CHAPTER 19

PRACTICE SET

Questions

Q19-1. The minimum length of the IPv4 header is 20 bytes and the maximum is 60 bytes. The value of the header length field defines the header length in multiples of four bytes, which means that HLEN can be between 5 and 15. It cannot be less than 5 and it cannot be greater than 15. It is exactly 5 when there is no option.

Q19-3. Since the fragmentation offset field shows the offset from the beginning of the original datagram in multiples of 8 bytes, an offset of 100 indicates that the first byte in this fragment is numbered 800, which means bytes numbered 0 to 799 (for a total of 800 bytes) were sent before.

Q19-5. The header length is $6 \times 4 = 24$. The option length is then $24 - 20 = 4$ bytes.

Q19-7. The protocol field and the port numbers both have the same functionality: multiplexing and demultiplexing. Port numbers are used to do these tasks at the transport layer; the protocol field is used to do the same at the network layer. We need only one protocol field at the network layer because payload taken from a protocol at the source should be delivered to the same protocol at the destination. The client and server processes, on the other hand, normally have different port numbers (ephemeral and well-known), which means we need two port numbers to define the processes. The size of the protocol field defines the total number of different protocols that use the service of the network layer, which is a small number (eight bits is enough for this purpose). On the other hand, many new applications may by added every day that needs a larger size of the port number field (sixteen bits is assigned).

Q19-9. Each datagram should have a unique identification number that distinguishes it from other datagrams sent by the same source. The identification number is copied into all fragments. In other words, the identification number glues all fragments belonging to the same datagram together.

Q19-11. If this happens, we may enter a loop, a vicious circle. The first datagram is in error; the second datagram reports error in the first. If the second datagram is also in error, the third datagram will be carrying error information about the second, and so.

Q19-13. These two messages need new fields and information that is not supported by the ICMP protocol. A new application-layer client and server are needed to send requests and receive responses. The designer of Mobile IP has decided to implement it at the application layer instead of at the network layer, which requires changes in this layer. UDP was selected instead of TCP as the transport layer because of its lower overhead and because the messages exchanged have clear boundaries and fixed size.

Q19-15. An ICMP solicitation message has all the necessary fields to be used as an agent solicitation. No extra fields are needed.

Problems

P19-1. The total length of the datagram is $(00A0)_{16} = 160$ bytes. The header length is $5 \times 4 = 20$. The size of the payload is then $160 - 20 = 140$. The efficiency = 140 / 160 = 87.5%.

P19-3. Using hexadecimal notations, we have

a. The wrapped sum can be found by adding the quotient and remainder when dividing the sum by the modulus, which is 2^{16} or $(10000)_{16}$ in this case. Note that the modulus is actually $(FFFF + 1)_{16}$. Since the wrapped sum has less than 8 digits (we can use the following steps); otherwise; we need a loop to continuously find the quotient and remainder. Note that we use the symbol (/) to define quotient and (%) to define remainder. All calculations are in hexadecimal.

(Wrapped sum) = quotient + remainder

(Wrapped sum) = (sum) / (10000) + (sum) % (10000)

(Wrapped sum) = (1344E) / (10000) + (1344E) % (10000)

(Wrapped sum) = (1) + (344E) = **344F**

b. The checksum can be calculated from the wrapped sum easily. All calculations are in hexadecimal.

(Checksum) = (modulus − 1) − (Wrapped Sum)

(Checksum) = (FFFF) − (344F) = **CBB0**

P19-5. We can calculate the sum, wrapped sum and checksum after each word if we keep track of the sum in each step. The following shows the process. The value of the last row in the last column shows the final checksum. All calculations are in decimal.

word	sum	wrapped sum	checksum
17664	17664	17664	47871
28	17692	17692	47843
49153	66845	1310	64225
0	66845	1310	64225
1041	67886	2351	63184
0	67886	2351	63184
2572	70458	4923	60612
3589	74047	8512	57023
3078	77125	11590	53945
1801	78926	13391	**52144**

P19-7. In each case, we first need to think about the value of *M* and then the value of the offset:

a. Since *M* = 1, it means there are more fragments and this is the first or middle; since the offset field is zero, it means this is the first fragment.

b. Since *M* = 1, it means there are more fragments and this the first or middle; since the offset field is nonzero, it means this is a middle fragment.

P19-9. We show the value of each word in hexadecimal and then add them to get the sum. Note the value of the sum in this case is less than FFFF + 1, so we do not need to wrap the sum using quotient and remainder. We subtract the sum from the FFFF to get the checksum.

Words	Hex values
8 & 0	0800
0	0000
1	0001
9	0009
T & E	5445
S & T	5354
Sum	**AFA3**
Checksum	**505C**

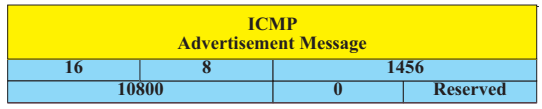
P19-11. Let us discuss each case separately:

a. Packet sniffing can be defeated if the datagram is encrypted at the source and decrypted at the destination using an unbreakable scheme.

b. Packet modification can be defeated using a strong message integrity scheme.

c. IP spoofing can be defeated using a strong entity authentication scheme.

P19-13. See the following figure:



P19-15. See the following figure: