Distributed System Structures

Practice Exercises

16.1 Most WANs employ only a partially connected topology. Why is this so?
Answer: Cost. A fully connected network requires a link between every node in the network. For a WAN, this may be too costly as communication links between physically distant hosts may be expensive.

16.2 Under what circumstances is a token-passing network more effective than an Ethernet network?
Answer: A token ring is very effective under high sustained load, as no collisions can occur and each slot may be used to carry a message, providing high throughput. A token ring is less effective when the load is light (token processing takes longer than bus access, so any one packet can take longer to reach its destination) or sporadic.

16.3 Why would it be a bad idea for gateways to pass broadcast packets between networks? What would be the advantages of doing so?
Answer: All broadcasts would be propagated to all networks, causing a *lot* of network traffic. If broadcast traffic were limited to important data (and very little of it), then broadcast propagation would save gateways from having to run special software to watch for this data (such as network routing information) and rebroadcast it.

16.4 Discuss the advantages and disadvantages of caching name translations for computers located in remote domains.
Answer: There is a performance advantage to caching name translations for computers located in remote domains: repeated resolution of the same name from different computers located in the local domain could be performed locally without requiring a remote name lookup operation. The disadvantage is that there could be inconsistencies in the

name translations when updates are made in the mapping of names to IP addresses. These consistency problems could be solved by invalidating translations, which would require state to be managed regarding which computers are caching a certain translation and also would require a number of invalidation messages, or by using leases whereby the caching entity invalidates a translation after a certain period of time. The latter approach requires less state and no invalidation messages but might suffer from temporary inconsistencies.

16.5 What are the advantages and disadvantages of using circuit switching? For what kinds of applications is circuit switching a viable strategy?
Answer: Circuit switching guarantees that the network resources required for a transfer are reserved before the transmission takes place. This ensures that packets will not be dropped and their delivery would satisfy quality of service requirements. The disadvantage of circuit switching is that it requires a round-trip message to set-up the reservations and it also might overprovision resources, thereby resulting in suboptimal use of the resources. Circuit switching is a viable strategy for applications that have constant demands regarding network resources and would require the resources for long periods of time, thereby amortizing the initial overheads.

16.6 What are two formidable problems that designers must solve to implement a network-transparent system?
Answer: One such issue is making all the processors and storage devices seem transparent across the network. In other words, the distributed system should appear as a centralized system to users. The Andrew file system and NFS provide this feature: the distributed file system appears to the user as a single file system but in reality it may be distributed across a network.
Another issue concerns the mobility of users. We want to allow users to connect to the "system" rather than to a specific machine (although in reality they may be logging in to a specific machine somewhere in the distributed system).

16.7 Process migration within a heterogeneous network is usually impossible, given the differences in architectures and operating systems. Describe a method for process migration across different architectures running:

a. The same operating system

b. Different operating systems

Answer: For the same operating system, process migration is relatively straightforward, as the state of the process needs to migrate from one processor to another. This involves moving the address space, state of the CPU registers, and open files from the source system to the destination. However, it is important that identical copies of the operating system are running on the different systems to ensure compatibility. If the operating system are the same, but perhaps different versions are running on the separate systems, then migrating processes must be

sure to follow programming guidelines that are consistent between the different versions of the operating system.
Java applets provide a nice example of process migration between different operating systems. To hide differences in the underlying system, the migrated process (i.e., a Java applet) runs on a virtual machine rather than a specific operating system. All that is required is for the virtual machine to be running on the system the process migrates to.

16.8 To build a robust distributed system, you must know what kinds of failures can occur.

a. List three possible types of failure in a distributed system.

b. Specify which of the entries in your list also are applicable to a centralized system.

Answer: Three common failures in a distributed system include: (1) network link failure, (2) host failure, (3) storage medium failure. Both (2) and (3) are failures that could also occur in a centralized system, whereas a network link failure can occur only in a networked-distributed system.

16.9 Is it always crucial to know that the message you have sent has arrived at its destination safely? If your answer is *yes*, explain why. If your answer is *no*, give appropriate examples.
Answer: No. Many status-gathering programs work from the assumption that packets may not be received by the destination system. These programs generally *broadcast* a packet and assume that at least some other systems on their network will receive the information. For instance, a daemon on each system might broadcast the system's load average and number of users. This information might be used for process migration target selection. Another example is a program that determines if a remote site is both running and accessible over the network. If it sends a query and gets no reply it knows the system cannot currently be reached.

16.10 Present an algorithm for reconstructing a logical ring after a process in the ring fails.
Answer: Typically distributed systems employ a coordinator process that performs functions needed by other processes in the system. This would include enforcing mutual exclusion and—in this case of a ring—replacing a lost token.
A scheme similar to the ring algorithm presented in Section 18.6.2 can be used. The algorithm is as follows:
The **ring algorithm** assumes that the links are unidirectional and that processes send their messages to the neighbor on their right. The main data structure used by the algorithm is the **active list**, a list that contains the priority numbers of all active processes in the system when the algorithm ends; each process maintains its own active list.

a. If process P_i detects a coordinator failure, it creates a new active list that is initially empty. It then sends a message *elect*(i) to its right neighbor, and adds the number i to its active list.

b. If P_i receives a message *elect*(j) from the process on the left, it must respond in one of three ways:
   1. If this is the first *elect* message it has seen or sent, P_i creates a new active list with the numbers i and j. It then sends the message *elect*(i), followed by the message *elect*(j).
   2. If $i \neq j$, that is, the message received does not contain P_i's number, then P_i adds j to its active list and forwards the message to its right neighbor.
   3. If $i = j$, that is, P_i receives the message *elect*(i), then the active list for P_i now contains the numbers of all the active processes in the system. Process P_i can now determine the largest number in the active list to identify the new coordinator process.

16.11 Consider a distributed system with two sites, A and B. Consider whether site A can distinguish among the following:

a. B goes down.

b. The link between A and B goes down.

c. B is extremely overloaded and its response time is 100 times longer than normal.

What implications does your answer have for recovery in distributed systems?

Answer: One technique would be for B to periodically send a *I-am-up* message to A indicating it is still alive. If A does not receive an *I-am-up* message, it can assume either B—or the network link—is down. Note that an *I-am-up* message does not allow A to distinguish between each type of failure. One technique that allows A better to determine if the network is down is to send an *Are-you-up* message to B using an alternate route. If it receives a reply, it can determine that indeed the network link is down and that B is up.

If we assume that A knows B is up and is reachable (via the *I-am-up* mechanism) and that A has some value N that indicates a normal response time, A could monitor the response time from B and compare values to N, allowing A to determine if B is overloaded or not.

The implications of both of these techniques are that A could choose another host—say C—in the system if B is either down, unreachable, or overloaded.