Distributed Systems

Tutorial 1: Introduction to Distributed Systems

1. Describe the key features of a Distributed System?

   A DS connects users to resources in a transparent, open and scalable way.

   Key features: transparency, openness, scalability.

   Transparency: The actual structure of the system is hidden from the user.

   Openness: New components/resources can be added at runtime.

   Scalability: The system can continue to grow as demand increases.

2. What is the difference between the following terms: *host*, *site*, *server*, *client?*

   A *host* is a particular computer attached to a DS.

   A *server* is a type of host that makes services available to other hosts.

   A *client* is a type of host that accesses the services made available by servers.

   A *site* is a geographical location where numerous *hosts* are located.

3. What is the difference between a *loosely-coupled* and a *tightly coupled* Distributed System?

   In a *loosely-coupled* system, the system components are not specifically designed to work closely together to satisfy a particular goal. In a *tightly coupled* system, each system component is designed with the others in mind so they are intended to work together.

4. The lecture notes state "a Distributed System has, by default, no global clock". Explain why this is the case.

   A DS is made up of multiple separate computers, each of which has its own clock. Thus in an *n*-node DS, there will be *n* separate, independent clocks. A global clock must be added separately if required.

5. In what way are the concepts of *redundancy* and *reliability* related?

   Redundancy is used to provide reliability. If a component of the DS fails, reliability is generally maintained by substituting a replacement copy of that component into the system without affecting users. This is generally only possible where redundant components are already built into the system.

6. Can you think of a situation where the use of concurrency for a processing task might be a disadvantage?

Concurrency incurs a management overhead. For example, a task needs to be divided into subtasks, these need to be allocated to different hosts, and the outputs of each subtask may need to be combined into a final outcome for the task. If the task is small, this management overhead may be greater than the time savings concurrency provides, thus causing the task to take longer to run.

7. Information security often consists of two things: security of information being transmission (encryption) and security of access to a resource (proof of identity). Describe a type of distributed system where these are both important and say why they are important.

   Sample answer: In a bank's DS, users can use ATMs to interact with the system. Encryption is necessary so that confidential information cannot be intercepted by other network users (e.g. bank account balances, login details, etc). Proof of identity is necessary (by means of a bank card and a PIN) to ensure that the person accessing the informant (and the money) is indeed the account holder.

8. Why is middleware important (from the point of view of the developer of a distributed application)?

   Middleware provides a layer of abstraction for application developers. This means that the developers do not need to be concerned with differences in Operating Systems or hardware on the hosts they deploy their DS application on. Other services such as message passing or access to shared data will frequently be provided also, meaning that the developer does not need to implement these separately.

9.
   a. p_1, q_1
   b. p_3, q_2, q_3, r_4
   c. r_1, r_2, r_3

10. Logical time requires that the processes within a distributed system have a mechanism to ensure that the order of operations in the system is preserved. Thus it is only the time relative to one another that is important. The real time is not required as it does not affect how the system performs.

11. It is not a consistent snapshot. The message sent at r_3 and received at p_3 has been recorded only as being received but not being sent. The snapshot represents a state that it was never possible for the system to be in (when this message had been received but not sent).

12. Timestamps are as follows for each process:

Process A	**Process B**	**Process C**	**T**
12	10	8	10
24 (sends to B)	20	16	20

36	30 (receives from A: no change since 30>24)	24	30
48	40	32	40
60 (sends to C)	50	40	50
72 (sends to A)	60	61 (receives from A: change timestamp since 60>48)	60
84	73 (receives from A: change timestamp since 72>70)	69	70
96	83	77 (sends to A)	80
108 (receives from C: no change since 77<108)	93 (sends to C)	85	90
120	**103**	**94 (receives from B: change timestamp since 93=93)**	100