Distributed Systems: Concepts and Design

Edition 3

By George Coulouris, Jean Dollimore and Tim Kindberg
Addison-Wesley, ©Pearson Education 2001

Chapter 3 Exercise Solutions

3.1 A client sends a 200 byte request message to a service, which produces a response containing 5000 bytes. Estimate the total time to complete the request in each of the following cases, with the performance assumptions listed below:

i) Using connectionless (datagram) communication (for example, UDP);

ii) Using connection-oriented communication (for example, TCP);

iii) The server process is in the same machine as the client.

[Latency per packet (local or remote,
incurred on both send and receive):5 milliseconds
Connection setup time (TCP only):5 milliseconds
Data transfer rate:10 megabits per second
MTU:1000 bytes
Server request processing time:2 milliseconds
Assume that the network is lightly loaded.]

3.1 Ans.

i) UDP: 5 + 2000/10000 + 2 + 5(5 + 10000/10000) = 37.2 milliseconds

ii) TCP: 5 + 5 + 200/10000 + 2 + 5(5 + 10000/10000) = 42.5 milliseconds

iii) same machine: the messages can be sent by a single in memory copy; estimate interprocess data transfer rate at 40 megabits/second. Latency/message ~5 milliseconds. Time for server call:

5 + 2000/40000 + 5 + 50000/40000 = 11.3 milliseconds

3.2 The Internet is far too large for any router to hold routing information for all destinations. How does the Internet routing scheme deal with this issue?

3.2 Ans.

If a router does not find the network id portion of a destination address in its routing table, it despatches the packet to a *default address* an adjacent gateway or router that is designated as responsible for routing packets for which there is no routing information available. Each router's default address carries such packets towards a router than has more complete routing information, until one is encountered that has a specific entry for the relevant network id.

3.3 What is the task of an Ethernet switch? What tables does it maintain?

3.3 Ans.

An Ethernet switch must maintain routing tables giving the Ethernet addresses and network id for all hosts on the local network (connected set of Ethernets accessible from the switch). It does this by 'learning' the host addresses from the source address fields on each network. The switch receives all the packets transmitted on the Ethernets to which it is connected. It looks up the destination of each packet in its routing tables. If the destination is not found, the destination host must be one about which the switch has not yet learned and the packet must be forwarded to all the connected networks to ensure delivery. If the destination address is on the

same Ethernet as the source, the packet is ignored, since it will be delivered directly. In all other cases, the switch tranmits the packet on the destination host's network, determined from the routing information.

3.4 Make a table similar to Figure 3.5 describing the work done by the software in each protocol layer when Internet applications and the TCP/IP suite are implemented over an Ethernet.

3.4 Ans.

Layer	Description	Examples
Application	Protocols that are designed to meet the communication requirements of specific applications, often defining the interface to a service. network representation that is independent of the representations used in individual computers. Encryption is performed in this layer.	HTTP, FTP, SMTP, CORBA IIOP, Secure Sockets Layer, CORBA Data Rep
Transport	UDP: checksum validation, delivery to process ports. TCP: segmentation, flow control, acknowledgement and reliable delivery.	TCP, UDP
Network	IP addresses translated to Ethernet addresses (using ARP). IP packets segmented into Ether packets.	IP
Data link	Ethernet CSMA CD mechanism.	Ethernet MAC layer
Physical	Various Ethernet transmission standards.	Ethernet base-band signalling

---

3.5 How has the end-to-end argument [Saltzer *et al.* 1984] been applied to the design of the Internet? Consider how the use of a virtual circuit network protocol in place of IP would impact the feasibility of the World Wide Web.

3.5 Ans.

Quote from [www.reed.com]: This design approach has been the bedrock under the Internet's design. The e-mail and web (note they are now lower-case) infrastructure that permeates the world economy would not have been possible if they hadn't been built according to the end-to-end principle. Just remember: underlying a web page that comes up in a fraction of a second are tens or even hundreds of packet exchanges with many unrelated computers. If we had required that each exchange set up a virtual circuit registered with each router on the network, so that the network could track it, the overhead of registering circuits would dominate the cost of delivering the page. Similarly, the decentralized administration of email has allowed the development of list servers and newsgroups which have flourished with little cost or central planning.

3.6 Can we be sure that no two computers in the Internet have the same IP addresses?

3.6 Ans.

This depends upon the allocation of network ids to user organizations by the Network Information Center (NIC). Of course, networks with unauthorized network ids may become connected, in which case the requirement for unique IP addresses is broken.

3.7 Compare connectionless (UDP) and connection-oriented (TCP) communication for the implementation of each of the following application-level or presentation-level protocols:

i) virtual terminal access (for example, Telnet);

ii) file transfer (for example, FTP);

iii) user location (for example, rwho, finger);

iv) information browsing (for example, HTTP);

v) remote procedure call.

3.7 Ans.

i) The long duration of sessions, the need for reliability and the unstructured sequences of characters transmitted make connection-oriented communication most suitable for this application. Performance is not critical in this application, so the overheads are of little consequence.

ii) File calls for the transmission of large volumes of data. Connectionless would be ok if error rates are low and the messages can be large, but in the Internet, these requirements aren't met, so TCP is used.

iii) Connectionless is preferable, since messages are short, and a single message is sufficient for each transaction.

iv) Either mode could be used. The volume of data transferred on each transaction can be quite large, so TCP is used in practice.

v) RPC achieves reliability by means of timeouts and re-trys. so connectionless (UDP) communication is often preferred.

3.8 Explain how it is possible for a sequence of packets transmitted through a wide area network to arrive at their destination in an order that differs from that in which they were sent. Why can't this happen in a local network? Can it happen in an ATM network?

3.8 Ans.

Packets transmitted through a store-and-forward network travels by a route that is determined dynamically for each packet. Some routes will have more hops or slower switches than others. Thus packets may overtake each other. Connection-oriented protocols such as TCP overcome this by adding sequence numbers to the packets and re-ordering them at the receiving host.

It can't happen in local networks because the medium provides only a single channel connecting all of the hosts on the network. Packets are therefore transmitted and received in strict sequence.

It can't happen in ATM networks because they are connection-oriented. Transmission is always through virtual channels, and VCs guarantee to deliver data in the order in which it is transmitted.

3.9 A specific problem that must be solved in remote terminal access protocols such as Telnet is the need to transmit exceptional events such as 'kill signals' from the 'terminal' to the host in advance of previously-transmitted data. Kill signals should reach their destination ahead of any other ongoing transmissions. Discuss the solution of this problem with connection-oriented and connectionless protocols.

3.9 Ans.

The problem is that a kill signal should reach the receiving process quickly even when there is buffer overflow (e.g. caused by an infinite loop in the sender) or other exceptional conditions at the receiving host.

With a connection-oriented, reliable protocol such as TCP, all packets must be received and acknowledged by the sender, in the order in which they are transmitted. Thus a kill signal cannot overtake other data already in the stream. To overcome this, an out-of-band signalling mechanism must be provided. In TCP this is called the URGENT mechanism. Packets containing data that is flagged as URGENT bypass the flow-control mechanisms at the receiver and are read immediately.

With connectionless protocols, the process at the sender simply recognizes the event and sends a message containing a kill signal in the next outgoing packet. The message must be resent until the receiving process acknowledges it.

3.10 What are the disadvantages of using network-level broadcasting to locate resources:

i) in a single Ethernet?

ii) in an intranet?

To what extent is Ethernet multicast an improvement on broadcasting?

3.10 Ans.

i. All broadcast messages in the Ethernet must be handled by the OS, or by a standard daemon process. The overheads of examining the message, parsing it and deciding whether it need be acted upon are incurred by every host on the network, whereas only a small number are likely locations for a given resource. Despite this,

note that the Internet ARP does rely on Ethernet braodcasting. The trick is that it doesn't do it very often - just once for each host to locate other hosts on the local net that it needs to communicate with.

ii. Broadcasting is hardly feasible in a large-scale network such as the Internet. It might just be possible in an intranet, but ought to be avoided for the reasons given above.

Ethernet multicast addresses are matched in the Ethernet controller. Multicast message are passed up to the OS only for addresses that match multicast groups the local host is subscribing to. If there are several such, the address can be used to discriminate between several daemon processes to choose one to handle each message.

3.11 Suggest a scheme that improves on MobileIP for providing access to a web server on a mobile device which is sometimes connected to the Internet by mobile phone and at other times has a wired connection to the Internet at one of several locations.

3.11 Ans.

The idea is to exploit the cellular phone system to locate the mobile device and to give the IP address of its current location to the client.

3.12 Show the sequence of changes to the routing tables in Figure 3.8 that would occur (according to the RIP algorithm given in Figure 3.9) after the link labelled 3 in Figure 3.7 is broken.

3.12 Ans.

Routing tables with changes shown in red (grey in monochrome printouts):

Step 1: costs for routes that use Link 3 have been set to ∞ at A, D

Routings from A

To	*Link*	*Cost*
A	local	0
B	1	1
C	1	2
D	3	∞
E	1	2

Routings from B

To	*Link*	*Cost*
A	1	1
B	local	0
C	2	1
D	1	2
E	4	1

Routings from C

To	*Link*	*Cost*
A	2	2
B	2	1
C	local	0
D	5	2
E	5	1

Routings from D

To	*Link*	*Cost*
A	3	∞
B	3	∞
C	6	2
D	local	0
E	6	1

Routings from E

To	*Link*	*Cost*
A	4	2
B	4	1
C	5	1
D	6	1
E	local	0

Step 2: after first exchange of routing tables

Routings from A

To	*Link*	*Cost*
A	local	0
B	1	1
C	1	2
D	3	∞
E	1	2

Routings from B

To	*Link*	*Cost*
A	1	1
B	local	0
C	2	1
D	1	∞
E	4	1

Routings from C

To	*Link*	*Cost*
A	2	2
B	2	1
C	local	0
D	5	2
E	5	1

Routings from D

To	*Link*	*Cost*
A	3	∞
B	3	∞
C	6	2
D	local	0
E	6	1

Routings from E

To	*Link*	*Cost*
A	4	2
B	4	1
C	5	1
D	6	1
E	local	0

Step 3: after second exchange of routing tables

Routings from A		
To	*Link*	*Cost*
A	local	0
B	1	1
C	1	2
D	3	∞
E	1	2

Routings from B		
To	*Link*	*Cost*
A	1	1
B	local	0
C	2	1
D	4	2
E	4	1

Routings from C		
To	*Link*	*Cost*
A	2	2
B	2	1
C	local	0
D	5	2
E	5	1

Routings from D		
To	*Link*	*Cost*
A	6	3
B	6	2
C	6	2
D	local	0
E	6	1

Routings from E		
To	*Link*	*Cost*
A	4	2
B	4	1
C	5	1
D	6	1
E	local	0

Step 4: after third exchange of routing tables.

Routings from A		
To	*Link*	*Cost*
A	local	0
B	1	1
C	1	2
D	1	3
E	1	2

Routings from B		
To	*Link*	*Cost*
A	1	1
B	local	0
C	2	1
D	4	2
E	4	1

Routings from C		
To	*Link*	*Cost*
A	2	2
B	2	1
C	local	0
D	5	2
E	5	1

Routings from D		
To	*Link*	*Cost*
A	6	3
B	6	2
C	6	2
D	local	0
E	6	1

Routings from E		
To	*Link*	*Cost*
A	4	2
B	4	1
C	5	1
D	6	1
E	local	0

3.13 Use the diagram in Figure 3.13 as a basis for an illustration showing the segmentation and encapsulation of an HTTP request to a server and the resulting reply. Assume that request is a short HTTP message, but the reply includes at least 2000 bytes of html.

3.13 Ans.

Left to the reader.

3.14 Consider the use of TCP in a Telnet remote terminal client. How should the keyboard input be buffered at the client? Investigate Nagle's and Clark's algorithms [Nagle 1984, Clark 1982] for flow control and compare them with the simple algorithm described on page 103 when TCP is used by (a) a web server, (b) a Telnet application, (c) a remote graphical application with continuous mouse input.

3.14 Ans.

The basic TCP buffering algorithm described on p. 105 is not very efficient for interactive input. Nagle's algorithm is designed to address this. It requires the sending machine to send any bytes found in the output buffer, then wait for an acknowledgement. Whenever an acknowledgement is received, any additional characters in the buffer are sent. The effects of this are:

a) For a web server: the server will normally write a whole page of HTML into the buffer in a single *write*. When the *write* is completed, Nagle's algorithm will send the data immediately, whereas the basic algorithm would wait 0.5 seconds. While the Nagle's algorithm is waiting for an acknowledgement, the server process can write additional data (e.g. image files) into the buffer. They will be sent as soon as the acknowledgement is received.

b) For a remote shell (Telnet) application: the application will write individual key strokes into the buffer (and in the normal case of full duplex terminal interaction they are echoed by the remote host to the Telnet client for display). With the basic algorithm, full duplex operation would result in a delay of 0.5 seconds before any of the characters typed are displayed on the screen. With Nagle's algorithm, the first character typed is sent immediately and the remote host echoes it with an acknowledgement piggy-backed in the same packet. The acknowledgement triggers the sending of any further characters that have been typed in the intervening period. So if the remote host responds sufficiently rapidly, the display of typed characters appears to be instantaneous. But note that a badly-written remote application that reads data from the TCP buffer one character at a time can still cause problems - each read will result in an acknowledgement indicating that one further character should be sent - resulting in the transmission of an entire IP frame for each character. Clarke [1982] called this the *silly window syndrome*. His solution is to defer the sending of acknowledgements until there is a substantial amount of free space available.

c) For a continuous mouse input (e.g. sending mouse positions to an X-Windows application running on a compute server): this is a difficult form of input to handle remotely. The problem is that the user should see a smooth feedbvack of the path traced by the mouse, with minimal lag. Neither the basic TCP algorithm nor Nagle's nor Clarke's algorithm achieves this very well. A version of the basic algorithm with a short timeout (0.1 seconds) is the best that can be done, and this is effective when the network is lightly loaded and has low end-to-end latency - conditions that can be guaranteed only on local networks with controlled loads.
See Tanenbaum [1996] pp. 534-5 for further discussion of this.

3.15 Construct a network diagram similar to Figure 3.10 for the local network at your institution or company.

3.15 Ans.

Left to the reader.

3.16 Describe how you would configure a firewall to protect the local network at your institution or company. What incoming and outgoing requests should it intercept?

3.16 Ans.

Left to the reader.

3.17 How does a newly-installed personal computer connected to an Ethernet discover the IP addresses of local servers? How does it translate them to Ethernet addresses?

3.17 Ans.

The first part of the question is a little misleading. Neither Ethernet nor the Internet support 'discovery' services as such. A newly-installed computer must be configured with the domain names of any servers that it needs to access. The only exception is the DNS. Services such as BootP and DHCP enable a newly-connected host to acquire its own IP address and to obtain the IP addresses of one ore more local DNS servers. To obtain the IP addresses of other servers (e.g. SMTP, NFS, etc.) it must use their domain names. In Unix, the *nslookup* command can be used to examine the database of domain names in the local DNS servers and a user can select approriate ones for use as servers.The domain names are translated to IP addresses by a simple DNS request.

The Address Resolution Protocol (ARP) provides the answer to the second part of the question. This is described on pages 95-6. Each network type must implement ARP in its own way. The Ethernet and related networks use the combination of broadcasting and caching of the results of previous queries described on page 96.

3.18 Can firewalls prevent denial of service attacks such as the one described on page 96? What other methods are available to deal with such attacks?

3.18 Ans.

Since a firewall is simply another computer system placed in front of some intranet services that require protection, it is unlikely to be able to prevent denial of service (DoS) attacks for two reasons:

- The attacking traffic is likely to closely resemble real service requests or responses.
- Even if they can be recognized as malicious (and they could be in the case described on p. 96), a successful attack is likely to produce malicious messages in such large quantities that the firewall itself is likely to be overwhelemed and become a bottleneck, preventing communication with the services that it protects.

Other methods to deal with DoS attacks: no comprehensive defence has yet been developed. Attacks of the type described on p. 96, which are dependent on IP spoofing (giving a false 'senders address') can be prevented at their source by checking the senders address on all outgoing IP packets. This assumes that all Internet sites are managed in such a manner as to ensure that this check is made - an unlikely circumstance. It is difficult to see how the targets of such attacks (which are usually heavily-used public services) can defend themselves with current network protocols and their security mechanisms. With the advent of quality-of-service mechanisms in IPv6, the situation should improve. It should be possible for a service to allocate only a limited amount of its total bandwidth to each range of IP addresses, and routers thorughout the Internet could be setup to enforce these resource allocations. However, this approach has not yet been fully worked out.
