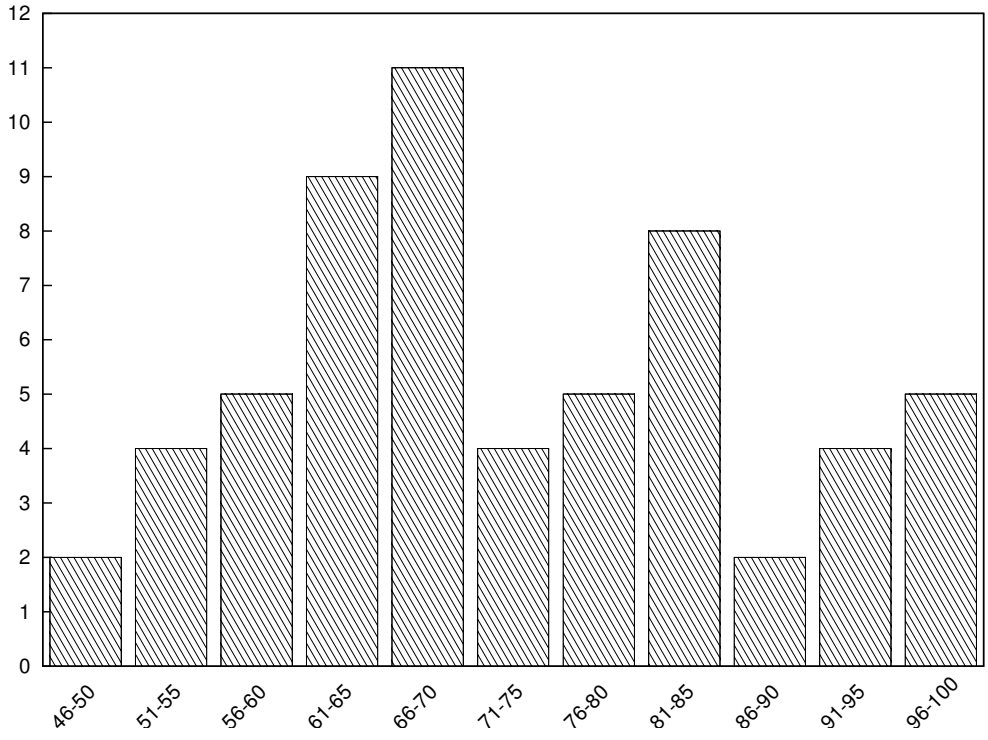
Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.824 Distributed System Engineering: Spring 2012

Quiz II Solutions

Grade histogram for Quiz II



min	=	47
max	=	100
median	=	69
average	=	73
σ	=	14

I SUNDR

1. [10 points]: Alice and Bob share a SUNDR server that uses the paper's strawman design (see Section 3.1). Each accesses SUNDR from his or her own client. Both mount the same SUNDR file system as "/sundr". One day, they perform operations on files x and y in the following order.

Operation Order	Alice	Bob
1	echo 0 > /sundr/x	
2	echo 0 > /sundr/y	
3		echo 1 > /sundr/x
4		echo 1 > /sundr/y
5	cat /sundr/y	
6		echo 2 > /sundr/x
7	cat /sundr/x	

Alice and Bob are the only users accessing the SUNDR server, and these are the only operations they send to the server. Each waits for the SUNDR server to indicate completion of their own previous command before running the next command. They also talk by telephone to ensure the order of operations shown above. For example, operation 6 starts after operation 5 has completed, and completes before operation 7 starts.

Alice sees 0 as the result of operation 5 (i.e. file y contains 0). The SUNDR clients do not detect any problem.

A. Which of 0, 1, and 2 could Alice see as a result of operation 7? For each value that is possible, give **all** possible operation histories that Alice's client could have downloaded from the SUNDR server during operation 7.

Answer: only 0 and 1 are possible. 0 results from the SUNDR server producing operations 1, 2, and 5 as the history. 1 results from the server producing 1, 2, 3, 5.

B. Suppose the SUNDR client didn't log fetches. Would this change affect the possible values that Alice could see as a result of operation 7? Explain why or why not.

Answer: Yes; this change would make 2 a possible value. The reason is that the SUNDR server could conceal operations 3 and 4 from Alice when she fetches during operation 5, but reveal 3 and 4 (and 6) when Alice performs operation 7.

2. [10 points]: Kate thinks that validating each user's most recent signature (see paragraph 3 of Section 3.1) is unnecessary. So she modifies her SUNDR client to check for the existence of her own last operation in the history, and to **validate only her own last operation and the last entry of the downloaded history**. Describe an attack that would succeed on Kate's modified SUNDR client, but that would not succeed with the original strawman SUNDR client.

Answer: Suppose there are three users, A, B, and C. The SUNDR server is malicious and is cooperating with user C (i.e. the server knows C's private key). The server could send A a history that included operations A4, A5, B6, C7, where A5 is A's last operation, B6 is an operation that the server made up (with an invalid signature), and C7 is properly signed. A would check only the signatures on A5 and C7, and thus would accept B6 as a valid operation by B even though it is not. This attack would not succeed with the paper's actual strawman design because A would check the signature on B6.

II E-mail anonymizer

Sammy is designing a simple e-mail anonymizer intended to hide the identities of senders from recipients. It consists of a single server. The server accepts incoming messages via https; the encryption prevents eavesdroppers from reading incoming messages. For each incoming message, the server deletes all headers other than Subject and To, adds an uninformative "From: nobody@nowhere.com" header, and forwards the message to its To address. Sammy's service becomes popular, and pretty soon his server is handling a few dozen messages per hour.

3. [5 points]: Sammy hears from his friends in 6.824 that delaying messages before forwarding them might help make his system more anonymous. Describe an attack which delaying would probably defeat.

Answer: Suppose the attacker can watch traffic leaving the host of one of Sammy's users, and can also watch traffic arriving at the 6.824 complaint server. If the attacker sees a message leaving the user's host with Sammy's server as IP destination, and shortly thereafter sees a message arriving at the 6.824 server from Sammy's server, the attacker could reasonably conclude that it was that user who submitted the complaint. If the server added delays, then this attack would be harder.

4. [5 points]: Sammy is trying to decide between two delaying schemes. Scheme A: delay each message for an hour before forwarding (e.g., if the message arrives at 10:15, delay it until 11:15). Scheme B: delay each message until the beginning of the next hour before forwarding (e.g., if the message arrives at 10:15, delay it until 11:00). Explain which is better, and why.

Answer: Scheme B is better. Scheme A preserves a predictable timing relationship between incoming and outgoing messages, while Scheme B does not.

III Web cache

Acme Enterprises runs a caching web proxy which its 1000 employees all use. The proxy can only cache 10 megabytes of web content. Acme is considering two plans to improve their web cache. Plan A: increase the size of the cache from 10 megabytes to 100 megabytes. Plan B: buy 9 more caching proxies, each with 10 megabytes of cache space, and divide the employees evenly among the 10 proxies; that way there would be 100 megabyte of cache space total. The proxies are independent (not cooperative).

One advantage of Plan B is that there would be more hardware resources in the caching system (CPU, network interfaces, etc), so that it would have higher throughput for local requests. One advantage of Plan A is that it would probably cost less.

5. [10 points]: Describe two significant ways in which Plan A would likely be better than Plan B (other than that mentioned above).

Answer: Plan A stores only one copy of each popular page, whereas Plan B might store one copy per caching proxy; thus Plan A might be able to store more distinct pages. Plan A is likely to fetch a popular page fewer times from the origin server. Plan A can cache web pages larger than 10 MB. Plan A is less subject to load balance problems.

IV Dynamo and PNUTS

One way in which Dynamo and PNUTS differ is in their support for atomic operations. PNUTS has an atomic test-and-set-write request that allows clients to implement things like atomic counter incrementing. When a client reads a value and then tries to modify it with test-and-set-write, the test-and-set-write only succeeds if the value hasn't been modified between the read and the test-and-set-write. For example, suppose two PNUTS clients try to increment a counter by reading it and using test-and-set-write to change its value to the read value plus one. If they do this concurrently, one will succeed, and the other will see a failed test-and-set-write and know that it should re-read the value and try again.

6. [10 points]: Why would it be difficult to add atomic test-and-set-write to Dynamo?

Answer: Atomic test-and-set-write (TASW) would require that Dynamo only apply the write if the latest version is the version mentioned as an argument to TASW. However, Dynamo doesn't have an easy way to decide if that's true: Dynamo doesn't sequence writes, so "latest version" is not well defined. Another way to look at it is that Dynamo allows updates in any partition, so that it might not be possible to communicate with the servers that hold various versions to find out if they've been updated recently.

V Frangipani

7. [10 points]: The Frangipani paper says that recovery is run for a crashed server when some other server tries to acquire a lock that the crashed server is holding. What would happen if a crashed server was holding no locks – would the fact that recovery might never be run for it be a problem? Why or why not?

Answer: If the server isn't holding any locks, it must have already written its entire log and all modified blocks to Paxos. Thus recovery wouldn't need to do anything, and there is no problem.

VI Harp

The Harp paper says that the primary sends a copy of its CP to the backups in each message, and that a backup does not apply operation i to the file system until it hears a CP $\geq i$, where i is the operation's index in the log (Section 4.2, paragraphs 3 and 4).

8. [10 points]: Consider an alternate design in which backups do not wait for the CP. Each backup applies operation i to the file system as soon as all operations through $i-1$ have completed, regardless of CP. This turns out to be a bad idea. Describe a sequence of events that would lead to a bad outcome with this alternate design, but that the real Harp handles well.

Answer: Suppose there are 5 servers, S1, S2, S3, S4, and S5. S1 starts out as the primary. S1 sends S2 operation X, S2 applies it to its disk, and then S1 and S2 crash. S3, S4, and S5 form a view and continue processing client operations. They have never heard of operation X. S2 comes back to life and tries to re-join the view. The problem is that its disk is now no longer usable, since it reflects operation X which (according to the replicated state machine as a whole) never occurred. This is not a problem in real Harp because the fact that S2 waits for CP means that it never applies an operation unless a majority of servers know about it, and thus that the operation will be known about in the next view.

VII Paxos

Here is pseudo-code for Paxos:

```
    --- Paxos Proposer ---

 1  proposer(v):
 2   choose n > n_p
 3   send prepare(n) to all servers including self
 4   if prepare_ok(n_a, v_a) from majority:
 5     v' = v_a with highest n_a; choose own v otherwise
 6     send accept(n, v') to all
 7     if accept_ok(n) from majority:
 8       send decided(v') to all


    --- Paxos Acceptor ---

 9  acceptor state on each node (persistent):
10   n_p     --- highest prepare seen
11   n_a, v_a  --- highest accept seen

12  acceptor's prepare(n) handler:
13   if n > n_p
14     n_p = n
15     reply prepare_ok(n_a, v_a)


16  acceptor's accept(n, v) handler:
17   if n ≥ n_p
18     n_p = n
19     n_a = n
20     v_a = v
21     reply accept_ok(n)
```

Suppose line 5 were changed to

```
v' = any non-nil v_a; choose own v otherwise
```

That is, suppose that a proposer picked any of the non-nil v_a's from acceptors responding to its prepare message, rather than the v_a with the highest n_a.

Assume that the v_a variable in each acceptor starts with value nil.

9. [10 points]: Describe a sequence of events where this change to Paxos would cause it to fail to agree correctly, whereas unmodified Paxos would correctly agree.

Answer: There are three acceptors, A1, A2, and A3. Proposer P1 sends propose messages to all, then accept(X) which only A1 receives, then proposer P1 stops. Proposer P2 sends propose messages to all, gets replies from just A2 and A3, sends accept(Y) which only A2 and A3 receive, then P2 crashes after hearing accept_ok from A2 and A3. Thus the agreed value is Y, though not everyone knows it. Proposer P3 now sends propose messages to all, gets value X from A1 and value Y from A2 and A3, chooses X, and sends accept(X) to all three acceptors. That is an error: Y has already been agreed on, so all future proposals must be for Y.

```
A1:  p1  aX          p3  aX
A2:  p1      p2  aY  p3  aX
A3:  p1      p2  aY  p3  aX
```

VIII Hypervisor-based Fault-tolerance

Bob is running Hypervisor Replication. Bob upgrades his PA-RISC CPU hardware to a new version that supports interrupt-free disk reads and writes. The new DISKREAD instruction reads a block from disk to memory and waits for the disk transfer to complete, while the DISKWRITE instruction writes a block of data from memory to disk, and waits for the write to complete. Bob plans to modify the operating system to use these two new instructions. He'll also modify the hypervisor so that the primary, at the end of each epoch, sends the backup the disk blocks read by each of the DISKREADs during the epoch. And he'll modify the hypervisor so that DISKWRITE on the backup doesn't do anything. He's excited because he thinks that the system will no longer have to wait until the end of each epoch for disk completion interrupts, and thus can execute multiple disk operations per epoch.

10. [10 points]: Explain why allowing multiple disk reads and writes per epoch will make fail-over hard or impossible.

Answer: There is no correct way for the backup to execute disk reads during the fail-over epoch. The old primary didn't complete the fail-over epoch, so it never sent the data read from disk to the backup. The backup cannot actually read the disk during the fail-over epoch, since the old primary might have written the same block later in the epoch. The backup cannot simply ignore disk reads during the fail-over epoch, since then it cannot execute conditional code (if statements) that depend on the data read from disk, and thus cannot execute the fail-over epoch identically to the primary – which it must do, since the old primary may have done all the disk writes for that epoch.

IX 6.824

11. [10 points]: If you could change one thing in 6.824 in order to improve the course for future generations, what would it be?

Answer: Less time for lab 6, more time for lab 7. More time for the final project. Less lecture time on big case studies, more time on core ideas. Don't require paper printouts during exams. Publish answers to all the paper questions. Document the lab code better.