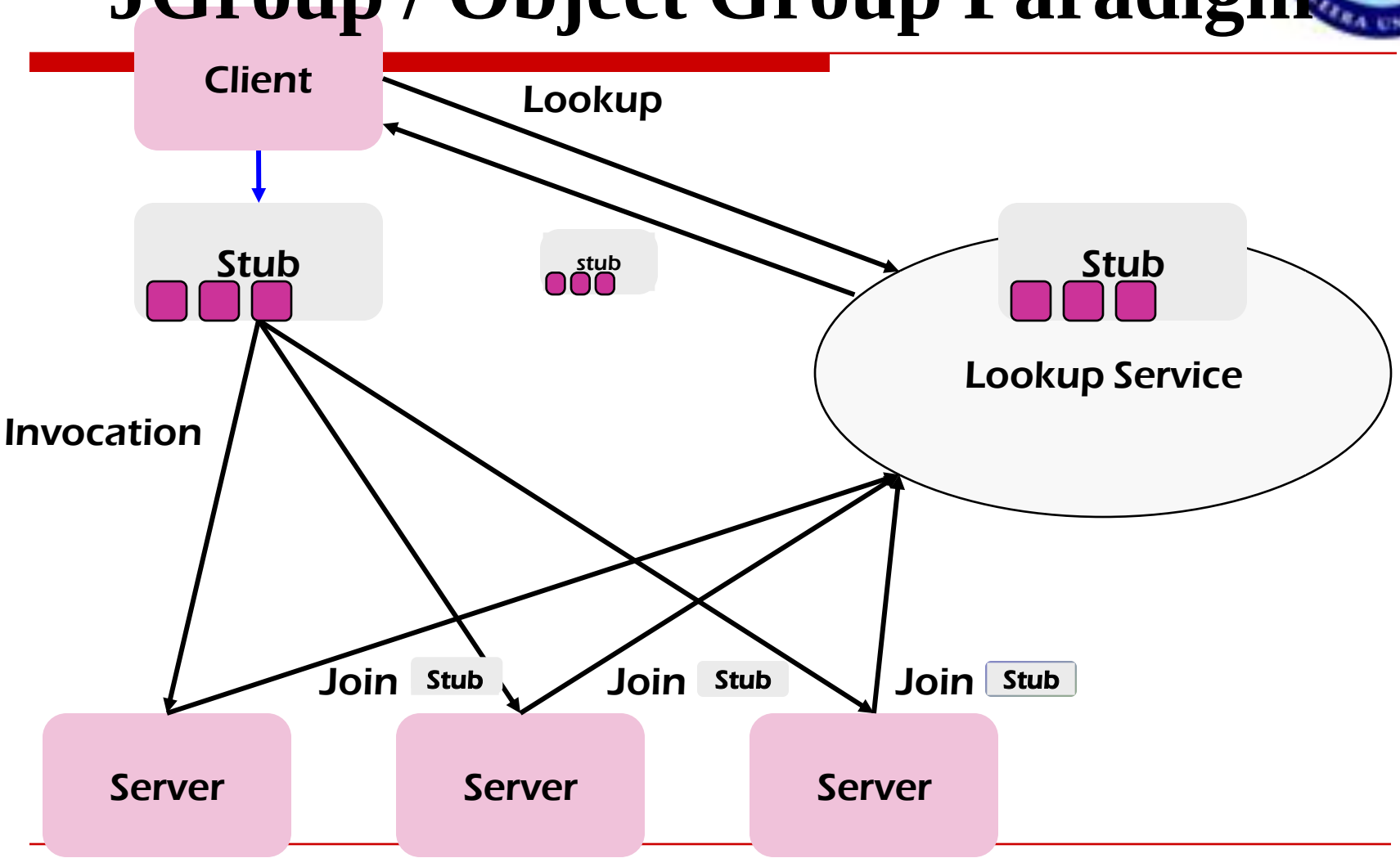
- Object group:
  - A dynamic collection of server objects that cooperate in order to deliver some service and maintain shared state.
- Group method invocations:
  - The act of invoking a method on an object group
  - The method is executed by a certain number of servers in the object group, depending on the invocation semantics.



JGroup / Object Group Paradigm



JGroup / Object Group Paradigm



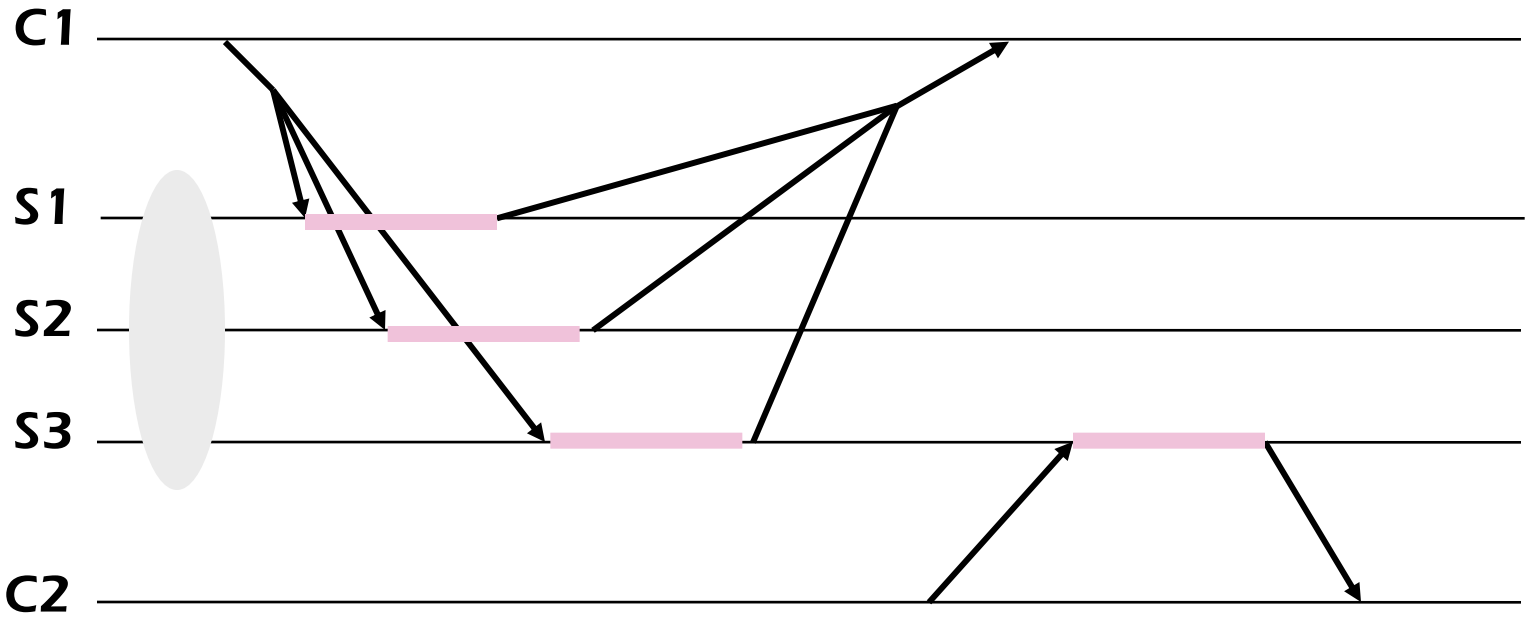
JGroup = Object Group Paradigm

- Group communication = a powerful paradigm to develop dependable distributed applications :
  - Reliable Multicast Service :
    - One-to-Many Communications
    - Ordering of events (FIFO, Causal, Total, Atomic)
  - Group Membership Service :
    - Management of dynamic groups (join/leave operations)
    - Failure monitoring (crashes, partitioning)
  - State Transfer Service :
    - State synchronization tools

JGroup

- **Group Membership Service**
  - Group composition vary dynamically (join/leave the group)
  - Using view synchronization
- **IGMI**
  - Synchronous / Asynchronous with callback methods
  - Respect to view synchronization
- **EGMI**
  - Transparent to programmer (ordinary RMI)
  - **Anycast** (read operations) / **Multicast semantic** (update operations)
  - Unique ID : used

JGroup / Semantic Invocation



Multicast invocation:

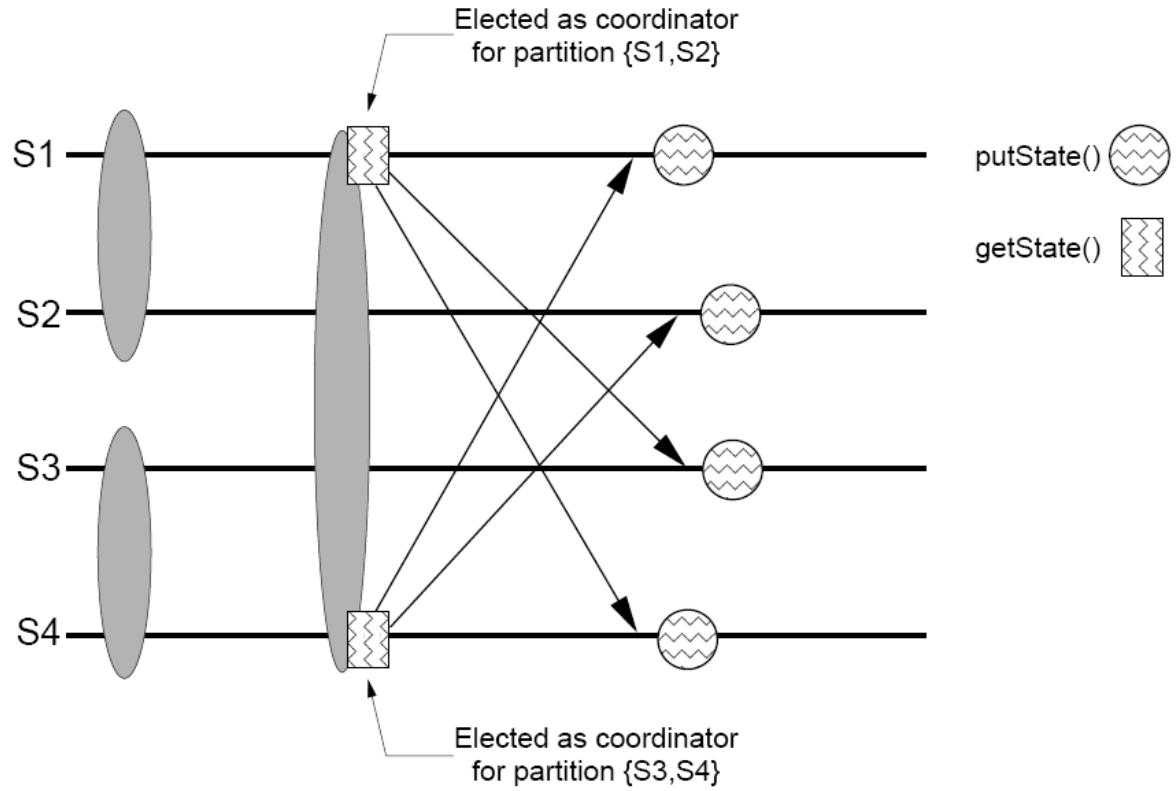
registry.bind("name", obj);

Anycast invocation:

registry.lookup("name");

JGroup / State Merging

- **State Merging Service**



Other Object Group Systems

- CORBA
  - Electra [Cornell, Zurich];
  - Object Group Service (OGS) [EPFL, Lausanne];
  - Eternal [UC Santa Barbara, Eternal Systems];
  - Newtop [Newcastle, UK].
- Java RMI
  - Filterfresh [Bell Labs];
  - JavaGroups [Cornell];
  - Aroma [UC Santa Barbara].
- DCOM
  - Quintet [Cornell].

Questions ?