Distributed Systems: **Replication Systems**

Dr Soumyabrata DEV
https://soumyabrata.dev/

School of Computer Science and Informatics
University College Dublin
Ireland

These course slides are adapted from the original course slides prepared by Dr Anca Jurcut, University College Dublin.

Distributed Systems: Active and Passive Replication

Consistency

- In replication systems, consistency refers to the correctness of the replicas within the system.

- Inconsistencies between replicas can cause errors in the operation of the system.
  - Delayed updates can cause the system to use incorrect data
    - E.g. when we deposit money into a bank account, we expect that the balance on our account will be updated globally.

- Strict consistency requires that:
  - Any read on a data item ‘x’ returns a value corresponding to the result of the most recent write on ‘x’ (regardless of where the write occurred).
  - This is often termed **linearisation**.

Linearisability

- A **linearisable system** is a system in which all the operations appear to have been performed in some sequential and non-overlapping order.
  - That is, the set of operations that are performed by the system can be written down sequentially (even if carried our by more than one process)
    - E.g. boil the water, get the cup, add the coffee, pour the water, …

- A replication system is said to be linearisable if, for any execution of the system, there is some interleaving of the series of operations issued by all the clients that satisfies that following criteria:
  - The interleaved sequence of operations meets the specification of a (single) correct copy of the objects.
  - The order of the operations in the interleaving is consistent with the real time at which the operations occurred in the actual execution.

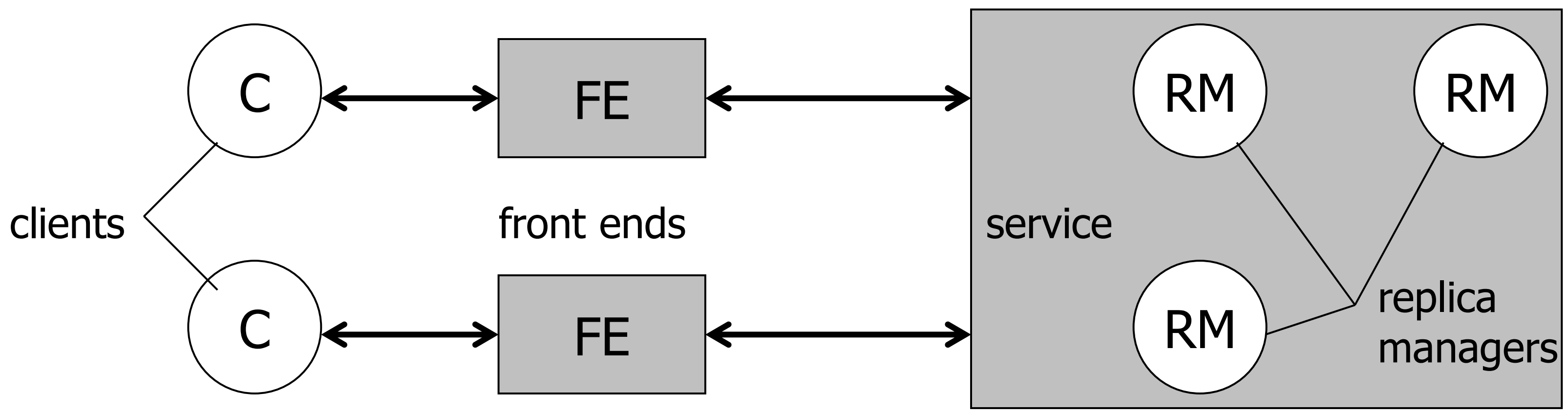
Sequential Consistency

- A weaker correctness condition for Replication Systems is **sequential consistency**:
  - The interleaved sequence of operations meets the specification of a (single) correct copy of the objects.
  - The order of operations in the interleaving is consistent with the program order in which each individual client executed them.

- The difference between Sequential Consistency and Linearisability is that:
  - Absolute time and total ordering of operations is not required.
  - Instead, ordering is relative to the order of events at each separate client.

- As a result, Linearisation requires that the ordering of events conform to the real-world, while Sequential Consistency requires only that the operations be ordered relative to each process.

Recap: Replication Systems



Recap: Replication Systems

- Handling a request to perform an operation on a logical object normally involves 5 steps:
  - **Request**: FE issues the request to one or more RMs
    - Single message or Multicast
  - **Coordination**: RMs coordinate to execute the request consistently.
    - They agree on whether or not the request is to be applied
    - They decide on the ordering of the request relative to others
    - FIFO, Causal, Total, …
  - **Execution**: The RMs execute the request
    - This may be done tentatively (i.e. it can be undone later)
  - **Agreement**: The RMs reach consensus on the effect of the request
  - **Response**: One or more RMs respond to the FE
    - Single/Multiple Responses
    - Response Acceptance/Synthesis (where necessary)

Passive Replication

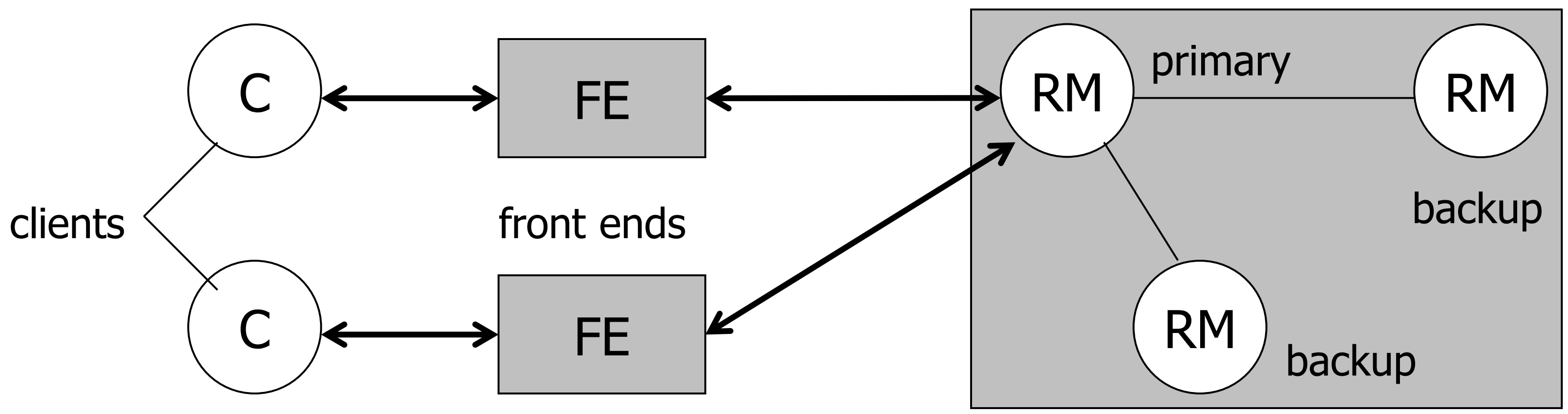
- In the passive or primary-backup model of replication, there is, at any one time:
  - a single primary replica manager, and
  - one or more secondary (backup) replica managers

- In its simplest form, the FEs communicate only with the primary RM to obtain the service.
  - The primary RM executes the operations and sends copies of the updated data to the backups.

- If the primary fails, then one of the backups is promoted to act as the primary
  - i.e. some election algorithm is employed…

- Because the primary RM sequences all operations upon the shared objects, passive replication systems are linearizable!

Passive Replication



Passive Replication

- When a FE issues a request, the following steps are executed:
- **Request**: The FE issues a request, containing a unique identifier, to the primary RM.
- **Coordination**: The primary takes each request atomically, in the order in which it receives it.
  - It checks the unique identifier, in case it has already executed the request and if so, it simply resends the response.
- **Execution**: The primary executes the request and stores the response.
- **Agreement**: If the request is an update then the primary sends the updated state, the response, and the unique identifier to the backups.
  - The backups send an ACKnowledgement.
- **Response**: The primary responds to the FE, which hands the response back to the client.

Passive Replication

- The primary and backups are organized as a group, and the primary uses view-synchronization group communication to send updates to the backups.

- When a primary fails, it is replaced by a unique backup.

- The GMS will eventually deliver a new group view to one of the backups.
  - This can be used to kick off an election algorithm.

- The RMs that survive agree on which operations had been performed at the point where the replacement primary takes over.
  - View-synchronization group communication ensures that either all the backups or none of them will deliver any given update before delivering the new view.

Passive Replication

- To survive up to f process crashes, a passive replication system requires f+1 replica managers.

- The front end requires little functionality:
    - It needs only to be able to locate the new primary when the current primary does not respond.

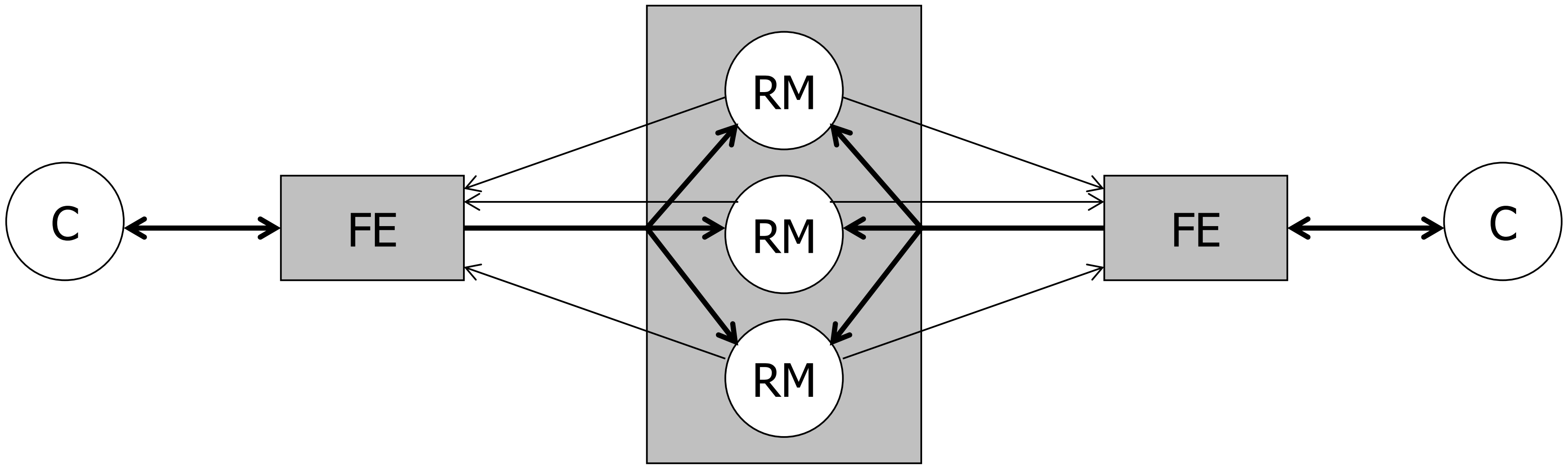
- Large overheads due to view-synchronization group communication

- Variant implementations allow clients to submit reads to the backups:
    - This reduces the load on the primary.
    - But we lose linearizability, and instead get only sequential consistency.

Active Replication

- In the active replication model, replica managers are state machines that play equivalent roles and are organised as a group.

- Front Ends multicast their requests to the group and all the RMs process the requests independently, but identically, and reply.



Active Replication

- When a FE issues a request, the following steps are executed:

- **Request**: The FE sends a multicast request to the replica manager group, containing a unique identifier.
  - The multicast is both totally-ordered and reliable.
  - The FE does not issue the next request until it has received a response.
- **Coordination**: The group communication system delivers the request to every correct RM in the same (total) order.
- **Execution**: The RM executes the request.
  - Since the RMs are state machines and all requests are delivered in the same order, correct RMs all process the request identically.
  - The response contains the clients unique request identifier.
- **Agreement**: No agreement phase is needed because of the multicast delivery semantics.
- **Response**: Each RM sends its response to the FE.
  - The number of replies that the FE collects before sending a response depends upon both the failure assumptions and the multicast algorithm.

Active Replication

- The system is sequentially consistent:
  - All correct RMs process the same sequence of requests.
  - The reliability of multicast ensures that every correct RM processes the same set of requests and the total order ensures that they process them in the same order.
  - The FE requests are served in FIFO order:
    - The FE waits for the response to the previous request before issuing the next request.

- When a process fails there is no impact on the performance of the service:
  - The remaining RMs continue to respond in the normal way.
  - The FE collects and compares the responses it gets before sending a response to the client.
    - This can be first response, majority response, …

Summary

- Consistency: How to keep replicas consistent.
  - Linearisability:
    - The sequence of interleaved operations that is totally ordered based on real time.
  - Sequential Consistency:
    - Any sequence of interleaved operations that satisfies the local ordering of operations carried out by the processes.

- Two types of replication:
  - **Passive Replication**:
    - Primary RM handles all requests and sends updates to the backup RMs.
    - Linearisable
  - **Active Replication**:
    - All RMs handle all requests concurrently.
    - Sequentially Consistent.

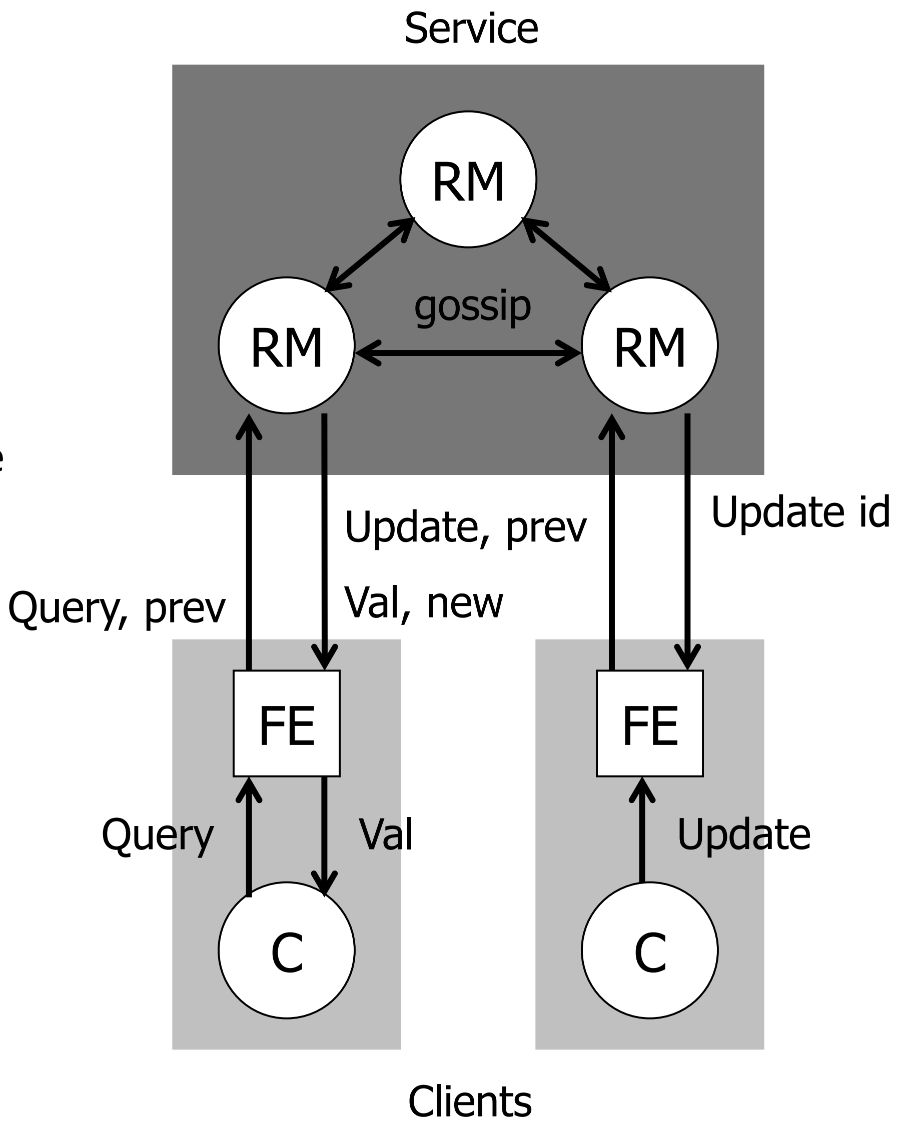
Distributed Systems:

Case Study: The Gossip Architecture

Introduction

- The **Gossip architecture** is a framework for implementing highly available services.
  - Data is replicated close to the groups of clients that need it.
  - Replica Managers then exchange "gossip" messages periodically in order to convey the updates they have each received from clients.

- The service supports two operations:
  - Query: read-only operations
  - Update: modify-only operations



Introduction

- The Front End is able to send queries and updates to any replica manager they choose.
  - Any that is available and can provide reasonable response times.

- The system makes two guarantees:
  - Each client obtains a consistent service over time:
    - Replica Managers respond to a query with data that reflects at least the updates that the client has observed so far.
    - This occurs even when the client communicates with a replica manager that is “less advanced” that the one it used before.
  - Relaxed consistency between replicas:
    - All replicas eventually receive all updates.
    - Updates are applied with ordering guarantees that make the replicas sufficiently similar to suit the needs of the applications.

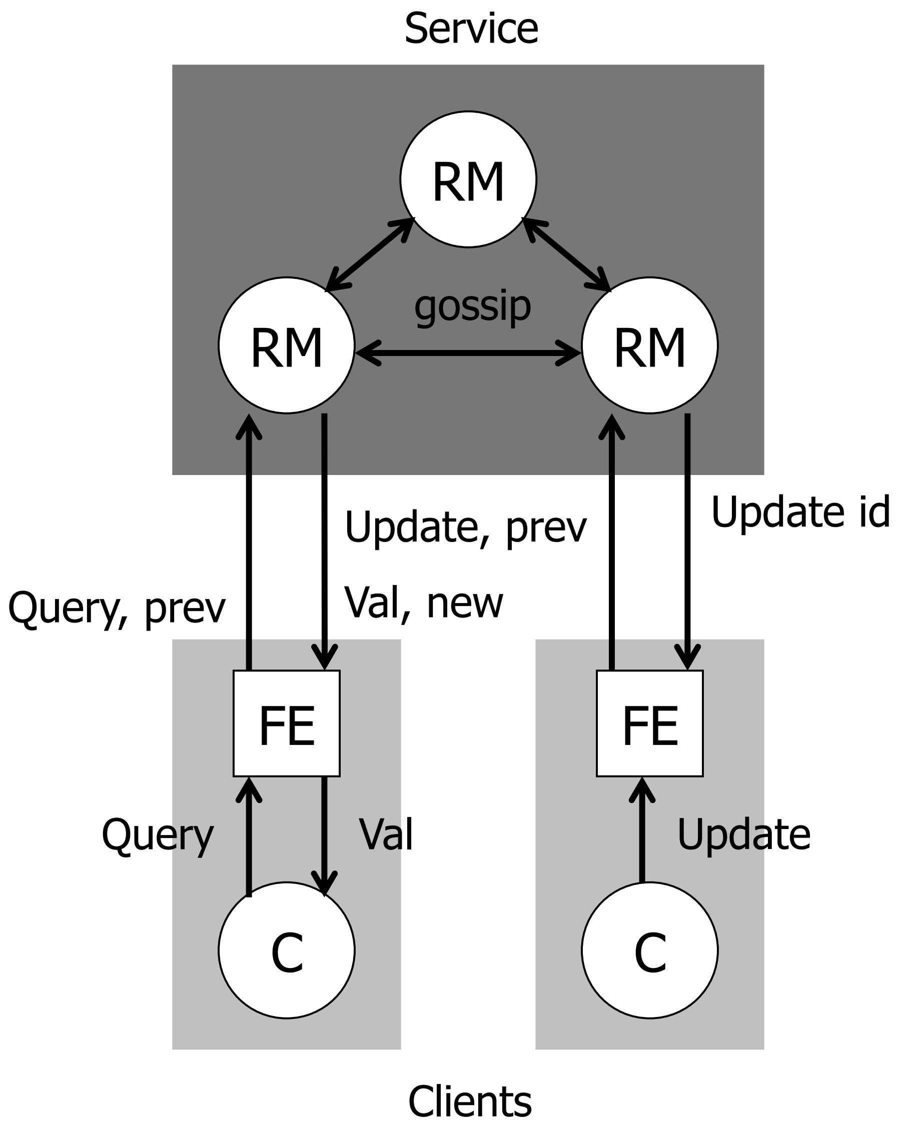
- Gossip supports many different consistency models:
  - From causal update ordering to total and causal (forced) ordering and immediate orderings.

Request Processing

- When a FE issues a request, the following steps are executed:
  - **Request**: The FE sends request to only a single replica manager at a time.
    - FE communicates with different RMs whenever the one it normally uses fails, becomes unreachable, or is overloaded.
    - The FE blocks for queries, but either returns immediately after an update or alternatively, it can be configured to return after f+1 updates.
  - **Update Response:** If the request is an update then the RM replies as soon as it has received the update.
  - **Coordination**: The RM that receives the request does not apply it until the ordering constraints are satisfied.
  - **Execution**: The RM executes the request.
  - **Query Response**: If the request is a query, then the RM replies at this point.
  - **Agreement**: The RM's update each other by exchanging gossip messages, which contain the most recent updates.
    - Regularity of updates can be from every few updates to whenever it finds out that it is missing an update.

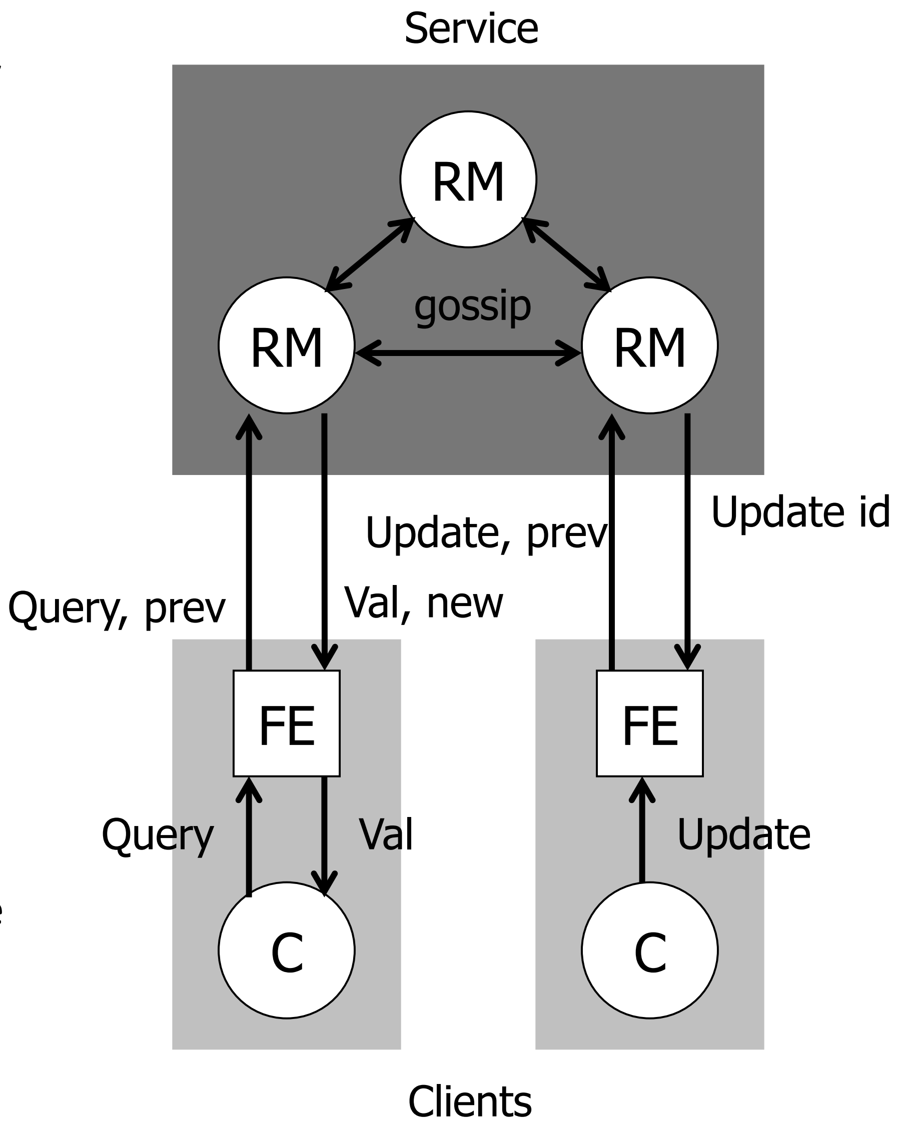
FE Timestamps

- In order to control the ordering of operation processing, the FE keeps a **vector timestamp**:
  - Each RM maintains a last update timestamp
    - Can be the actual time of the last update or linear time
  - Each FE maintains a list of the latest timestamps for each RM.
  - This list is sent in every request that the FE makes.
    - This is **prev** in the diagram



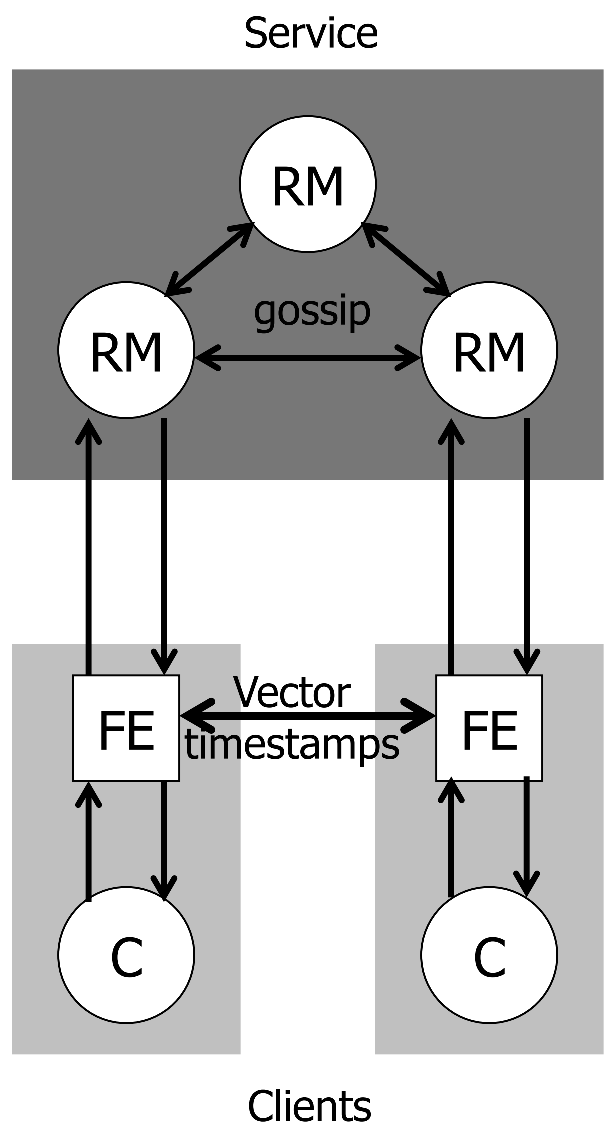
FE Timestamps

- When the FE receives the result of a query operation:
  - The responding RM returns a new vector timestamp.
    - This is **new** in the diagram.
  - Similarly, an update operation returns a vector timestamp that is unique to the update.
    - This is the **update id** in the diagram
- Each returned timestamp is merged with the FE's previous timestamp.
  - This allows the FE to record what version of the replicated data has been observed by the client.



FE Timestamps

- Clients exchange data by:
  - accessing the same gossip service, and
  - communicating directly with one another.
- Since client-to-client communication can also lead to causal relationships between operations it also occurs via the clients FEs
  - This allows FEs to piggyback their vector timestamps on messages to other clients.
- The recipients merge them with their own timestamps in order that causal relationships can be inferred correctly.



Replica Manager State

- Every Replica Manager maintains the following state components:
  - **Value**: The value of the application state as maintained by the RM.
    - Each RM is a state machine
    - Begins with an initial value
    - Subsequent values are derived from the application of update operations to that state.
  - **Value timestamp**: Vector timestamp that represents the updates that are reflected in the value
    - One for each RM
    - Updated whenever an update operation is applied to the value
  - **Update Log**: All update operations are recorded in this log as soon as they are received.
    - Only **stable updates** are applied to the value.
    - Stability of the update depends on the ordering guarantees (causal, forces, immediate)
    - The RM uses the log to ensure that updates are sent to all other RMs.

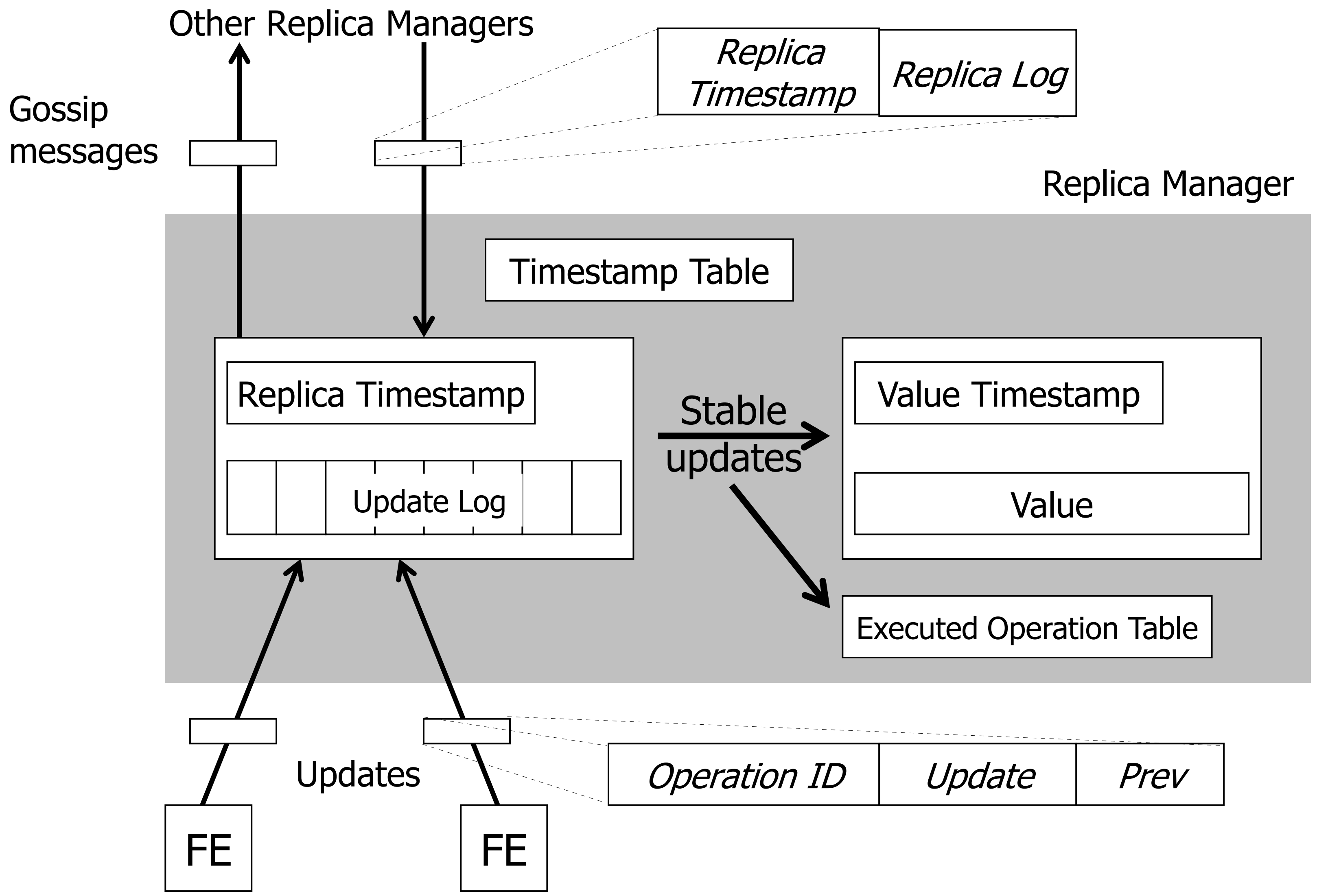
Replica Manager State

- **Replica Timestamp**: Vector timestamp that represents those updates that have been accepted by the RM.
  - That is, those updates that have placed in the managers log.
  - It differs from the value timestamp because not all updates in the log are stable.

- **Executed Operation Table**: A table containing the unique FE supplied identifiers of updates that have been applied to the value.
  - The same update may arrive at a given RM from a FE and in gossip messages from other RM's.
  - To prevent an update being applied twice the RM checks this table before adding an update to the log.

- **Timestamp Table**: Contains a vector timestamp for each other RM.
  - It is filled with timestamps that arrive from them in gossip messages.
  - RMs use the table to establish when an update has been applied to all RMs

Replica Manager State



Query Operations

- The simplest operation to consider is a query.

- A query request contains:
  - A description of the operation
  - A timestamp q.prev sent by the FE

- The latter reflects the latest version of the value that the FE has read or submitted has an update.

- The task of the RM is to return a value that is at least as recent as this.
  - If **valueTS** is the replicas timestamp, then q can be applied to the replica’s value if:

$$\mathbf{q.prev \leq valueTS}$$

Gossip Messages

- RMs send gossip messages to other RMs.
  - Each message contains information concerning one or more updates so that the other RMs can bring their state up to date.
  - Each RM uses the entries in its timestamp table to estimate what updates the other RMs have not yet received.
- A gossip message **m** sent by a RM contains two items:
  - its log, **m.log**, and
  - its replica timestamp **m.ts**
- When an RM receives a message it does three things:
  - Merges the arriving log with its own
  - Applies any updates that have become stable but have not yet been executed
  - Remove records from the log and entries from the executed operation table when it is known that the updates have been applied everywhere and there is no danger of repeats.

Summary

- So, Gossip is a general framework for implementing highly available services.
  - Clients interact with Gossip RMs via a FE
  - The FE chooses a single RM with which to communicate until:
    - RM crashes
    - Communication link failure
    - Overloading.
  - RMs are treated as state machines that can execute two basic operations:
    - Query
    - Update
  - Updates are passed to other RMs through **Gossip Messages**.

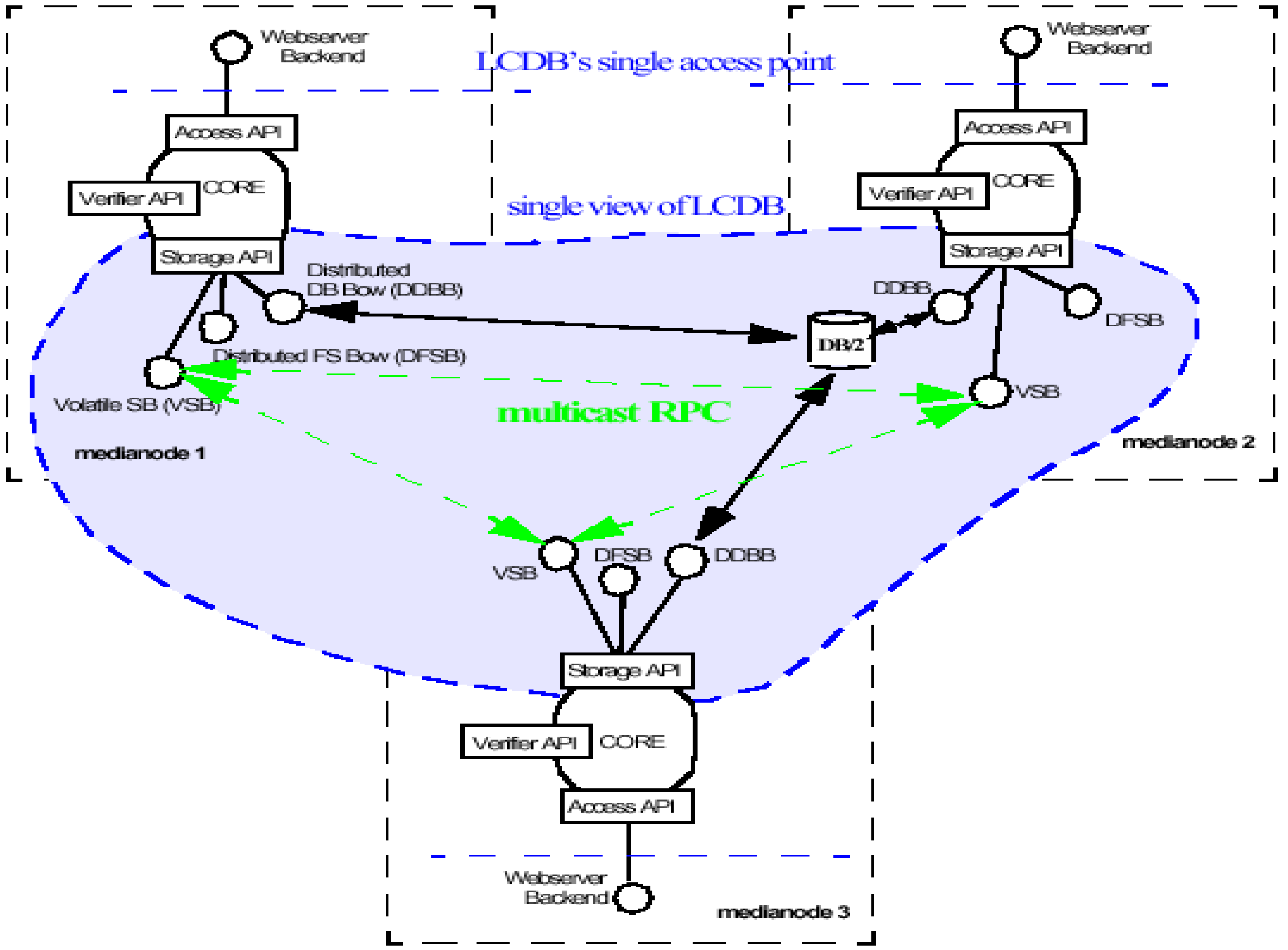
- Issues not covered by the Gossip Architecture are:
  - When to exchange gossip messages
  - How a RM picks who to send messages to.

Distributed Systems:
Case Study: Medianode

Example: Medianode

- An open infrastructure for sharing multimedia-enhanced teaching materials amongst lecture groups.
  - On et al., “Replication for a Distributed Multimedia System”, 2001
- Designed For:
  - High Availability:
    - Connected and Disconnected Modes
    - geographically distributed replicas
  - Consistency: Concurrent updates and failures
  - Location and Access Transparency
  - Cost Efficient Update Transport: Multicast based update mechanism
  - QoS Support

Example: Medianode



Presentation Data Types

- Presentation Contents
- Presentation Description Data (XML)
- Metadata:
  - User, System, Domain and Organisation
  - System resource usage information
  - User session information

Replica Classification

- Metareplicas:
  - Replicated meta-data objects
  - Eg. a list of medianodes that contain up-to-date replicas of a given file
- Softreplicas:
  - Small non-persistent meta-data objects
  - Eg. system resource information, user session information, …
- Truereplicas:
  - Large persistent objects
  - Eg. Media content files

Replication Mechanism

- Replica Manager keep track of a local file table that includes replicas.
- Information about whether and how many replicas are created is contained in every file table.
- Read access is allowed for local replicas.
  - Local cached replica is deemed valid until notified otherwise.
- Updates cause the RM to send a multicast-based update signal to RMs that have replicas.
- Multicast addresses are associated with replica sub-groups.
  - Sub group examples: file directory, or set of presentations about the same topic

Thank you

For general enquries, contact:

Please contact the Head Teaching Assistant: Xingyu Pan (Star), Xingyu.Pan@ucdconnect.ie