1.
2.11
Consider a simple server that carries out client requests without accessing other servers. Explain why it is generally not possible to set a limit on the time taken by such a server to respond to a client request. What would need to be done to make the server able to execute requests within a bounded time? Is this a practical option? page 78

In order to make the server able to execute requests within a bounded time, the network should be stable, the number of requests in a given time should be limited, or the bandwidth of the network should be enough for all the requests, and there should be other methods to prevent system problems happening to the server.
It is not a practical option to set a bounded time of response. Because the time to process a request is dependent on multiple aspects, and it is hard to promise a guarantee of a time that the task will be done. For example, if the situation of the network is bad, the response time would be long; if there are too many tasks waiting in the message queue, the process time would be long; if there are multiple threads processing one task, and there are sufficient resources, the process time could be very short.

2.14
Consider two communication services for use in asynchronous distributed systems. In service A, messages may be lost, duplicated or delayed and checksums apply only to headers. In service B, messages may be lost, delayed or delivered too fast for the recipient to handle them, but those that are delivered arrive with the correct contents. Describe the classes of failure exhibited by each service. Classify their failures according to their effects on the properties of validity and integrity. Can service B be described as a reliable communication service? page 83, page 87

Failures could happen to A:
Omission Failure: the process could crash if the messages are lost or delayed. Because when the messages are lost, other systems in the distributed system can detect the timeout of system A, and treat it as a crash. And it breaks the property of validity.
Arbitrary Failure: if the messages in A are duplicated, A might return a wrong value in response. And if the checksums only applies to headers, the body of the messages might be corrupted. And it breaks the property of integrity.

Failures could happen to B:
Omission Failure: the process could crash if the messages are lost or delayed. Because when the messages are lost, other systems in the distributed system can detect the timeout of system B, and treat it as a crash. And it breaks the property of validity.
Service B does not break the property of integrity, but since it breaks the property of validity, it cannot be called reliable.

2.
What is middleware for Dist Systems/ What are its uses (list and explain).
Consider a multiplayer game as a Dist Sys. It is played on 4 players PCs and a Game Server.

What are the heterogeneous elements among these 5 systems at each layer? Give examples to illustrate this (e. g. MacOs on PC1; Linux on PC 2 etc.) and then explain how middleware helps here.

Middleware is software that runs between operating system and application, in order to solve the problem of heterogeneity inherent and reduce the complexity of distributed systems.
Uses:
communication:
mask heterogeneity of networks and hardware: solve the problem of protocols of networks incompatible with protocols of hardware.
mask heterogeneity of operating systems or programming languages, or both: when systems are in different operating systems, or softwares is coded in different programming languages, or a programming language cannot run in an operating system, middleware can solve the heterogeneity.
systems in distributed systems are in different locations, so they are in different clocks, middleware can help communicate.
processing and storage:
mask heterogeneity among vendor implementations of the same middleware standard
provide transparency in location, concurrency, replication, failures and mobility: there are some middlewares created to store the messages temporarily, solve the problems in concurrency, help with the replication process, handle and report the failures, Remote procedure call middleware can invoke a procedure across a network.

heterogeneous elements among these 5 systems at each layer:
In the network layer, some computers might not support the current network format, for example, some computers might not support wireless networks. In this case, we can use middleware to provide support and configuration for wireless networks.

In the operating system layer, the 5 systems might be in different operating systems like Linux, MacOS and Windows, but the service may not support some operating systems, so we can use a middleware to solve the heterogeneity of operating systems.

In the application layer, some services could be coded into different coding languages like java, python or C, but some systems may not support these languages, or not support the corresponding version of languages. So we can use a middleware to solve the heterogeneity of coding languages.

3.
3.1 A client sends a 200 byte request message to a service, which produces a response containing 5000 bytes. Estimate the total time required 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) when the server process is in the same machine as the client.
[Latency per packet (local or remote,
incurred on both send and receive): 5 ms
Connection setup time (TCP only): 5 ms
Data transfer rate: 10 Mbps

MTU: 1000 bytes
Server request processing time: 2 ms
Assume that the network is lightly loaded.]
pages 98, 138

Message transmission time = latency + length／data transfer rate
i) 5 + 2000/10000 + 2 + 5000/1000(5 + 10000/10000) = 37.2
ii) 5 + 2000/10000 + 2 + 5000/1000(5 + 10000/10000) + 5 = 42.2
iii) Assume the data transfer rate is 500 Mbps when the server and client are in the same machine. Then the time to complete the request is:
5 + 2000/500000 + 5 + 50000/500000 = 10.104

3.7.ii
Compare connectionless (UDP) and connection-oriented (TCP) communication for the implementation of each of the following application-level or presentation-level protocols:
ii) file transfer (for example, FTP);
If we use UDP to transfer files, the transfer speed can be fast, and it is suitable for big file transfer. However, we cannot guarantee the quality of transferring. The client might receive the corrupted file from the server using UDP.
If we use TCP to transfer files, the transfer speed is slower compared to UDP, but it is more reliable. TCP can be used to transfer large files, and we can guarantee the correctness and integrity of the file.

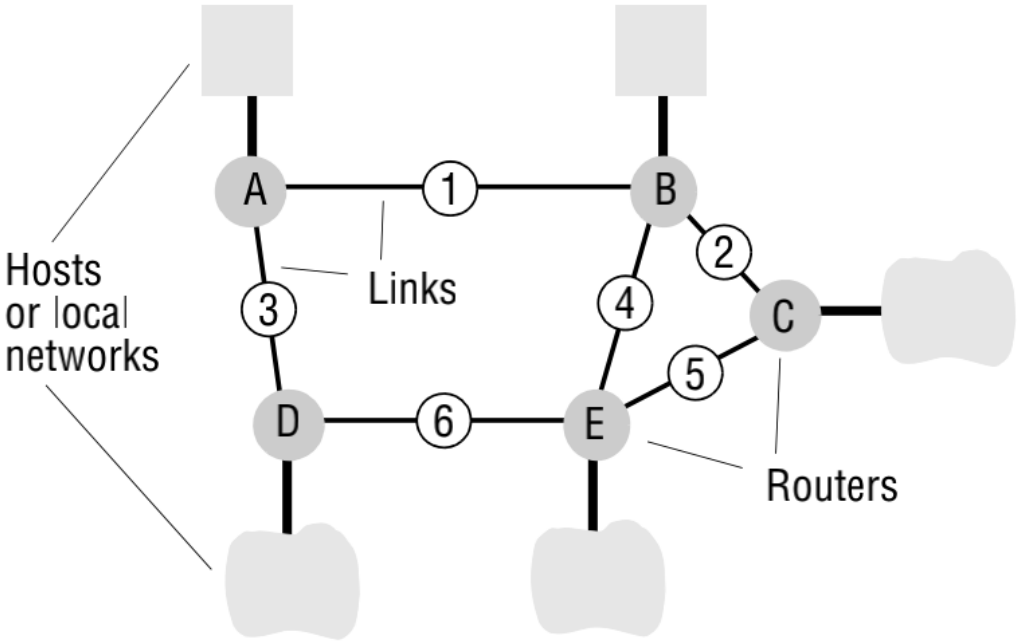
3.8
Explain how TCP ensures the reliable delivery of long sequences of bytes via streambased programming abstraction. Why can’t this happen in a UDP? pages 138, 139

TCP is connection-oriented, there are processes for the sender and receiver to acknowledge each other, and transfer data only after the connection channel is created.

Sequencing: TCP protocol attaches a sequence number to each segment, and segment can be placed in the input stream only after all lower-numbered segments have been placed in the stream, therefore, it guarantees the integrity of the message.

Flow Control: TCP records and sent the received sequence number, so it can monitor how many messages are sent before the next segment get acknowledged, and when the traffic is too busy, exceeding the size of the maximum window, it can control the flow.

3.12
Show the sequence of changes to the routing tables in Figure 3.8 that will occur (according to the RIP algorithm given in Figure 3.9) after the link labeled 3 in Figure 3.7 is broken. pages 114–117



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

After the link labeled 3 is broken, the cost routing through link 3 is infinite, so
cost(AD) = ∞, cost(DA) = ∞, cost(DB) = ∞
Since the current route from B to D goes through link 1 and link 3, and link 3 is broken,
cost(DA) = ∞
After the exchange of routing tables,
The route from D to B goes through link 6 and link 4,
so Link(DB) = 6, cost(DB) = cost(DE) + cost(EB) = 2
The route from D to A goes through link 6, link 4 and link 1,
so Link(DA) = 6, cost(DA) = cost(DB) + cost(BA) = 3
The route from B to D goes through link 4 and link 6,

so Link(BD) = 4, cost(BD) = cost(BE) + cost(ED) = 2
The route from A to D goes through link 1, link 4 and link 6,
so Link(AD) = 1, cost(AD) = cost(AB) + cost(BD) = 3
cardinality: the number of occurrences of an entity