Question 1: Give two examples to demonstrate the importance and the need of synchronization mechanism between processes in distributed systems.

1. Distributed Database System

In a distributed database system, data is stored and managed across multiple nodes or servers. Each node may handle a subset of the data and perform various operations on it. To ensure data consistency and avoid conflicts, a synchronization mechanism is essential to prevent wrong data. It can effect importantly to the business data.

1. Distributed File Sharing System

Consider a distributed file sharing system where multiple users can upload, download, and modify files stored across different nodes. To maintain data integrity and prevent conflicts, synchronization mechanisms are crucial in such systems.

For example, user A and user B both want to modify the same file simultaneously. Without synchronization, they may end up overwriting each other's changes, leading to data corruption and loss of information.

Question 2: Compare Network Time Protocol and Berkeley algorithm.

Both Network Time Protocol (NTP) and the Berkeley algorithm are used to synchronize clocks in distributed systems.

* Time Synchronization Approach:

NTP: Network Time Protocol is designed to synchronize clocks in a distributed system by exchanging time information between servers and clients. It uses a hierarchical structure of time servers, with higher-level servers synchronizing with reference clocks and lower-level servers synchronizing with higher-level servers.

Berkeley algorithm: The Berkeley algorithm follows a master-slave model, where one node is designated as the time master, and the other nodes are time slaves. The master periodically polls the slaves, collects their clock values, and calculates an average time, which is then distributed back to the slaves.

* Clock Adjustment:

NTP: NTP adjusts the clock frequency and time offset of the clients to bring them in sync with the reference time source. It uses various algorithms and statistical techniques to estimate and compensate for network delays and clock drift.

Berkeley algorithm: The Berkeley algorithm adjusts the clock by applying a time correction based on the average time calculated by the master node. Each slave adjusts its clock to match the average time received from the master.

* Communication Model:

NTP: NTP operates using a client-server model, where clients request time information from servers. NTP packets are exchanged between the client and server to synchronize the clocks.

Berkeley algorithm: The Berkeley algorithm works in a decentralized manner, with the master node periodically requesting clock values from the slave nodes. The slaves respond with their clock values, and the master calculates the average time.

* Accuracy and Precision:

NTP: NTP aims for high accuracy and precision in time synchronization. It utilizes advanced techniques, such as timestamp interpolation, clock filtering, and selection algorithms, to achieve better accuracy.

Berkeley algorithm: The Berkeley algorithm focuses more on achieving a reasonable level of clock synchronization rather than high precision. It provides synchronized time within an acceptable range, but it may not achieve the same level of accuracy as NTP.

Question 3: What is the typical characteristic of synchronization algorithm for wireless networks?

The usual critical path in determining network delays.

Question 4: What is the difference between physical synchronization and logical synchronization?

* Physical clock:

It is a physical process and also a method of measuring that process to record the passage of time. For example, the rotation of the Earth measured in solar days. Most of the physical clocks are based on cyclic processes such as a celestial rotation.

* Logical clock

It is a mechanism for capturing causal and chronological relationships in a distributed system. A physically synchronous global clock may not be present in a distributed system. In such systems a logical clock allows global ordering on events from different processes.

Question 5: In using the Vector Clock concept for enforcing causal communication, answer the following questions:

1. List two conditions the receiving process use to check whether the message satisfies causality.
2. See an example of 3 processes as shown below. We assume that the vector clock values at the beginning are (0;0;0). What are the vector clock values for 4 points X1, X2, X3, and X4?

A diagram of a triangle

Description automatically generated

* 1. 2 conditions:

ts(m)[i] = VCj[i] + 1

ts(m)[k] <= VCj[k] for all k != i

with i is the index of sending process, j is the index of receiving process

X1(0,1,0) => before sending, P1 updates its vector clock

X2(0,1,0) => P0 receive message from P1, it will update its vector clock too

X3(0,0,0) => P2 receive message from P0 but because of casual forcing, it will not update its vector clock right after receiving but storing it

X4(0,1,0) => P2 receive message from P1 and update its vector clock, after that it will update to (1,1,0)