Northeastern University – Silicon Valley

CS 6650 scalable Distributed Systems **Final Exam 8/15/22** [100 points]

This is an open book exam. You can refer PDF of Text book on laptop. Duration: **120** minutes
Please answer all questions, with brief but complete explanations. *Please keep your answers concise and to the point. Please write legibly. If I can't read it, I can't grade it. WRITE your NAME on all sheets of answers.*

1. Explain token based distributed mutual exclusion algorithm among N peer processes by Ricart-Agrawala. Consider how mutual exclusion is guaranteed and how the token is passed after a process has left the critical section. How many messages are passed in order a process to get permission to a critical section? Compare to the Central Server algorithm for mutual exclusion.

   The token based distributed mutual exclusion is that each process establishes a communication channel to the next process, which makes up a ring. The message and token will be passed through those channels in a single direction clockwise. Processes will record their states of being outside the critical section with RELEASED, if a process wants to enter the CS, it will change its state to WANTED and multicast the request to all the processes through the ring channel connection. Then the process marks the request's timestamp and waits for N – 1 replies. In the view of other processes, if they receive the request from the process, if the timestamp is smaller than its request's timestamp, then immediately reply to the process that sent the request. Otherwise, if certain process is holding the token, or it generated a new request with smaller timestamp, then queue the request without replying. After the process received N -1 replies, then means the process is allowed to enter the CS and change it state to HELD. If the process wants to leave the CS, it will change its state to RELEASED, and reply to the head of request in the queue. It tells other process that the CS right now is available. During the whole procedures, messages will be passed as less as 0 (the process just received the token at the time) or as much as N (the token just passed, will be sent, and go through the ring then come back). Compared to the central server algorithm, which token is managed by the server, the message sending is fast, but the central server consumes extra processes, performance bottleneck, and has the single point of failure problem.

2. What is Consensus? Provide 2 examples which require Consensus. Failure and Asynchrony are two problems which make Consensus difficult to achieve. Why? Explain using one use case.

   Consensus is all correct processes of the system are agreed on a decision. This can tolerate fault that some processes are failed in the system, but the remainders are correct, and the system can continue to work properly.
   NTP service should be required consensus for time sensitive applications. Because all correct processes should be consensus in time, otherwise the system will have a gap timestamp.
   Storage service should be consensus on data for all replicas. If they are not chosen the same value in the consensus procedures, the storage will become inconsistent.

A process failed that means it will be absent for vote for a decision or take a decision and execute it. For example, PAXOS, if more than half of processes are failed in the system, the consensus is going to be never reached, so the system will fail either.
Asynchronous is the problem because each process can reply to the consensus message at arbitrary times, we are not able to clarify if a process is working slow or it just crashed.

3. Explain the two phase commit protocol, and its uses. How would you modify the two phase commit protocol to make sure it can function in the presence of Byzantine failures?

   The two phase commit protocol will be used in distributed transaction applications, like bank debit, credit services. In the first phase, coordinator will call all participants to ask them if they are able to commit a transaction. Then participants will verify the transaction object locally and vote for commit or abort. In the second phase, the coordinator collects the votes and analyze. If all votes for Yes, then the coordinator will send a do commit message to participants, otherwise, send do abort message to participants. Then participants will get the message from the coordinator and execute commit or abort then send ACK back to the coordinator means the action has done.
   I will assign a TID for each transaction object, it should be unique and can be verified. Then, add timeout mechanism for the coordinator, if a participant cannot respond in the given time, then timeout the participant.

4. (a) Explain Byzantine failure and Fail-stop Failure, and how they are different.
   (b) Do they require a leader election algorithm? If so, which algorithm would you use and why?

   (a) Byzantine failure is that a process can give any arbitrary issues, like response wrong content in messages or commit wrong values.
   The Fail-stop failure is that a process gets this failure will fail to work from now on.
   The difference is that a process has the Byzantine failure can still work but incorrectly, instead of just shut down or crash as Fail-stop failure.
   (b) Yes, they require a leader election algorithm that all processes agree on a choice. The bully algorithm. Because the algorithm for synchronous system, we can construct a reliable failure detector to figure out the Fail-stop processes and if Byzantine failure happened in a process, when processing the leader election algorithm, it has three types of messages, and the process with Byzantine failure cannot properly figure out and response correctly to those messages, then the process will be replaced by other processes.

5. What is Concurrency control? Why is using mutual exclusion alone not a good idea to achieve Concurrency control? What are the 3 additional, different ways to achieve Concurrency control?

   Concurrency control is involved in concurrent transactions. If two and more transactions are happening at the same time and intend to update the same objects, there will have two problems, lost update problem and inconsistent retrievals. To solve the problems, we have to control the transactions that occurred concurrently.

Mutual exclusion is not good in concurrency control because it will only allow one process to go to the critical section and do updates. Then, other transaction can go to the critical section and do updates. It will reduce the throughput of the system and too many processes are waiting.

1. Lock. If objects involved in a transaction, then the objects will be locked until the transaction has done.
2. Timestamp methods. Assign unique and monotonic timestamps for the transactions that come to the system. We can order transactions based on total, partial, etc ordering.
3. Optimistic methods. This will work efficiently for the system that rare operations, low conflicts.

6. (a) What is Group communication in Distr. Systems? Describe 4 scenarios where it is necessary.
   (b) Explain the below key requirements for Group communication in Distributed Systems - Atomicity and Ordering. What are they and why are they required, how are they achieved?

   (a) Group communication in distributed systems is exchanging file, Java object, data among multiple processes, those processes can communicate at once. It can be used as unicast, multicast, and broadcast. (1) Leader election. (2) Distributed consensus. (3) Failure detector. (4) Two phase commit transaction. Those algorithms are required multiple or all processes to participate, then processes will frequently send messages across different processes. The group communication is the basic thing that guarantee the messages can be delivered to different processes at the same time.
   (b) Atomicity is that a message multicast or broadcast to a group, it will only have one result that all members of the group receive the message or none of them do. It is important because it can guarantee that a message has been seen by all members of the group, or the message will be dropped because one of the members didn't see it. It will ensure that the message only have one state so that if the message carry the operations that require members to do some change in their local resource, it will not result in inconsistency because all members received the message. We can achieve it by using 2 phase commit protocol. If we find a process doesn't response an ACK for the last message, then send abort message to all members to ignore the last message.
   Ordering is that the group multicast or broadcast a message, the message will send to members in order. For example, the global ordering is members deliver messages in order of they were sent. Causal ordering: if a multicast of message 1 is happened before message 2, then m1 is delivered before m2 to all members. This is important because for the case that a process require other processes to report their states and ask them to do some operations based on their states. If multicast without ordering, the operation messages will be sent before the report state message will crash those members. We can achieve ordering by timestamp those messages. Therefore, we can figure out the happened before relations. Then we can maintain a priority queue to enqueue those timestamped messages to be multicast with casual order.

7. An international coalition with 4 members A, B, C and D must exchange highly secure Financial Disclosure Documents (FDD) over the Web. Show a basic design of this secure document exchange system which can ensure confidentiality, authentication of all parties and integrity checking, where you use public key cryptography at least once.

   It is easy to achieve this by using asymmetric encryption. We can use a public key to encrypt the documents and the 4 members has their own unique private key to decrypt the documents that encrypted by the public key. They system will be like, assume A start sharing a document. The system will generate a public key to encrypt the document. Then A wants to share the document with B, C, D.

So, our system will send the encrypted document with the public key to B, C, D. Once they received the document, they can use their private keys to decrypt the document and see. If they want to send some documents back, they can also use the public key to encrypt the document and send. Then A can use it own private key to decrypt the documents from B, C, D. We can use RSA for the asymmetric encryption.

8. The Internet does not currently offer any resource reservation or quality of service management facilities. How do the existing Internet-based audio and video streaming applications achieve acceptable quality? What limitations do the solutions they adopt place on multimedia applications?

   We can use traffic shaping, which can buffer at the destination. Then, the media data display at destination after some delays like 5 seconds. Therefore, we have enough buffered data to continuously display to users, instead of stuck streaming. Then, use traffic shaping to control the data delivery rate to solve the latency of the Internet. Finally, we can compress media to deliver fast in reduced size.

   Limitations are the size or quality of the video; audio cannot be large and good. Because if deliver high resolution video like 4K movies, the system still works slowly. For interactive applications like multiplayer video games, they require high OS and network performance, the solution is not good either.

9. You're in a moving car crossing state boundaries and watching a popular NFL game via a news channel video. In this context, please explain these concepts – what it is and how it operates in this use case:
   a) Association
   b) Leaky Bucket Algorithm
   c) Buffering

   a) In a moving car, we may use wireless radio network to watch the game in news channel, the spontaneous association is to secure the channel between us and the news side by securely exchanging a session key between our device and the news side device. The data communication in the channel will be encrypted. It can operate by comparing the secure hashes of the keys of the two devices, running the Diffie-Hellman protocol to figure it out.
   b) Leaky Buckey Algorithm is to control the data stream that will not flow higher than rate R. And the bucket is the size of buffer that defines the max burst that the data stream can incur without elements lost. It also limits the time that an element can keep in the bucket. Running the algorithm that will base on bandwidth, delay, loss those factors to do the operations.
   c) Buffering is to buffer the data in the destination for limited size. This can be operated by checking the size of current bucket and remove the those old and filled up with new data.