CSE 486/586 Distributed Systems
Replication with
View Synchronous Group Communication

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Recap: Non-Exclusive Locks

non-exclusive lock compatibility

Lock already set	Lock requested: read	Lock requested: write
none	OK	OK
read	OK	WAIT
write	WAIT	WAIT

Recap: Two-Version Locking

lock compatibility

Lock already set	Lock requested: read	Lock requested: write	Lock requested: commit
none	OK	OK	OK
read	OK	OK	WAIT
write	OK	WAIT	
commit	WAIT	WAIT	

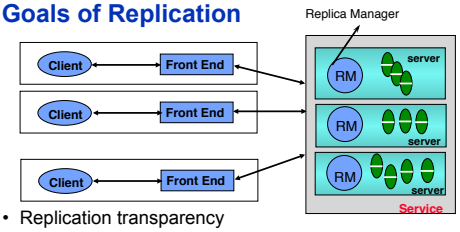
Recap: Distributed Transactions

- Atomic commit problem
  - Either all commit or all abort
- 2PC
  - Voting phase
  - Commit phase

Replication

- Enhances a service by replication
  - In what ways?
- Increased availability of service. When servers fail or when the network is partitioned.
  - P: probability that one server fails= 1 – P= availability of service. e.g. P = 5% => service is available 95% of the time.
  - P^n: probability that n servers fail= 1 – P^n= availability of service. e.g. P = 5%, n = 3 => service available 99.875% of the time
- Fault tolerance
  - Under the fail-stop model, if up to f of f+1 servers crash, at least one is alive.
- Load balancing
  - One approach: Multiple server IPs can be assigned to the same name in DNS, which returns answers round-robin.

Goals of Replication



- Replication transparency
  - User/client need not know that multiple physical copies of data exist.
- Replication consistency
  - Data is consistent on all of the replicas (or is converging towards becoming consistent)

Replica Managers

- Request Communication
  - Requests can be made to a single RM or to multiple RMs
- Coordination: The RMs decide
  - whether the request is to be applied
  - the order of requests
    - » FIFO ordering: If a FE issues r then r', then any correct RM handles r and then r'.
    - » Causal ordering: If the issue of r "happened before" the issue of r', then any correct RM handles r and then r'.
    - » Total ordering: If a correct RM handles r and then r', then any correct RM handles r and then r'.
- Execution: The RMs execute the request (often they do this tentatively – why?).

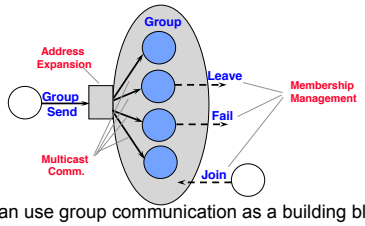
Replica Managers

- Agreement: The RMs attempt to reach consensus on the effect of the request.
  - E.g., two phase commit through a coordinator
  - If this succeeds, effect of request is made permanent
- Response
  - One or more RMs respond to the front end.
  - The first response to arrive is good enough because all the RMs will return the same answer.

Replica Managers

- One way to provide (strong) consistency
  - Start with the same initial state
  - Agree on the order of read/write operations and when writes become visible
  - Execute the operations at all replicas
  - (This will end with the same, consistent state)
- Thus each RM is a *replicated state machine*
  - "Multiple copies of the same State Machine begun in the Start state, and receiving the same Inputs in the same order will arrive at the same State having generated the same Outputs." [Wikipedia, Schneider 90]
- Does this remind you of anything? What communication primitive do you want to use?
  - Group communication (reliable, ordered multicast)

Revisiting Group Communication



- Can use group communication as a building block
- "Member"= process (e.g., an RM)
- Static Groups: group membership is pre-defined
- Dynamic Groups: members may join and leave, as necessary

Revisiting Reliable Multicast

- Integrity: A correct (i.e., non-faulty) process *p* delivers a message *m* at most once.
  - "Non-faulty": doesn't deviate from the protocol & alive
- Agreement: If a correct process delivers message *m*, then all the other correct processes in group(*m*) will eventually deliver *m*.
  - Property of "all or nothing."
- Validity: If a correct process multicasts (sends) message *m*, then it will eventually deliver *m* itself.
  - Guarantees liveness to the sender.
- Validity and agreement together ensure overall liveness: if some correct process multicasts a message m, then, all correct processes deliver m too.

Multicast with Dynamic Groups

- How do we define something similar to reliable multicast in a dynamic group?
- Approach
  - Make sure all processes see the same *versioned* membership
  - Make sure reliable multicast happens within each version of the membership
- Versioned membership: views
  - "What happens in the view, stays in the view."

CSE 486/586 Administrivia

- PA3 deadline: 4/11 (Friday)
- Midterm next Monday

Views

- A group membership service maintains group views, which are lists of current group members.
  - This is NOT a list maintained by one member, but…
  - Each member maintains its own local view
- A view $V_p(g)$ is process p's understanding of its group (list of members)
  - Example: $V_{p,0}(g) = \{p\}$, $V_{p,1}(g) = \{p, q\}$, $V_{p,2}(g) = \{p, q, r\}$, $V_{p,3}(g) = \{p,r\}$
  - The second subscript indicates the "view number" received at p
- A new group view is disseminated, throughout the group, whenever a member joins or leaves.
  - Member detecting failure of another member reliable multicasts a "view change" message (requires causal-total ordering for multicasts)
  - The goal: the compositions of views and the order in which the views are received at different members is the same.

Views

- An event is said to occur in a view $v_{p,i}(g)$ if the event occurs at p, and at the time of event occurrence, p has delivered $v_{p,i}(g)$ but has not yet delivered $v_{p,i+1}(g)$.
- Messages sent out in a view i *need to be delivered in that view* at all members in the group
- Requirements for view delivery
  - Order: If p delivers $v_i(g)$ and then $v_{i+1}(g)$, then no other process q delivers $v_{i+1}(g)$ before $v_i(g)$.
  - Integrity: If p delivers $v_i(g)$, then p is in all $v_{*,i}(g)$.
  - Non-triviality: if process q joins a group and becomes reachable from process p, then eventually, q will always be present in the views that delivered at p.
    - » Exception: partitioning of group
    - » We'll discuss partitions next lecture. Ignore for now.

View Synchronous Communication

- View Synchronous Communication = Group Membership Service + Reliable multicast
- "What happens in the view, stays in the view"
- It is *virtual*
  - View and message deliveries are allowed to occur at different physical times at different members

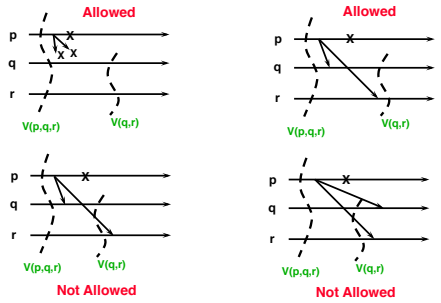
Reminder: Reliable Multicast

- Integrity: A correct (i.e., non-faulty) process *p* delivers a message *m* at most once.
  - "Non-faulty": doesn't deviate from the protocol & alive
- Validity: If a correct process multicasts (sends) message *m*, then it will eventually deliver *m* itself.
  - Guarantees liveness to the sender.
- Agreement: If a correct process delivers message *m*, then all the other correct processes in group(*m*) will eventually deliver *m*.
  - Property of "all or nothing."
- Validity and agreement together ensure overall liveness: if some correct process multicasts a message m, then, all correct processes deliver m too.

View Synchronous Communication Guarantees

- Integrity: If p delivered message m, p will not deliver m again. Furthermore, p and the process that sent m is in the same view in which p delivers m.
- Validity: Correct processes always deliver all messages. That is, if p delivers message m in view v(g), and some process $q \in v(g)$ does not deliver m in view v(g), then the next view v'(g) delivered at p will not include q.
- Agreement: Correct processes deliver the same sequence of views, and the same set of messages in any view.
  - If p delivers m in V, and then delivers V', then all processes in $V \cap V'$ deliver m in view V
- All view delivery conditions (order, integrity, and non-triviality conditions, from last slide) are satisfied

Examples



State Transfer

- When a new process joins the group, state transfer may be needed (at view delivery point) to bring it up to date
  - "state" may be list of all messages delivered so far (wasteful)
  - "state" could be list of current server object values (e.g., a bank database) – could be large
  - Important to optimize this state transfer
- View Synchrony = "Virtual Synchrony"
  - Provides an abstraction of a synchronous network that hides the asynchrony of the underlying network from distributed applications
  - But does not violate FLP impossibility (since can partition)
- Used in ISIS toolkit (NY Stock Exchange)

Summary

- Replicating objects across servers improves performance, fault-tolerance, availability
- Raises problem of Replica Management
- Group communication an important building block
- View Synchronous communication service provides totally ordered delivery of views+multicasts
- RMs can be built over this service

Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC).