Network Structure

Practice Exercises

19.1 Why would it be a bad idea for routers 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 (such as network routing information), then broadcast propagation would save routers from having to run special software to watch for the information and rebroadcast it.

19.2 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 can be performed locally without requiring a remote name-lookup operation. The disadvantage is that there may be inconsistencies in the name translations when updates are made in the mapping of names to IP addresses. These consistency problems can be solved by invalidating translations (by the server telling the caching clients that a change has occurred), which require state to be managed regarding which computers are caching a certain translation and also require a number of invalidation messages. Alternatively, the problems can be solved 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.

19.3 What are two formidable problems that designers must solve to implement a network system that has the quality of transparency?

Answer:

One 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).

19.4 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 are: (1) network link failure, (2) host failure, (3) site failure 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.

19.5 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 the program sends a query and gets no reply, it knows the system cannot currently be reached.

19.6 A distributed system has 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 an ***I-am-up*** message to A indicating that it is still alive. If A does not receive an ***I-am-up*** message, it can assume that either B—or the network link—is down. Note that an ***I-am-up*** message does not allow A to distinguish between the types of failure. One technique that allows A to better determine if the network is down is to send an ***Are-you-up*** message to B using an alternate route. If A receives a reply, it can determine that, indeed, the network link is down and 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 the values, 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.