Homework-1

Mohan Srinivas Naga Satya: 001821918

How might the clocks in two computers that are linked by a local network be synchronized without reference to an external time source? What factors limit the accuracy of the procedure you have described? How could the clocks in a large number of computers connected by the Internet be synchronized? Discuss the accuracy of that procedure.

A: Clocks of two computers in a network of distributed systems are synchronized only by passing on messages. But the very fact that there is no notation of single global time limits the accuracy of clock synchronization between the computers in the distributed system. Also, as passing messages between systems is the way to synchronize clocks, some factors effecting the synchronization are:

- **Clock Skew:**
  Though the clocks are initially set to be equal, due to counting time at different rates, the times diverge, and significance difference is built up over-time. This is because the underlying oscillators are subject to different physical variations because of which their frequencies of oscillations differ.
- **Temperature**:
  Not considering the physical variations, the frequency of oscillations differs with temperature, and hence develop a drift between the clocks.

Clocks in large number of computers connected by Internet can be synchronized by always referring to the physical systems which have a small deviation from Coordinated Universal Time. Network Time protocol (NTP) is an algorithm which selects the physical machine with most accurate time in comparison to Coordinated Universal time and propagates the same to the other peers in the network using the UDP protocol.

The accuracy of NTP varies from 1ms in local network to 10ms in public internet. It is designated to mitigate variable network latency, but the accuracy of the algorithm is affected in asynchronous networks. Apart from that, network congestion can also cause inaccuracies of up to 100ms in this algorithm as it depends on communicating the information over UDP.

A server process maintains a shared information object such as the BLOB object of Exercise 1.7. Give arguments for and against allowing the client requests to be executed concurrently by the server. In the case that they are executed concurrently, give an example of possible 'interference' that can occur between the operations of different clients. Suggest how such interference may be prevented.

A: In a distributed system, there would be the need for a server to handle requests from multiple clients and processing them concurrently. For doing so, the server needs to be constantly available, provide updated information, and maintain partition of information being sent to multiple clients.

So, some of the benefits of single server multi-client architecture in a distributed system are:

- **Throughput**:
  Serving requests of multiple clients will allow the system to clear the tasks in the pipe and in turn increasing the throughput. Higher throughput reflects on the capability of the system to process multiple tasks and allowing the server to provide faster speeds of execution. For example, in an image uploading scenario, higher throughput provides higher bandwidth and in turn provides, faster upload speeds.
- **Shared-Resource**:
  In a use-case where a server stores shared information object, it provides clients making requests with a common point of resource and decreases the load on the clients. In doing so, an application can be shared among multiple clients and share the same state.

Similarly, there are some disadvantages of server multi-client architecture:

- **Data Consistency**:
  In an environment, where multiple client requests are being executed by a single server, data consistency needs to be maintained. In scenarios where multiple requests wants to access the same resource, concurrency can lead to delivering resource with incorrect state to both the requests.
- **Availability**:
  As there is a single server handling multiple client requests, in a distributed system, that would imply a single point of failure and hence can lead to loss or unavailability of resources.

A use-case in which there would be possible interference would be when the clients are accessing the same resource. The server is supposed to maintain the state of resources in a concurrent processing environment. So, when multiple clients request to perform the same operation, the concurrent environment choses the execution of resource associated with that operation in an order. A possible way to prevent the interference is to provide a multi-threaded environment to the server where the resources can be shared between the client requests. In doing so, the state of the resource can be maintained between the clients and hence, can avoid any interreference from the requests.

A service is implemented by several servers. Explain why resources might be transferred between them. Would it be satisfactory for clients to multicast all requests to the group of servers as a way of achieving mobility transparency for clients?

A: A service when implemented by several servers will have its resources distributed among the different servers. So, in order to satisfy a request from the client, a server needs to communicate the state of request to make use of the available resources in the other servers.

For example, let us consider a distributed file sharing system. In such a system, different files are stored in different servers by being divided into several chunks. So, when an incoming request for a file is transmitted, the several chunks from different servers are collected together or transferred into a single server and transmitted back to the client.

In achieving mobility transparency, it is necessary for the client/user to be unaware of the movement of information within the service. And this needs to be satisfied without effecting the operations of the user and the applications that are running. So, in this case, multicasting all the requests to the group of servers allows the client to be unaware of the servers location and allows the servers to transmit/communicate the response by transferring resources between them.