Qing Peng

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 idea of Ricart-Agrawala's algorithm is that exclusion is conferred by obtaining a token in the form of a message passed from process to process in a single direction around the ring. If a process does not require to enter the critical section when it receives the token, then it passes the token to its neighbor. The process that requires the token waits until it receives it, and retains it. When the process exit the critical section, it sends the token to its neighbor. There are 0 messages(when it has just received the token) to N messages(number of processes) (when it has just passed on the token) passed to get permission to a critical section. The synchronization delay between one process's exit from the critical section and the next process's entry is from 1 to N message transmissions.

The central server algorithm can tolerate the crash failure of a client process that does not hold or request the token. While the Ricart-Agrawala's algorithm can be adapted to tolerate the crash failure of such a process by taking it to grant all requests implicity.

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.

In the context of group communication, the consensus problem is to agree on some value when there are domain of the values in question.

Failure is a problem to make consensus, because when a process fails, it might propose a incorrect value If other processes just take this incorrected value, then there will be a lot of incorrected values in the group, and the consensus is not reached. And when there are more than half processes fails, they cannot reach a consensus using PAXOS algorithm.
For example, there are 5 processes in the system, and 3 of them fails in a consensus process, if the algorithm is to pass a proposal when there are more than half processes agreeing on the value, then this proposal fails.

The failure detectors are not reliable in asynchronized systems, which makes it a problem to reach consensus.
For example, in a asynchronized system, there are 5 processes, 3 of them faces with slow network, so the proposals in the 3 processes arrives after the limited time, makes the failure detectors detect the failures. Thus this proposals fails. However, the 3 processes might receives the proposal after some time.

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 failure

The two phase commit protocol is designed to allow any participant to abort its part of a transaction, and if one part of a transaction is aborted, then the whole transaction fails. In the first phase of the protocol, each participant votes to commit a transaction. Each participant saves in permanent storage all of the objects that is has altered in the transaction, together with its status. In the second phase of the protocol, every participant carries out the joint decision. If any one participant votes to abort, then the decision must be to abort the transaction. If all the participants vote to commit, then the decision is to commit the transaction.

To make the protocol works in the presence of Byzantine failure, we can replicate the coordinator, so the failure of the coordinator will not make the protocol fail. Also we can run a Byzantine aggrement algorithm among the coordinator replicas.

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) The Byzantine failure is that one process fails, but the process keeps running and produces the incorrect results.
The fail-stop failure is that one process fails, and then the process stop running.

(b) Yes they do.
To solve the fail-stop failure, we can use the select a process as a leader, and it counts the votes from all the processes, when there are more than half votes from processes, the consensus is reached. Otherwise, the proposal fails.
To solve the Byzantine failure, we need to have more than 3n+1 processes, and select one of them as a commander. To reach the agreement, in the first round, the commander sends a value to each of the lieutenants. And in the second round, each of the lieutenants sends the value it received to its peers. If a process sends a value different from it receives, then we can know that it is a faulty process and do not count its vote. After all the process vote, we can reach the consensus.

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 to cope with conflicts between operations in different transactions on the same object.
If we use mutual exclusion alone to achieve concurrency control, when we lock the process when one thread executes, other threads cannot execute the same process, and the efficiency is low. Also, in distributed systems, we do not have shared memory or a common physical clock, so we cannot solve mutual exclusion problem using shared variables.
We can use locks, optimistic concurrency control and timestamp ordering to achieve concurrency control.

6. (a) What does a DNS server do? What are the inputs it accepts and outputs it provides?
(b) Which machines (nodes) in a Distributed Systems use DNS services?
(c) When might a DNS server provide multiple answers to a single name look up, and why?

(a) The DNS server maps domain names to the attributes of a host computer: its IP address, the type of entry, the length of time the host's entry will remain valid and other information. The inputs is the domain names, the outputs is the resource ID, including the IP address, the type of entry, length of time the host's entry will remain valid, port number, pathname and other information about the server.

(b) Each server that provides public services uses the DNS service. Each domain has a domain name server handling its requests. When users look up a domain name in the DNS server, the DNS server finds the IP address and port number for the user, and the user can access the server of the given address.

(c) A DNS server always provides multiple answers to a single name. Because the server might know multiple addresses of a given domain. It increases the availability of the DNS service to returning all the addresses.

7. (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 is the communication among all the machines in a distributed system. In group communication, all members of a group must receive copies of the messages sent to the group.

In our projects, in order to provide the same services from the 5 different servers, we need to use group communication to synchronize the values in the servers. Also, there are other cases that group communication should be uses, like conference calls, digital messages in group members, DNS servers and so on.

(b) In group communication, the central comcept is that of a group with associated group membership, processes may join or leave the group. Processes can send a message to this group and have it propogated to all members of the group, so the ordering and atomicity are important.
We need to guarantees the relative ordering of messages delivered to multiple destinations. To achieve this, we can use ordered multicast with the option of one or more of the folowwing properties: FIFO ordering, causal ordering and total ordering.
The atomicity of the message is important because we do not want the message being corrupted in the transmission. We can use reliable transmission protocol such as TCP to guarantee the atomicity of the message.

8. (a) What is a Distributed File System (DFS), and how is it different from a Distributed Storage system?
(b) List and explain any four design considerations for DFS, with detail on how they are addresses.

(a) Distributed file systems is the file system in distributed systems, which support the sharing of information in the form of files and hardware resources in the form of persistent storage throughout an intranet. A distributed file system store data in hierarchical file and folder strucure, which is more efficient to access and manage than distributed storage system. However, if we do not want to save a lot of different files in a system, a distributed storage system might be more efficient since it saves efforts to do file management.

(b) Flat file service: The flat file service implement operation on the contents of files, and uses unique file identifiers(UFID) to refer to files in all requests for flat file service operations. When the flat file service receives a request to create a file, it generages a new UFID for it and returns the UFID to the requester.

Directory service: The directory service provides a mapping between text names for the files and their UFIDs. Users can access the file by quoting the test name to the directory service. To add new file names to directories and to obtain UFIDs from directories, the directory service provides the functions needed to generate directories. Directories hold references to other directories.

Client module: The client module runs in each client computer, integrating and extending the operations of the flat file service and the directory service under a single application programming interface in a user-level program. The client module also holds information about the network locations of the flat file server and directory server processes.

Flat file service interface: The flat file service interface is a PRC interface used by client modules. There are several operations in the interface: Read(FileId, i, n), Write(FileId, i, Data), Create(), Delete(), GetAttributes(FileId), SetAttributes(FileId, Attr). A fileId is invalid if the file that it refers to is not present in the server processing the request or if its access permissions are inappropriate for the operation requested.

9. An international coalition with 4 members A, B, C and D must excahnge 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.

In networks that integrated for management purposes, this need can be met by a secure key service that issues session keys in the form of challenges.

In the login session, first is the login session setup. Clients sends its username and password(private key) to the server to request for TGS ticket, and after the Authentication Service verify the client’s identification, it sends the TGS ticket to the client. We can incorporate the use of public-key certificates there.

Secondly, it comes with the server session setup, the client request for the server ticket, and the Authentication database grants the service to return a server ticket to the client.

Finally, it is the DoOperation. The client request encrypted with session key to another server S, and the server S gives back an encrypted with session key.

We use three kinds of security object: ticket, authenticator and session key.

The client processes must possess a ticket and a session key for each server that they use. On login, users are authenticated by the AS, using a network-secure variation of the password method, and the client process acting on behalf of the user is supplied with a ticket-granting ticket and a session key for communicating with the TGS. Then the original client process and its descendants can use the ticket-granting ticket to objain tickets and session keys for specific services from the TGS.

When a valid ticket has been obtained from the authentication service, the client can use it to communicate with the ticket-granting service to objain tickets for other servers until the ticket expires. And for each transaction, the ticket can only be used for once.