1.

a)

Distributed system is a system that all of its components located at networked computers communicate and coordinate only by passing messages. The World Wide Web is a system for publishing and accessing resources on the Internet. Users using the World Wide Web are located all over the world, users should be able to retrieve resources from any conformant server. And all the resources are provided from different machines in different locations. Those machines communicate through networks.

Challenges in distributed systems: Heterogeneity, openness, security, scalability, failure handling, concurrency, transparency, quality of service and so on.

The openness determines whether the system can be extended and reimplemented. In distributed systems, it is a challenge to keep adding new resources, and make it available for a lot of client programs.

Security contains three components: confidentiality(protection against disclosure to unauthorized individuals), integrity(protection against alteration or corruption) and availability(protection against interference with the means to access the resources).

Scalability means the system is effective when there is a significant increase in the number of resources and the number of users.

b)

Transparency is the concealment from the user and the application programmer of the separation of components in a distributed system. In the World Wide Web, users and programmers are not supposed to feel the separation of components they are accessing.

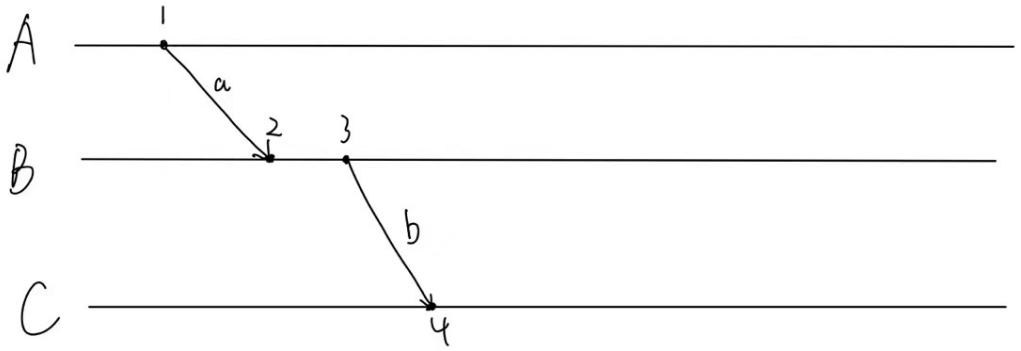
Access transparency: it should be the same if users are accessing the servers locally or accessing the service remotely.

Location transparency: users do not need to know their physical or network location like which building or IP address they are in to access the Web service.

Concurrency transparency: several processes can access the same shared resources at the same time.

2.

a)



The mechanism of Lamport timestamps is based on happened_before ordering. Since we only care about the ordering of the events, we do not need to care about the physical clock. We denote the timestamp of event a at A with 1, and each following event is issued at process with timestamp + 1, so the timestamp of event a at B is 2. Since B waits for 2 units

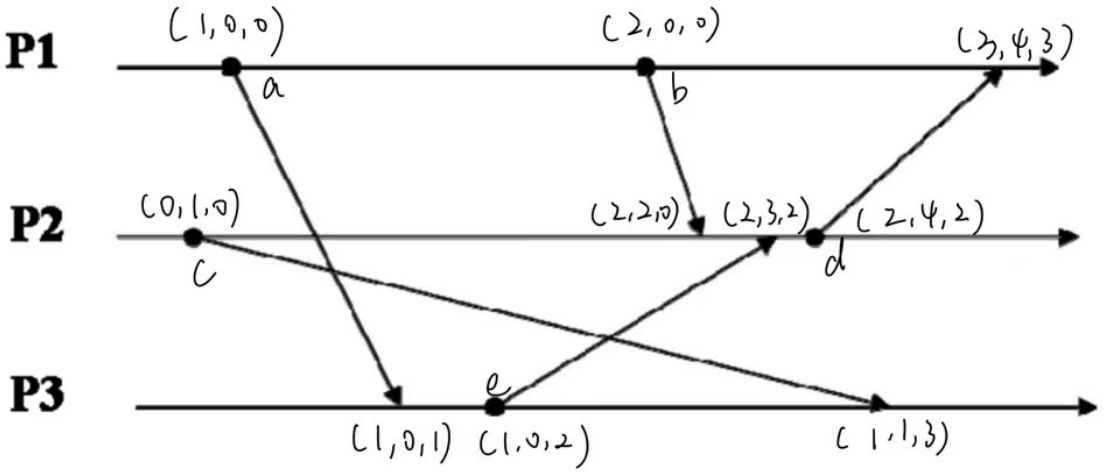
of time to start another event, we can denote the timestamp of event b at B with 3, and timestamp of event b at C with 4.

b)
In distributed systems, it is important to measure time in different systems. We need to know the accurate time that a particular event occurred. And some algorithms depend on clock synchronization. But since every system has its own physical clock, this process can be problematic. And thus the order of events can be false. In order to synchronize the processes' clocks, firstly we need to know the order of events. And Lamport Logical Clocks can help find the order of timestamps. The algorithm is based on happened-before ordering.

3.

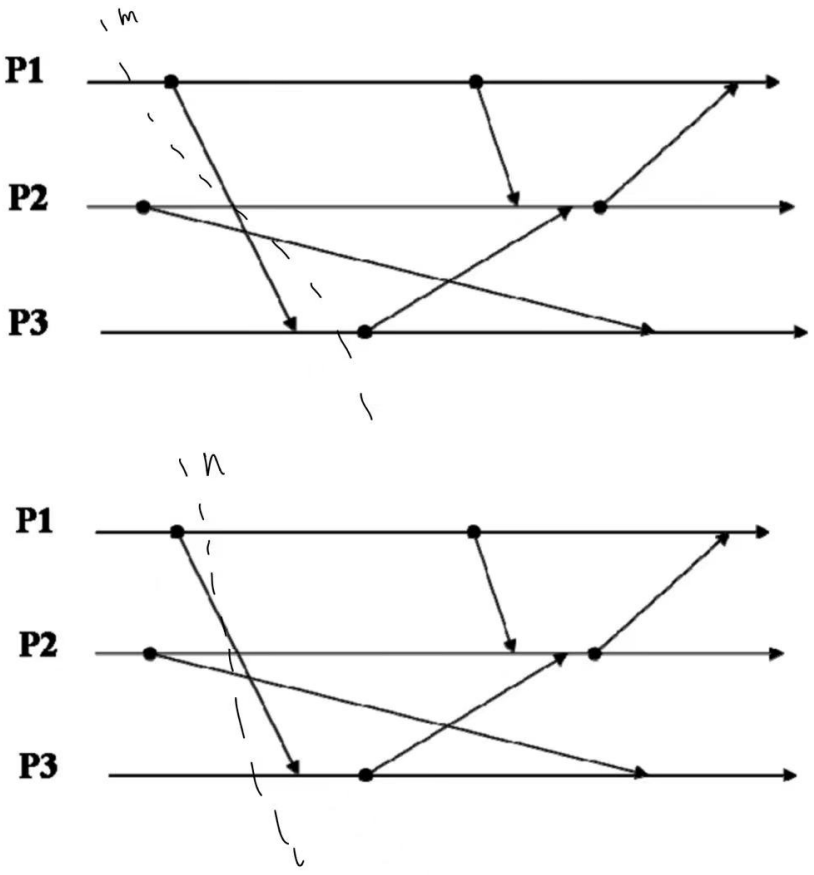
a)

a(1,0,0)
b(2,0,0)
c(0,1,0)
d(2,4,2)
e(1,0,2)



Initialize the vector in each process with (0,0,0), every time an event happens, increase the corresponding value of the process by 1, and when a process receives an event from another process, sync the maximum values of all dimensions.
In Lamport timestamps, we only know the order of the timestamps each event happens, but since the Chandy Lamport algorithm records the values in all processes, we can know the order of the current event compared to all the other processes.

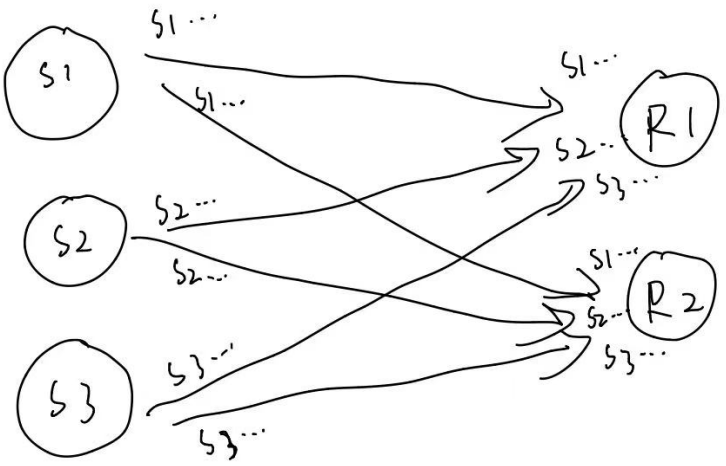
b) consistent cut: n
inconsistent cut: m



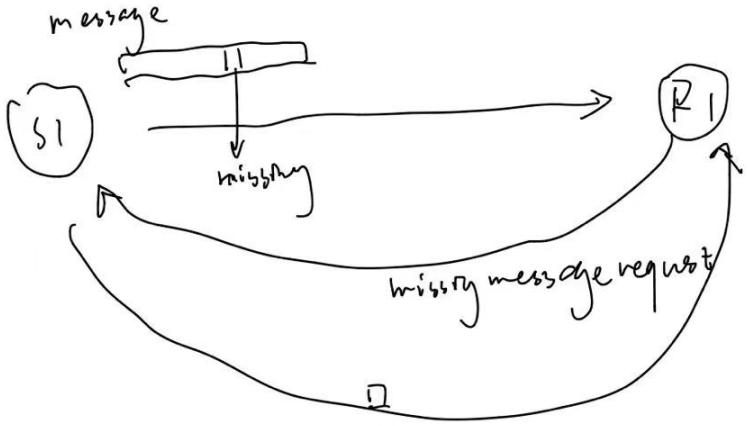
in cut m, the event from P1 to P3 has not started in the beginning of m, but then m records part of the event, it violates the happened-before rule.
in cut n, before the beginning of the cut n, the event from P1 to P3 started, the event from P2 to P3 started, and n does not record any event that have not started. So it is a consistent cut.

c)
In consistent cuts, all the events recorded in the cut are happening at the time, and those event contains some information of the processes, which can be used to construct global states. Synchronization is important in distributed systems, with Chandy Lamport algorithm, we can find the states of all the processes at a given time. And keep the data of the process consistent at the same moment.

4.
Since there might be multiple senders, there should be identifiers attaching to each message to specify which sender it is. So each message should contain a sequence number, and the recipients record the sequence number and check if it receives messages from all senders.



Since there are only a small proportion of messages dropped generally, it is not efficient to resend all the messages, we can only resend the messages lost. So after senders send the messages, they store all the messages, and if the recipient requests missing messages, the senders can send the missing message to the recipients.



Since the messages may not arrive in sender order, the senders should attach a sequence number for each message, so when the recipients receive the messages, they can sort the messages according to the sequence number.

5.

a)

Application layer: the layer interacts with end-users to provide support for services like email, file transfers and data sharing.

Presentation layer: the layer converts data formats between applications and networks.

Session layer: the layer manages sessions between servers to coordinate communication.

Transport layer: the layer provides mechanisms such as error control, flow control, congestion control to keep track of the data packets, check for errors and duplication.

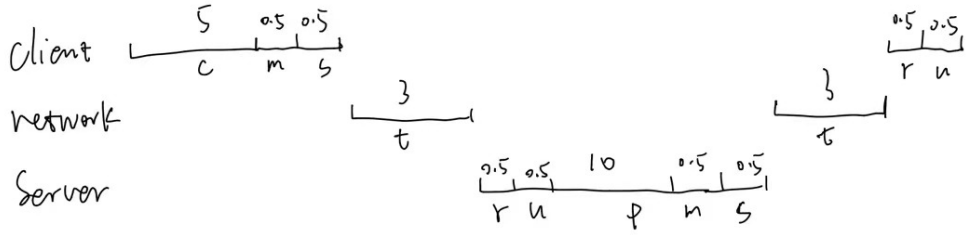
b)
The communication between machines is complex, in order to make each task easier to analyze, we must divide the task between different layers, so, we need to follow a protocol at each layer, this technique we used to call protocol layering. This layering allows us to separate the services from the implementation.

A machine transmits data to another machine down through all the layers to the physical layer where the data is put onto the network cabling, and then sent to the physical layer of the receiving computer where the process reverses and the data travels up through the layers to the application layer of the receiving computer.

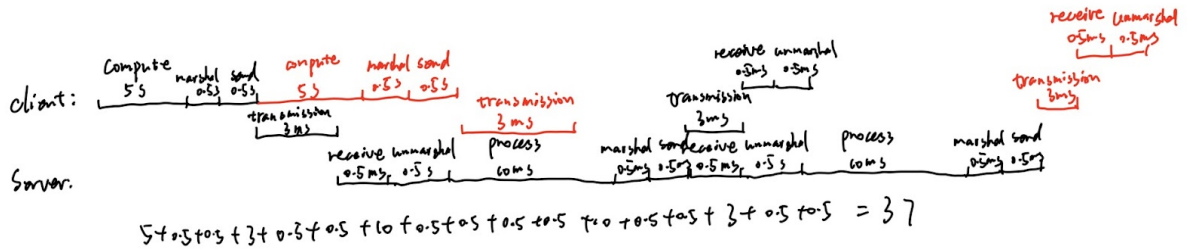
6.
a)
differences between RPC and RMI:

1. RPC is C bases, RMI is Java based
2. RPC supports procedural programming paradigms, RMI supports object-oriented programming paradigms. If we want to make each service an object, we can only use RMI to program.
3. The parameters passed in RPC are ordinary or normal data, we can only pass ordinary data, which is not practical since the data in Web service is complex. parameters in RMI are objects, which is easier to pass data.
4. RPC creates more overhead, RMI creates less overhead
5. RPC is less efficient
6. the development cost is huge in RPC, the development cost is less in RMI
7. there is a problem of versioning in RPC
8. RPC does not provide any security, RMI provides client level security

b)
client compute time + marshal time + OS sending time + unmarshal time + server processing time + marshal results + OS sending time + network transmitting time + OS receiving time + unmarshal time = 5 + 10 + 4 * 0.5 + 4 * 0.5 + 2 * 3 = 25 ms
two request: 2 * 25 = 50ms



The first request is in black line, the second request is in red line. The total process time is 37ms.



$5+0.5+0.3+3+0.5+0.5+10+0.5+0.5+0.5+0.5+0+0.5+0.5+3+0.5+0.5 = 37$

Yes, there is a need for asynchronous invocation if the client and server processes are multi-threaded