DESIGN DOCUMENT

Advanced Operating Systems – CS6378

Ikjun Jang & Emrah Cem

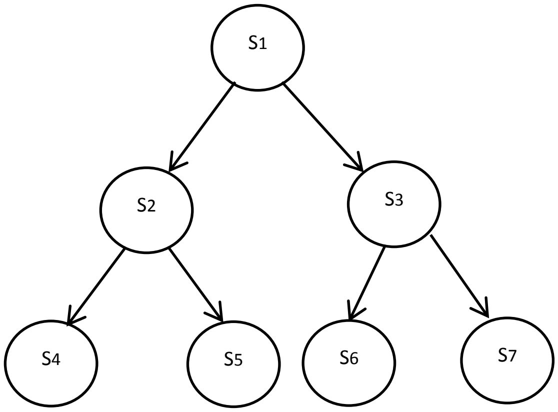
Project 3

12/10/2013

This document includes the design details of the project.

1. Basic Assumptions

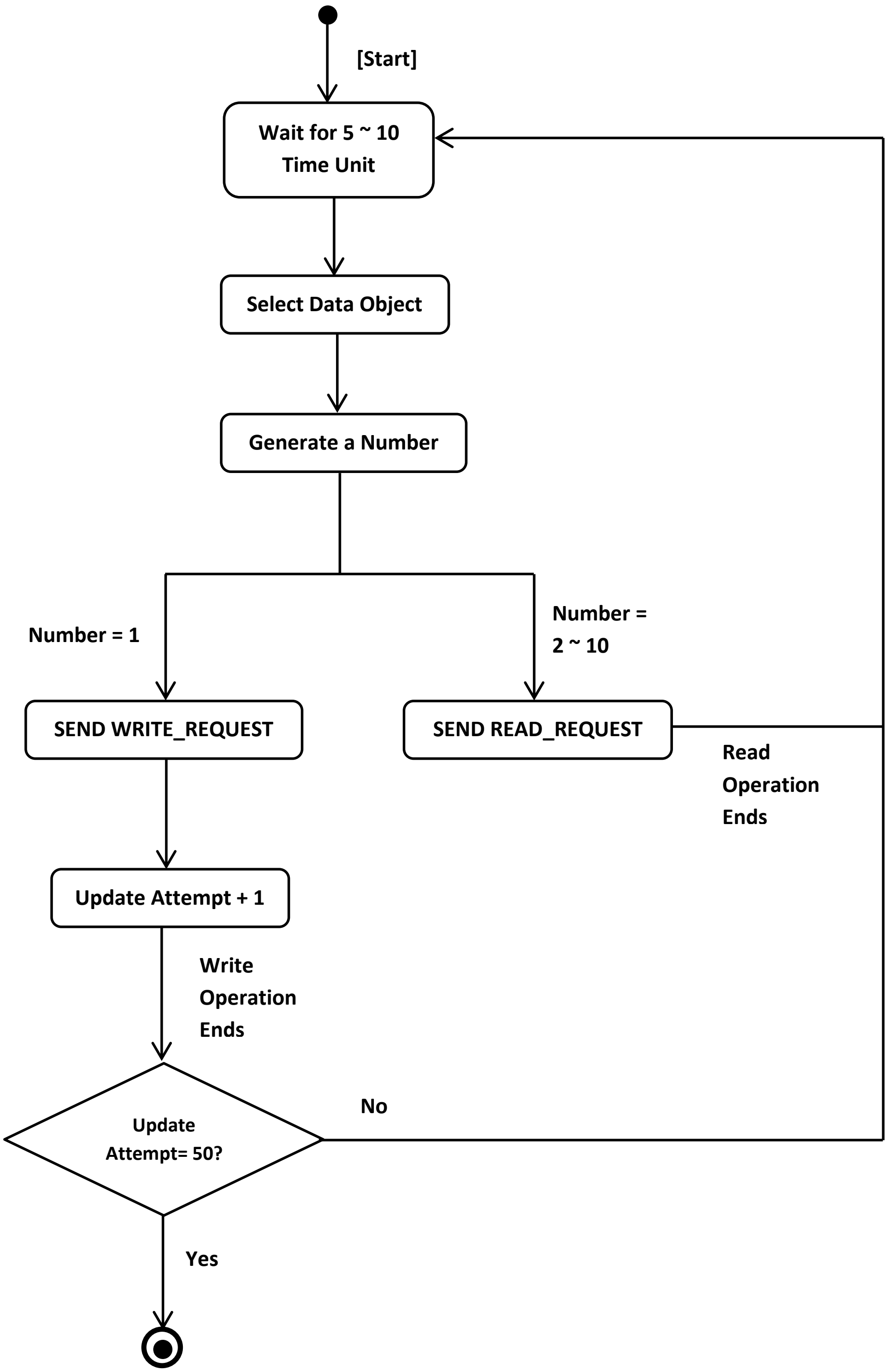
- There are seven servers: S1 to S7
- Each server maintains replicas of integer type data objects initialized as:
  - D0 = 0, D1 = 1, D2 = 2, D3 = 3
- There are five clients: C0 to C4
- All communication channels are FIFO using IP stream sockets.
- Servers are logically arranges as a balanced binary tree as follows:



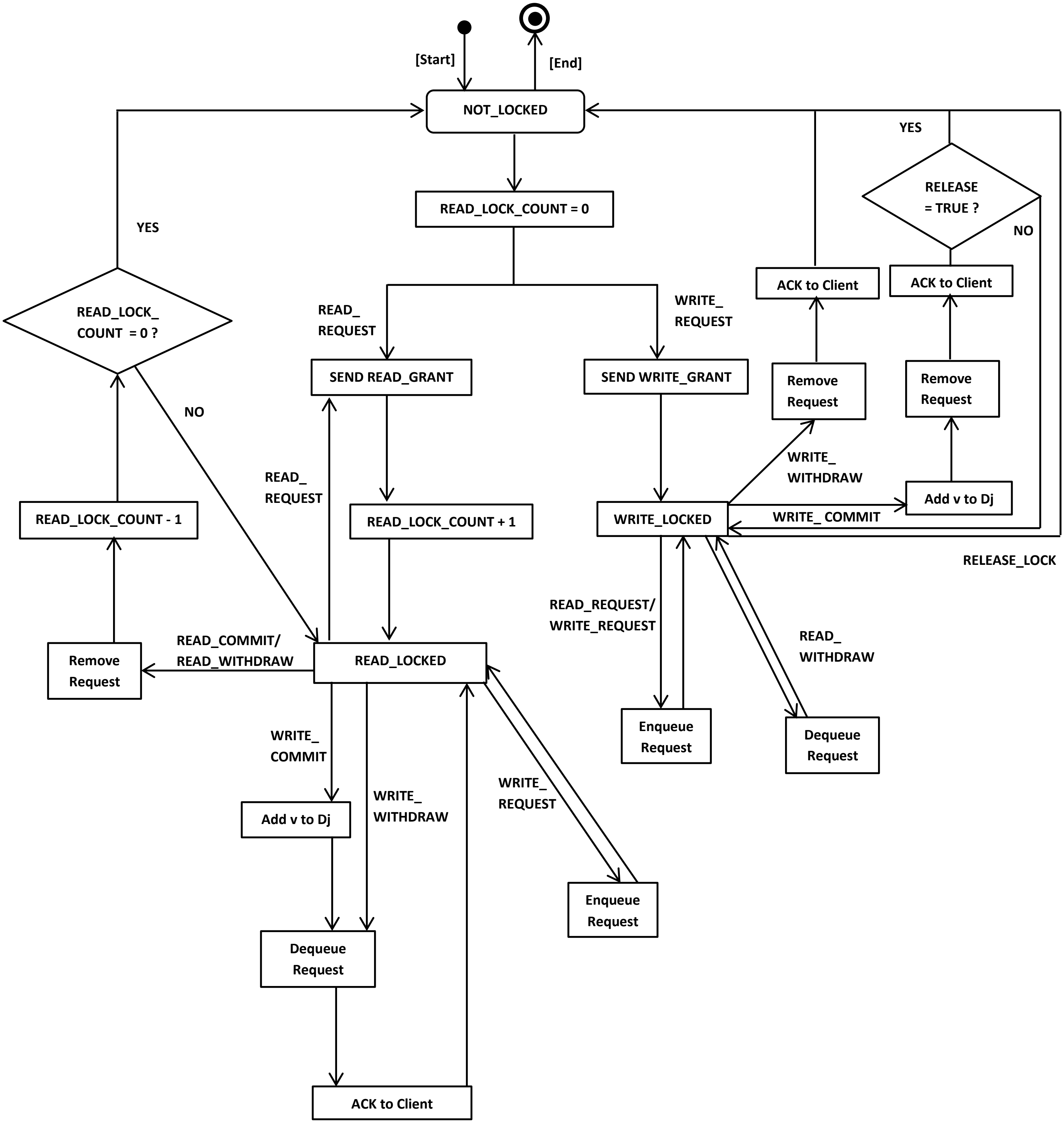
- Update on a data object can be performed when a client receives grants from one of the following 15 sets of servers:
  - {S1, S2, S4}, {S1, S2, S5}, {S1, S4, S5}, {S1, S3, S6}, {S1, S3, S7}, {S1, S6, S7}, {S2, S4, S3, S6}, {S2, S4, S3, S7}, {S2, S4, S6, S7}, {S2, S5, S3, S6}, {S2, S5, S3, S7}, {S2, S5, S6, S7}, {S4, S5, S3, S6}, {S4, S5, S3, S7}, {S4, S5, S6, S7}

We omit protocol and operation description since it is provided in the project description document. Instead, we choose to interpret the description into state and sequence diagrams.

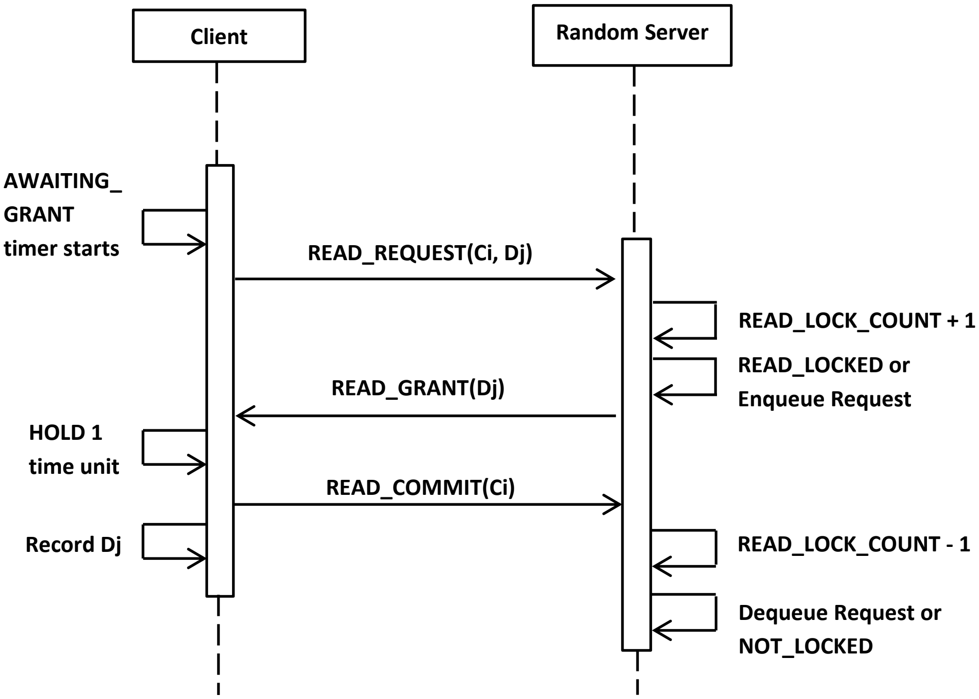
2. Client Operation State Diagram



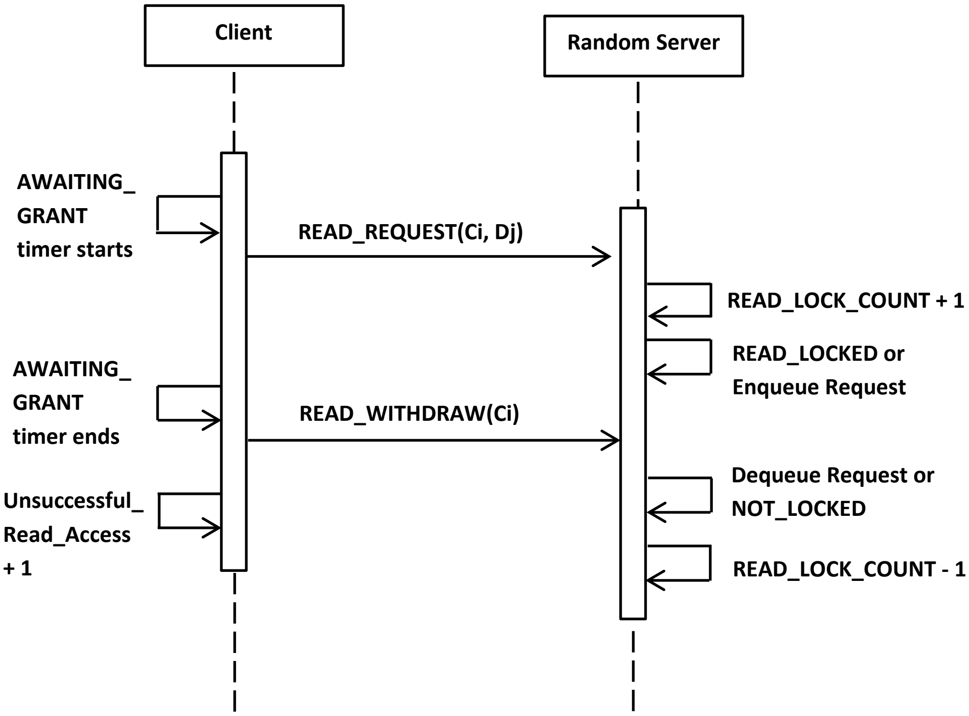
3. State of Lock Diagram for Data Object



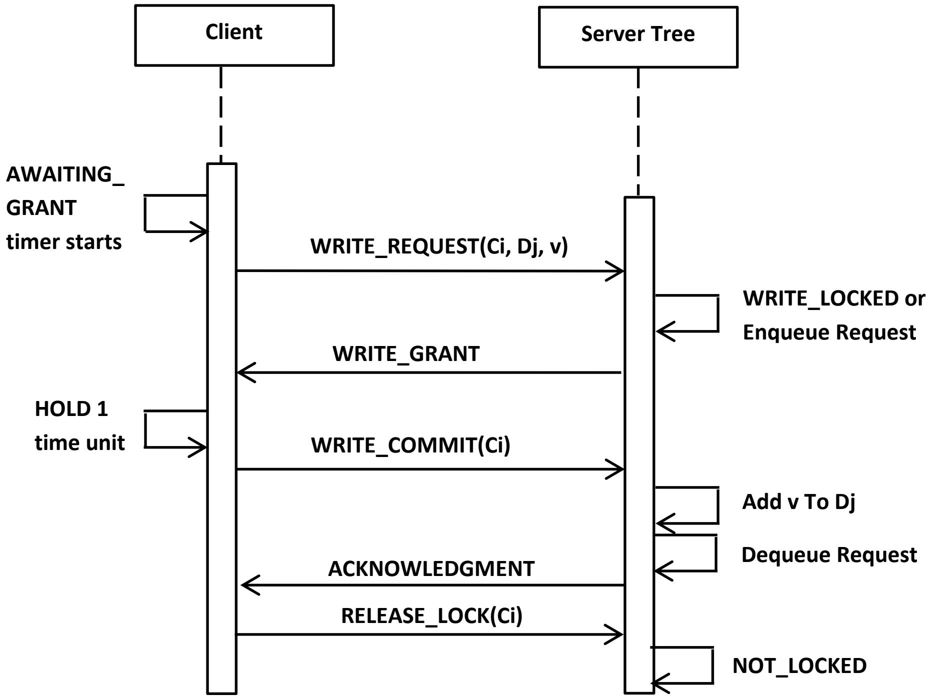
4. Successful Read Operation Sequence Diagram



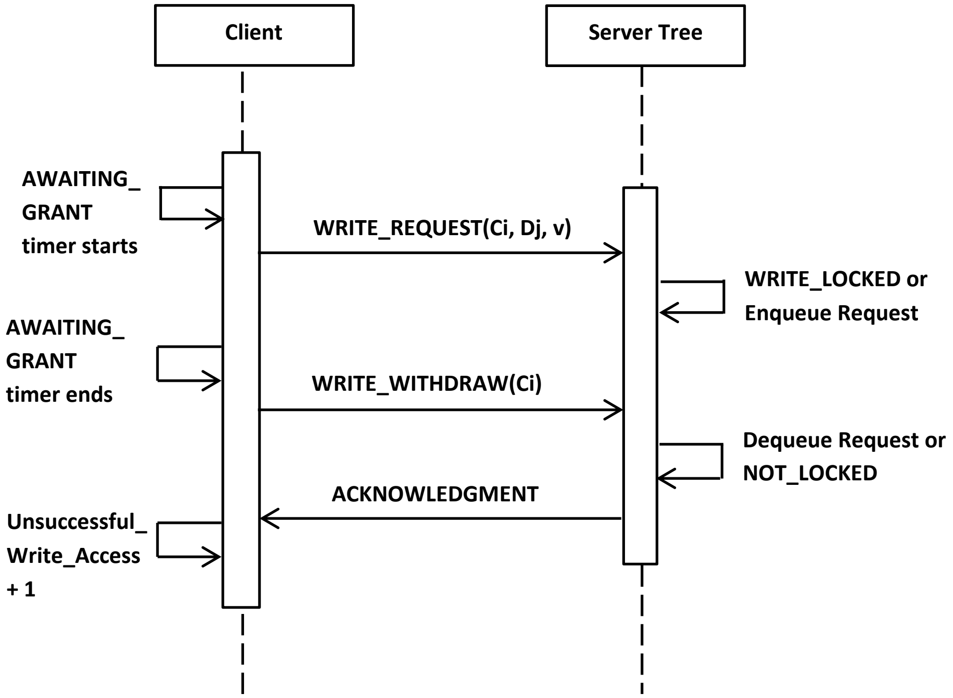
5. Unsuccessful Read Operation Sequence Diagram



6. Successful Write Operation Sequence Diagram



7. Unsuccessful Write Operation Sequence Diagram



8. Class Diagram (Details) for principal classes

*We omit some of trivial constructors, fields and methods for simplification.

MessageReceiver
#id: Integer {final} #lock: ReentrantLock #numberOfMessagesReceived: Long # messageQueue: PriorityBlockingQueue<FIFOEntry<AbstractMessage>> # messageHandlerFactory: AbstractMessageHandlerFactory
+MessageReceiver(id: Integer)

MessageConsumer
-queue: BlockingQueue<FIFOEntry<AbstractMessage>> -msg_receiver: MessageReceiver

+MessageConsumer(msg_receiver: MessageReceiver, queue: BlockingQueue<FIFOEntry<AbstractMessage>> {final})
+run()

Client
-numberOfDataObjects: Integer = 4 -holdTime: Double = 1 -awaitingGrant: Double = 20 -timeUnit: CustomTimeUnit -waitTimeRange: TimeRange -state: ClientState -remoteHash: HashMap<Integer, RmoteSite> -servers: List<RemoteSite> -lockingServerIds: Set<Integer> -ackedServerIds: Set<Integer> -requestFactory: AbstractRequestFactory -serverInfoProvider: AbstractRemoteSiteProvider -serverSelector: AbstractServerSelector -rand MyRandom -lastRequest: AbstractRequest -canStartNextRound: Boolean -canStartNextAccess: Condition {final} = lock.newCondition() #sendTimeInMillis: Long - grantLock: Long
+Client(id: integer, seed: Integer, holdTime: Double, amount: Long, pathToConfigFile: String) +runClient() -waitForATrigger() -createTCPConnectionsToServers() -resetState() -sendRequests() -sendRequest(request: AbstractRequest, serversToSend: List<RemoteSite>)

ClientState
+WaitState: Enumeration -waitState: WaitState -granted: Boolean -successfullRead: Integer -successfullWrite: Integer -unsuccessfullRead: Integer -unsuccessfullWrite: Integer

-receivedValues: Map<Integer, List<Integer>> -readTimes: List<Long> -writeTimes: List<Long> - totalNumberOfMessagesSent: Long
-summaryStatsInStr(list: List<Long>): String

<<enumeration>> **WaitState**
NOT_WAIT READ_WAIT WRITE_WAIT

ServerHandler
-client: Client -server: RemoteSite

+ServerHandler(client: Client, server: RemoteSite)
+run()

AbstractMessageHandler
+handleMessage(message: AbstractMessage): Boolean

ClientMessageHandler
#client: Client #message: AbstractMessage
+handleMessage(message: AbstractMessage): Boolean

ServerMessageHandler
#server: Server
+handleMessage(message: AbstractMessage): Boolean

AbstractMessageHandlerFactory
validMessageNames: Map<MessageType, AbstractMessageHandler>
+createHandler (message: AbstractMessageHandler): AbstractMessageHandler # addMessageHandler(type: MessageType, messageHandler: AbstractMessageHandler):AbstractMessageHandler

ClientMessageHandlerFactory
-client: Client
+ClientMessageHandlerFactory(client: Client) +createHandler(message: AbstractMessage):AbstractMessageHandler

ServerMessageHandlerFactory
-server: Server
+ServerMessageHandlerFactory(server: Server) +createHandler(message: AbstractMessage):AbstractMessageHandler

AckMessageHandler
+handleMessage(message: AbstractMessage): Boolean

EndMessageHandler
+handleMessage(message: AbstractMessage, outStream: OutputStream): Boolean

GrantMessageHandler
+handleMessage(message: AbstractMessage): Boolean
+wrtieGrantReceived(servers: List<RemoteSite>): Set<Integer>

ReadCommitHandler
+handleMessage(message: AbstractMessage): Boolean

ReadRequestHandler
+handleMessage(message: AbstractMessage): Boolean

ReleaseLockHandler
+handleMessage(message: AbstractMessage): Boolean

WithdrawHandler
+handleMessage(message: AbstractMessage, outStream: OutputStream): Boolean

WriteCommitHandler
+handleMessage(message: AbstractMessage, outStream: OutputStream): Boolean

WriteRequestHandler
+handleMessage(message: AbstractMessage, outStream: OutputStream): Boolean

AbstractMessage
#type: MessageType #senderId : Integer #sequenceNumber: Long #priority: Integer = 1
+AbstractMessage(senderId: Integer, sequenceNumber: Long) +compareTo(AbstractMessage: other): Integer

AbstractRequest
-serialVersionUID: long = 1 {final} #objectIndex: Integer
+AbstractRequest(senderId: Integer, sequenceNumber: Long, objectIndex: Integer) +equals(obj: Object): Boolean

<<enumeration>> **MessageType**
READ_REQUEST WRITE_REQUEST WRITE_COMMIT

READ_COMMIT
WITHDRAW_MESSAGE
GRANT_MESSAGE
END_MESSAGE
TRIGGER_MESSAGE
ACK_MESSAGE
RELEASE_LOCK

WriteRequest
-serialVersionUID: long = 1 {final}
-v: Integer
+WriteRequest(senderId: Integer, sequenceNumber: Long, objectIndex: Integer, v: Integer)
+setType()

WriteCommit
-serialVersionUID: long = 1 {final}
-releaseLock: Boolean
+WriteCommit(senderId: Integer, sequenceNumber: Long, releaseLock: Boolean)
+setType()
+isReleaseLock(): Boolean

GrantMessage
+value: Integer
+GrantMessage(value: Integer)
+setType()
+getValue(): Integer
+setValue(value: Integer)
+toString(): String

AbstractRequestFactory
+createRequest(): AbstractRequest

ProjectRequestFactory
-clientId: Integer
-numberOfDataObjects: Integer
-seqNumber: Long = 0
+ProjectRequestFactory(clientId: Integer, numberOfDataObjects: Integer)
+createRequest(): AbstractRequest

ClientHandler
-server: Server
-sock: Socket
+ClientHandler(server: Server, sock: Socket)
+handleConnection()
+run()

Data
+State: Enumeration
-dataObject: T
-state: State

-readLockCount: Integer -lockingRequest: AbstractRequest requestQueue: LinkedList<AbstractRequest>
+Data(dataObject T)

<<enumeration>> State
NOT_LOCKED READ_LOCKED WRITE_LOCKED

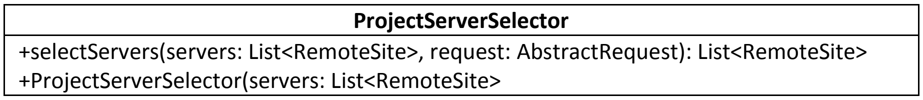
Server
-serverSocket: ServerSocket -connectedClients: ArrayList<RemoteSite> -remoteHash: HashMap<Integer, RemoteSite> -port: Integer -numberOfUpdateAttempts: Integer = 0 -date: Data<Integer>[] -maxNumberOfUpdateAttempts: Integer = 50 -longFileWriter: FileWriter
+Server(id: Integer, port: Integer, numOfDataObjects: Integer) +runServer() +allFinished(): Boolean -createServerSocket(port: Integer) -handleServerSocketClosing()

RemoteSiteFromFile
-pathToFile: String
+RemoteSiteFromFile(pathToFile: String) +getServerInfo(): List<RemoteSite>

AbstractRemoteSiteProvider
#strToRemoteSite(line: String): RemoteSite +getServerInfo(): List<RemoteSite>

RemoteSite
-ipAddress: String -port: Integer -outStream: ObjectOutputStream -inStream: ObjectInputStream -socket: Socket -id: Integer = -1 -grantReceived: Boolean -lastRequest: AbstractRequset
+RemoteSite(ipAddress: String, port: Integer)

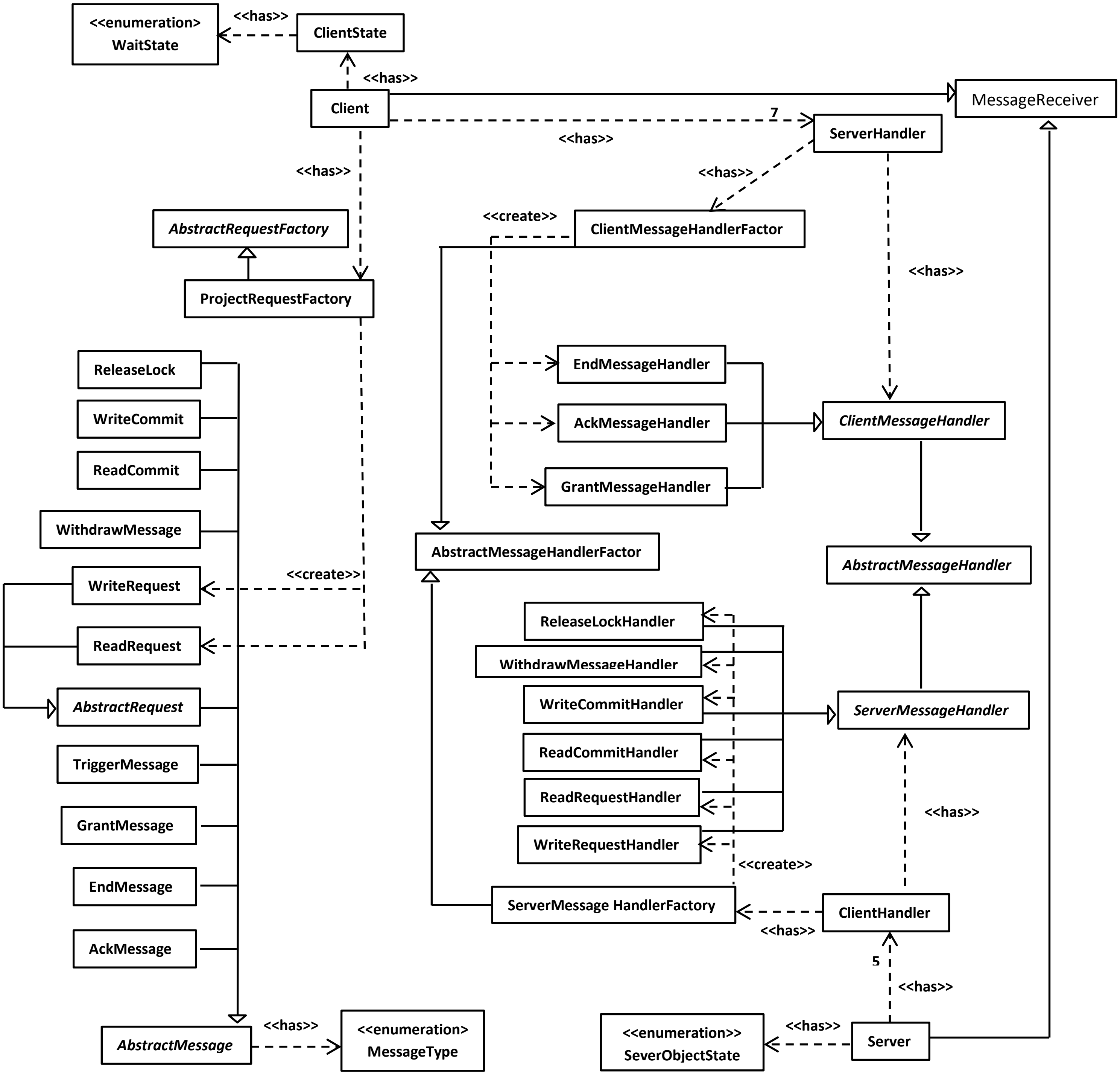
AbstractServerSelector
#servers: List<RemoteSite>
+selectServers(servers: List<RemoteSite>, request: AbstractRequest): List<RemoteSite> +AbstractServerSelector(servers: List<RemoteSite> +setServers(servers: List<RemoteSite>)



ProjectServerSelector

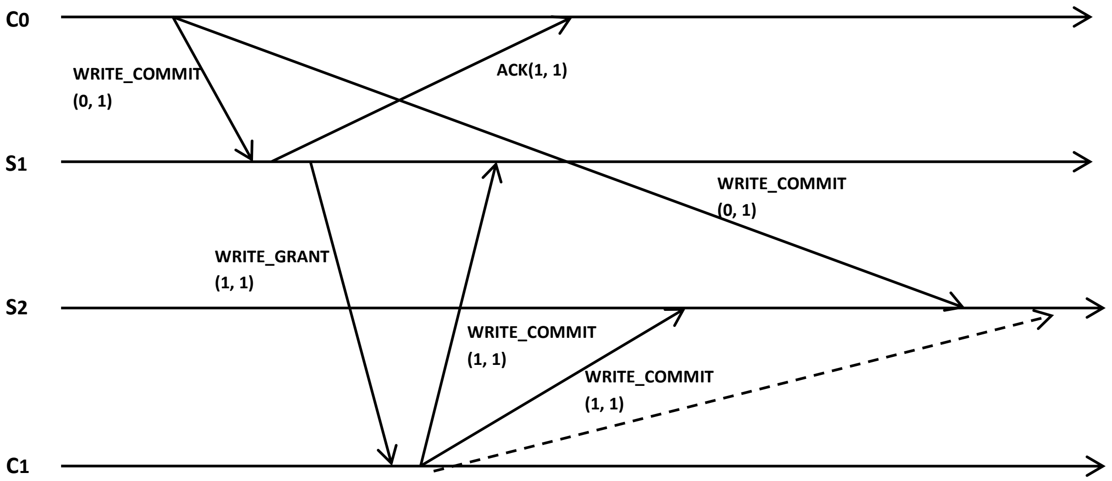
+selectServers(servers: List<RemoteSite>, request: AbstractRequest): List<RemoteSite>
+ProjectServerSelector(servers: List<RemoteSite>

9. Class Diagram (Relationship) for principal classes



10. Prevention to replicas in different sequence of updates for Dj

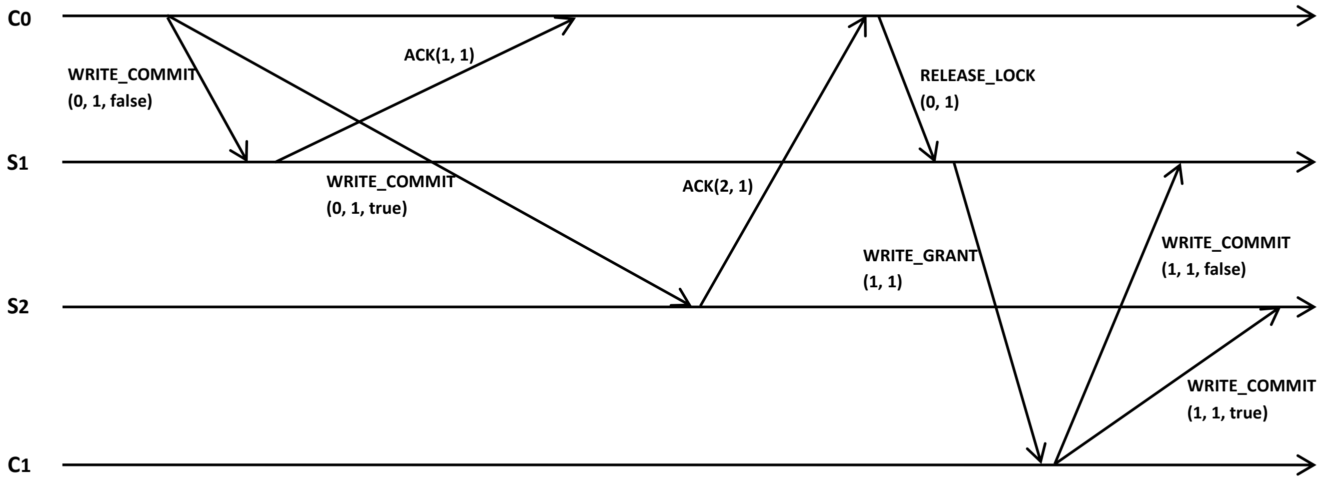
Possible Senario



*Every message carries sender's ID and sequence number of the message in order

Let us assume that a client(C0) sends WRITE_COMMIT to all servers including S1 and S2. After S1 received the WRITE_COMMIT from C0, S1 changes the state of data object(Dj) to NOT_LOCKED and sends WRITE_GRANT to C1 as WRITE_REQUEST from C1 was at the head of queue of pending requests. Then C1 sends WRITE_COMMIT to all servers. However, WRITE_COMMIT from C1 arrived earlier than WRITE_COMMIT from C0 to S2. This caused replicas different sequence of updates for Dj in S1(C0 -> C1) and S2(C1 -> C0). The dotted line shows desirable arrival of the message.

Our Solution



We include CanUnlock in WRITE_COMMIT which is true or false. We assume that S1 is one of the servers who caused C0 to send WRITE_COMMIT. When S1 received WRITE_COMMIT from C0, CanUnlock value is false. Thus, S1 cannot set state of Dj to NOT_LOCKED and it stays in WRITE_LOCKED.   This results in S1 not sending WRITE_GRANT to C1. Since write grant is based on quorum system, C1 cannot send WRITE_COMMIT unless Dj in S1 becomes NOT_LOCKED. After receiving all ACKs from every server, C0 sends RELEASE_LOCK to S1, then S1 changes the state of Dj to NOT_LOCKED and sends WRITE_GRANT to C1. This solution definitely prevents the violation described above because WRITE_COMMIT from C1 cannot be arrived in any sever ahead of WRITE_COMMIT from C0.