1. Compare the advantages and disadvantages of implementing a concurrent server using process forking versus using multiple threads. Focus on aspects like resource isolation, communication overhead, and system protection.

Forking:

- Advantages: High resource isolation because each forked process gets their own address space. This also gives it better system protection compared to threads since they act independent of each other.
- Disadvantages: Since forked processes are isolated, it requires pipes or sockets to communicate with other processes, giving it a higher communication overhead.

Multi Threads:

- Advantages: Since threads share an address space, communicating with other threads is easy.
- Disadvantages: Threads share an address space under a process, so if one thread crashes, it might bring down the whole process.

2. The X Window protocol faces two main scalability challenges: bandwidth consumption and geographical distribution. Explain how compression techniques and caching can address these issues.

The X window protocol experiences higher latency over longer distances. With compression, it can help reduce the amount of packets that has to be sent and received in order for the protocol to work, thus reducing latency. This also reduces bandwidth as less data is being transferred. Caching further improves this by storing frequently used elements that doesn't require a lot of update locally, so it doesn't have to make a request every time it references it.

3. Describe how UNIX systems could implement strong mobility by extending fork() to work across machines. Consider aspects like process state transfer, resource handling, and network communication.

To implement strong mobility, we want the process being transferred to maintain its state as it moves across machines. This would include its address space like stacks and heaps for the processes memory, but if we want the

process to run seamlessly from where it left off, we also need a program counter to capture the next line of command to be executed. So the idea would probably be to somehow serialize all of these states and transfer it to the new device, and reconstruct the states locally, loading pc onto the cpu and allocating memory space. If the program accesses local resources such as files in the original machine, then those files probably has to be sent along with the rest of the states so that it can open properly on the new device.

4. Given a BitTorrent node with:
- Outgoing bandwidth: Bout
- Incoming bandwidth: Bin
- Single-seed connection constraint Calculate the maximum theoretical download speed when connected to one seed.

According to the textbook, if node P sees that node Q is downloading more than it is uploading, P will decrease the rate that it is sending data to Q. Thus the max download speed is likely determined by whichever bandwidth (outgoing and incoming) is lower, since BitTorrent will match the lowest.

5. Analyze how proxy-based replication transparency might affect server-side applications when multiple replicas receive the same call.

If multiple replicas receive the same call such as a write operation, it might create unnecessary work load because the same data is being written multiple times. And even if the data is the same, if the calls reach at different times, it can lead to message ordering inconsistencies.

6. For the procedure incr(i,i) where i=0:
- Analyze the final value of i using call-by-reference
- Compare with the result using copy/restore semantics

For call by reference, since it is directly accessing the integer in memory, any changes made to that reference will update the actual variable. Copy/restore on the otherhand creates a copy of the variable at the start of the function, manipulates that copied variable, and then updates the passed in variable to that value. If this function was incr(x, y) where x and y are independently

being incremented, pass by reference will return 2 because it increments once for x, and since y is still the same variable I, itll increment once again to 2. For copy /restore, x gets a copy of I, and y gets a copy of i, they increment separately, both resulting in 1. X gets copied back to I so I is now 1. Y also gets copied back to I, but since Y is also 1, I stays as 1.

7. Explain the step-by-step process of connectionless communication between client and server using UDP sockets, including packet handling and addressing.

UDP contains 8 byte of header that stores information such as the source port, destination port, length of message, and checksum. With UDP, there is no acknowledgement of message received. Once the server gets a request, it sends a continues stream of data to the receiver and just hopes most of it makes it. This makes it quick but unreliable. In fact, the order of the packets isn't even guaranteed. Each packet only knows where it is coming from and going to.

8. Describe how to implement synchronous communication primitives using only asynchronous receive operations and other asynchronous primitives.

We can implement a queue and only execute items in the queue one by one. That way we force the execution to stay in order even if the operations are async. The system won't have access to all the items at once, which simulates the blocking nature of a sync call.

9. Compare:
- Asynchronous RPC with wait
- Traditional synchronous RPC Analyze their equivalence and behavioral differences.

In a traditional RPC, the client sends a request and waits for the result, only continuing execution once the result returns from the call. Async RPC on the other hand also calls and waits, but instead of waiting for the result, it waits for an acceptance message from the server, letting the client know it has received the request. Then, even before the server finishes its procedure call,

the client continues with what it is doing, and gets the return from the server sometimes in the future.

10. Imagine two computers (A and B) sharing memory over a network. Computer A writes data to a shared memory location, and Computer B needs to read it.
a. What problems might occur when Computer B tries to read the data?

If B tries to read at the same time that A is currently writing, it might have been locked by the write operation, and therefore B is unable to read the data. Or worse, if it isn't locked, it starts reading unfinished data. Or, if B reads right before A starts writing the data, the data could be stale.

b. Name and explain two different ways to ensure Computer B gets the correct data

If the data is locked, B can wait until the lock lifts before reading. If the data is stale, B can check the time of the current data and the time of the data it has read previously. If it sees the data has updated, then it will update its data.

c. Which method would work better if:
- The data updates are frequent but small
- The data updates are rare but large

If the updates are frequent but small, stale data is the bigger issue since it will be overwritten a lot faster, which means periodically comparing timestamps and updating to the latest data is important. But if the data updates are rare and large, then a locking mechanism and a wait on the lock is better, since longer updates means it is more likely for someone to read at the same time it is being written, therefore we need the reader to wait for the write operation to finish.