Table of Contents

Distributed Systems [41 points] *2*

Lamport's Logical Clock [3 points] 2

Lamport's ME Lock [20 points] 2

RPC Latency limits [8 points] 5

Active Networks [8 points] 6

System from Components [2 points] 7

Distributed Objects and Middleware [18 points] *8*

SPRING OS [12 points] 8

Java RMI [6 points] 9

Distributed Subsystems [40 points] *11*

DSM [15 points] 11

GMS [17 points] 13

DFS [8 points] 15

Distributed Systems [41 points]

Lamport's Logical Clock [3 points]

1. Students in the class are implementing a group chat application. They want to use Lamport's logical clock to order the communication. Assume no loss of messages, in-order delivery of messages between any two nodes, and variable communication latencies between nodes.

- Each message will be <sender, message> which will be broadcast to all the members of the group.
- A member can reply to a message that she receives which will also be broadcast to all the members of the group.
- The implementation should ensure that the original message and the replies to a specific message should appear in each member's user-interface in the order in which they are generated.

[No credit without an answer to the why/why not part.] (1 point) Will the use of Lamport's logical clocks to order the messages work for implementing this application?

[] Yes

[] No

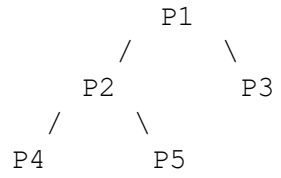
(2 points) Why or why not?

Answer:

No. If U1 sends a message and U2 replies to it, a third user may see U2's reply before U1's original message.

Lamport's ME Lock [20 points]

2. (14 points) Consider a N-node distributed system in which the nodes are arranged a binary tree topology. That is, a given node P can communicate ONLY with its parent node (if applicable) and ONLY to its children nodes (if applicable). The binary tree need not be a complete tree, and a given node may have at most one parent and at most 2 children.



The above diagram is an example topology. In this scenario, P2 can communicate ONLY with P1, P4, and P5.

Assume that there is no loss of messages, and the messages go in-order between any two nodes.

You are implementing Lamport's M.E algorithm in this system.
You should use three types of messages in your implementation: LOCK, ACK, UNLOCK. You should use Lamport's logical clock to order the communication events.
[Note: Your answer to the following questions, should work for any arbitrary N.]

(a) (4 points) List the actions at a node upon receiving a LOCK message.

(b) (4 points) List the actions at a node upon receiving an ACK message.

(c) (4 points) List the actions at a node upon receiving an UNLOCK message.

(d) (2 point) When does a node know that it has the LOCK?

Answer: (In all the answers below the timestamp associated with an incoming message is NOT changed in forwarding the message to others.)
(a)
LOCK message from parent:
- Enter the request into node's local queue at the appropriate spot (based on the timestamp)
- Forward the request to the children (if any)

LOCK message from a child:
- Enter the request into node's local queue at the appropriate spot (based on the timestamp)
- Forward the request to the parent (if there is one)
- Forward the request to the other child (if there is one)

(b)
ACK message from parent:
- If it is ACK for this node's request digest it locally
- If it is NOT an ACK for this node's request, forward it to the children (if any)

ACK message from child:
- If it is ACK for this node's request digest it locally
- If it is NOT an ACK for this node's request, forward it to the parent (if there is one) and to the other child (if there is one)

(c)
UNLOCK message from the parent:
- Remove the request from the node's local queue
- Forward request to the children (if any)

UNLOCK message from a child:
- Remove the request from the node's local queue
- Forward the request to the parent (if there is one)
- Forward the request to the other child (if there is one)

(d)
- Its request is at the top of its local queue
- It has received either LOCK requests later than its own, or ACKs from all the peer nodes

3. [6 points]
Lamport's ME algorithm as discussed in the paper and Kishore's lecture slides does not consider node failures. Your job is to extend the algorithm to accommodate node failures.
Assumptions for this problem:
- in-order delivery of messages between any two nodes
- the round-trip latency between any two nodes is a constant for all pairs of nodes
- every lock request is ACKed by a peer node
- messages can be lost due to node failure
- once a node is dead, it cannot rejoin the system
- resource protected by the lock is unaffected by node failures (i.e., the resource can be re-used by other nodes)

Your friend suggests an idea, namely, using timeout on messages to solve the problem. You have taken this idea and added a timeout mechanism on each message type (LOCK, ACK, UNLOCK).

(a) (2 points)
In your implementation what is the action at a node that wishes to acquire a lock?

(b) (2 points)
When can the node know it has the lock in your implementation?

(c) (2 points)

A node holding a lock dies. What will happen with your implementation using timeouts?

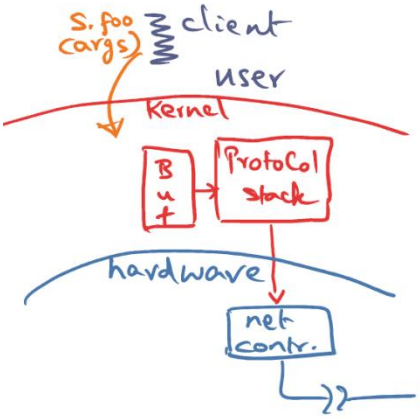
Answer:
(a) Node needing a lock does the following:
- Places its lock request at the appropriate slot in its local queue
- LOCK message to all the peers; set a TIMER > RTT
- If it does not receive an ACK from a node and the TIMER expires, all nodes from whom no ACK was received is marked as implicitly ACKed.

(b) Node knows it has the lock
- If its request is at the top of the local queue
- If it has received ACKs from all the peers either explicitly or implicitly as stated in (a)

(c) The system is deadlocked since the node holding the lock has to send unlock message to its peers and it is dead.

RPC Latency limits [8 points]

4. (6 points) The figure below shows the components involved in making an RPC call to a server.



In the absence of any optimizations, we say that the arguments of the call must be copied three times first by a client stub, then into the kernel, and finally into the NIC buffer.

(a) (2 points)
What is the reason for the first copy by the client stub?

Answer:

The kernel has no idea of the RPC semantics and where the arguments are in the client process's address space. The client stub "marshalls" the arguments into a contiguous set of bytes in the client address space and gives a pointer to the buffer to the kernel.

(b) (2 points)
What is the reason for the second copy by the kernel?

Answer:
The device driver that lives in the kernel's hardware address space and runs the NIC has no visibility to the client address space. So, the kernel has to copy the user-space buffer into the kernel for the NIC to have access to the arguments of the call.

(c) (2 points)
What is the reason for the third copy by the NIC?

Answer:
Most modern day NICs have an internal buffer from which the device electronics places the packet on the physical medium for transmission. So, the NIC has to DMA the contents of the kernel buffer into its internal buffer.

5. (2 points)
Thekkath and Levy suggest that a client spin wait after issuing an RPC call thus reducing the number of control transfers in the critical path of RPC latency. What is a potential downside to this approach?

Answer:
If the RPC takes a long time to return from the server, then processor resource is unnecessarily wasted on the client machine which could have been gainfully used for running some other processes.

Active Networks [8 points]

6. (4 points)
The principle of giving QoS guarantees to network flows, which is the vision behind "Active Networks" did not take off when it was first proposed. Give two reasons, at least one of them *technical*, for its lack of success at the time it was proposed.

Answer:
- software routing cannot keep up with the line rate of packets in the core of the network

- hard to convince network vendors (e.g., Cisco) to open up the routers for executing arbitrary code carried via the packets

7. (4 points)
Fast forward to the era of cloud computing. Active Networks has earned a new lease of life in the name of "software defined networking", and is widely used both in data center networks and wide-area networks. Give two technical reasons for its success.

Answer:
- Availability of chipsets that allows the software to set up the routing in a switch for a given network flow ONCE at the beginning, so that for the duration of the flow the routing happens through hardware table look up ensuring that the routers can keep up with the line rate
- Need for network traffic isolation to allow multiple tenants to use the data center resources both for performance and confidentiality

System from Components [2 points]

8. (2 points)
Why does the core principle in VLSI-based design of chips (namely, re-using components) does not carry over to the implementation of large software systems?

Answer:
- Hardware components compose seamlessly in a VLSI chip; software components do not compose similarly since there are overheads (data copying, access checking, etc.) in the composition.

Distributed Objects and Middleware [18 points]

SPRING OS [12 points]

1.[2 points]

(Answer True/False with justification) The spring kernel is responsible for network calls when a client on node A makes a procedure call to a server running on node B.

Answer:
False. The network proxies which are not part of the microkernel are responsible for network calls.

2. [2 points]
In the Spring kernel, a client domain can make a call to a server domain using the door mechanism. A server may wish to give differential treatment to cross-domain calls coming from different client domains. How is this accomplished?

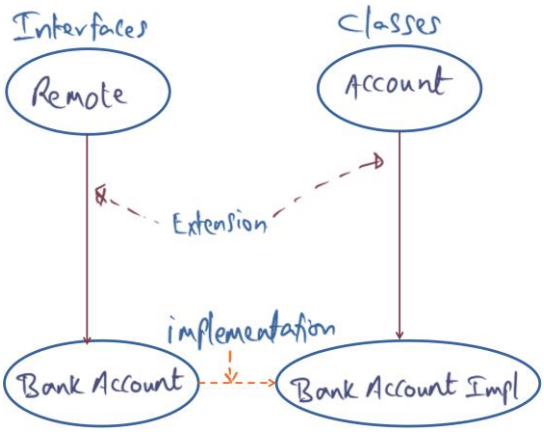
Answer:
- The server maintains an access control list (ACL)
- The door call comes into a front object of the server domain which would do the access check first before the server object is invoked

3. [8 points] Imagine the following scenario in Spring Kernel. A file server starts up on a node in the LAN. List the steps involved in the server being ready to take calls from the network clients.

Answer:
- The server object via the server-side stub) calls the "create" call provided by the subcontract mechanism to create a communication endpoint
- A server-side proxy object is created to listen on this communication endpoint
- The proxy object interaction with the server object is set up through the nucleus door mechanism
- The communication endpoint is registered by the subcontract mechanism in a nameserver for clients to discover the file server

Java RMI [6 points]

4. Ayrton is hired by Truist bank to implement a banking service in Java. He ends up with a system that looks like so:



When he instantiates the "Bank Account Impl" object, he finds that the service is not visible for clients on the Internet.

(a) [2 points]
With succinct bullets explain why.

Answer:
- The "Bank Account Impl" object may not have been registered with the RMI registry. Objects need to be bound to the registry to be accessible to clients.
- He has to make the service visible externally manually since he is not using key Java built in mechanisms. Ayrton needs to ensure that the "Bank Account Impl" object is exported to the RMI runtime so that it can receive remote calls.

(b) [4 points]
With succinct bullets how he can fix the problem so that the service becomes available on the Internet.

Answer:
- He should use the Java built-in mechanisms to do the heavy lifting automatically using "Remote Object" and "Remote Server" in creating the "Bank Account Impl" object
- He should use the Java name binding mechanism to publish the service so that clients can lookup and then make RMI calls to the service
- Utilize Java RMI Infrastructure: Automatically export the remote object when it is instantiated or explicitly export it
- Bind the service to the RMI registry

Distributed Subsystems [40 points]

DSM [15 points]

Context for this question is same as the previous question:

1. [8 points] You are helping a friend implement an efficient software DSM library in a local area network wherein the nodes are connected by 100 Gbps network links. The CPU architecture of the system uses a page size of 8KB. She is interested in achieving the following implementation objectives:

- O1: Avoid false sharing
- O2: Overlap computation with communication to increase performance
- O3: Allow concurrent execution of threads so long as they do not need mutual exclusion for shared memory access.
- O4: Reduce the number of communication events to increase performance

You offer the following helpful suggestions to her:

(a) (2 points) To couple the DSM implementation to the OS virtual memory management, you suggest her to use "page" as the unit of coherence maintenance and use "write-invalidate" coherence protocol. What is the downside of this suggestion with respect to the above objectives?

Answer:

- Violates O1: within a page, there might be a lot of different data structures. If coherence maintenance is being done at the level of a page, then we are invalidating copies of the page in several nodes to allow one node to make modifications to a portion of a page. Hence, a page appears to be shared from coherence maintenance perspective, even though programmatically it is not, leading to false sharing.
- Also violates O4: Considerable ping-ponging of a page when different parts of the page hosting different data structures need to be accessed.

(b) (2 point) To keep the implementation simple, you suggest using sequential consistency as the memory model. What is the downside of this suggestion with respect to the above objectives?

Answer:

- In SC every shared memory access has to be globally performed (e.g., invalidating the copies of the pages

before a node can write to it) during which the thread that modified the page has to be blocked; so, violates O2
- It also violates O3 since SC memory model does not distinguish between accesses to synchronization variables and data variables.

(c) (2 points) You change your mind and suggest that she use "eager" release consistency memory model. What is the downside of this suggestion with respect to the above objectives?

Answer:

- Violates O4: consistency actions sent to all peer nodes that have copies of pages modified in a critical section
- Violates O2: at release point a thread is blocked until all the consistency actions have been globally performed

(d) (2 points) You change your mind yet again and suggest using "lazy" release consistency memory model. What is the downside of this suggestion with respect to the above objectives?

Answer:

- Does not violate any of the objectives she had to start with

2. [7 points]

A cluster of 4 nodes connected by a LAN is used to deploy a program that uses the Treadmarks DSM software library for implementing parallelism. The program has two critical sections – CS1 governed by lock L1 and CS2 governed by lock L2. Processors P1, P2, P3 and P4 execute these critical sections and modify pages X and Y, which are locally present on all the nodes.

The following diffs are generated in the increasing order of time as shown below,

- Time T1: Processor P1 executes CS1 and generates a diff Xd1 for page X and a diff Yd1 for page Y
- Time T2: Processor P2 executes CS2 and generates a diff Xd2 for page X
- Time T3: Processor P3 executes CS1 and generates a diff Yd3 for page Y

Now at time T4, processor P4 wants to execute CS1.

(a) [2 points] What actions should happen before P4 starts executing CS1? Justify your answer.

Answer:

At the point of acquiring the lock, pages X and Y are invalidated since they are locally present.

(b) [5 points] While executing CS1, P4 accesses pages X and Y. What actions should take place to ensure correct execution?

Answer:

- Treadmarks DSM will get the pristine copy of pages X and Y from their owners (+1)
- DSM will get the diffs Xd1 and Yd1 from P1 (+1) and Yd3 from P3 (+1)
- DSM will apply the diff Xd1 to page X (+1) and the diffs Yd1 and Yd3 in that order to page Y (+2) to create updated copies of both pages for use by P4

(Note that diff Xd2 generated by P2 after executing CS2 is not relevant here and hence will not be fetched and applied)

-1 point if order of applying diffs Yd1 and Yd3 is wrong.

-1 point if Xd2 is also fetched and applied to X

GMS [17 points]

3. [1 point] Upon a page fault, GMS converts the VA to a UID. The UID includes the IP-ADDR. Whose IP-ADDR is this?

Answer:

This is the IP address of the file server that backs that virtual page.

4. [6 points]

Assume that there is a designated node (which never fails) in charge of additions/churns of the nodes participating in GMS. A new node joins the GMS.

(a) [2 points] List the actions that ensue.

Answer:

1. The designated node (master) recomputes the partitions of the UID space for each node (including the new node)
2. Each node sends the GCD entries that it is no longer responsible to the new node that has come online.

(b) [2 points] What data structures of GMS will get modified? Why?

1. POD because of the reassignment by the master
2. GCD since each node is responsible for managing a different subset.

(c) [2 points] What data structures of GMS will not get modified? Why?
PFD because the node that is currently housing a page in its DRAM does not change due to the addition of a node

5. [4 points] We say that the mechanics of GMS follows the principle of "think globally act locally". Explain with a couple of succinct bullets.
Answer:
- The only "global" action is at the beginning of every epoch when the weight vectors and minAge has to be computed by the initiator
- During the epoch itself all actions at a node are entirely based on local decision-making

6. [4 points] Consider the following scenario in GMS:
- Node N2 evicts a dirty page (P) due to memory pressure.
- It chooses Node N1 as the destination node to stash this page.
- N1 fails
- N2 page faults on P and needs to bring it back into its working set
- How does N1's failure affect N2?

Answer:

- By design, all pages kept in the global part of a node are clean.
- At the time of page eviction, N2 would have written the dirty page to disk before stashing it in the global part of N1
- N2 will simply get the page from the disk even though N1 is down
- Thus, N2 is unaffected by the node failure.

7. [2 points] The geriatrics algorithm chooses the node (say N1) with the maximum weight (i.e., the node that has most of the old pages that are likely to be replaced in the upcoming epoch) as the initiator for the next epoch. Give a hypothetical scenario when this strategy may not be the right one.

Answer:

- 1

DFS [8 points]

8. [8 points] xFS took some design decisions. We want you to justify those design decisions.

(a) (2 points) They decided to use LFS. Give two reasons to justify that design decision.

Answer:

- small write problem alleviated

- size of the logseg becomes a file system design parameter independent of the sizes of the files created by the users

(b) (2 points) They decided to use software RAID on a LAN. Give two reasons to justify that design decision.

Answer:

- Cheaper than hardware RAID
- Networks are fast, so starting parallel I/O on all the disks attached to the compute nodes on the LAN can be quick and lead to exploiting I/O parallelism

(c) (2 points) They decided to stripe a file NOT on all the disks but only a subset of the disks. Give two reasons to justify that design decision.

Answer:

- More concurrency for independent file I/O accessing different subsets of disks

- Failure of nodes will allow file system to continue to function with reduced availability since it will affect only the files that are hosted on the failed nodes.

(d) (2 points) They decided to decouple meta data management node for a file from the physical location of the file. Give two reasons to justify that design decision.

Answer:

- If a file server is hosting most of the hot files, then that server will get overloaded if it has to do the meta data management for all those files.
- Decoupling allows all the compute capacity (CPU and memory) in the cluster to be used for meta data management