CHAPTER 4

PRACTICE SET

Questions

Q4-1. The number of virtual circuits is $2^8 = 256$.

Q4-2. The three phases are setup phase, data transfer, and teardown phase.

Q4-3. The transport layer communication is between two ports; the network layer communication is between two hosts. This means that each layer has a different source/destination address pair; each layer needs a different header to accommodate these pair of addresses. In addition, there are other pieces of information that need to be separately added to the corresponding header.

Q4-4. Routing cannot be done at the transport layer, because the communication at the transport layer is one single logical path between the source port and the destination port. Routing cannot be done at the data-link layer because the communication at the data-link layer is between two nodes (one single path); there is no need for routing. On the other hand, there are several possible paths for a packet between the source host and destination host at the network layer. Routing is the job of selecting one of these paths for the packet.

Q4-5. Forwarding is delivery to the next node. A router uses its forwarding table to send a packet out of one of its interfaces and to make it to reach to the next node. In other words, forwarding is the decision a router makes to send a packet out of one of its interfaces. Routing, on the other hand, is an end-to-end delivery resulting in a path from the source to the destination for each packet. This means a routing process is a series of forwarding processes. To enable each router to perform its forwarding duty, routing protocols need to be running all of the time to provide updated information for forwarding tables. Although forwarding is something we can see in the foreground, in the background, routing provides help to the routers to do forwarding.

Q4-6.

a. In the datagram approach, the forwarding decision is made based on the destination address in the packet header.

b. In the virtual-circuit approach, the forwarding decision is based on the label in the packet header.

Q4-7. The throughput is the smallest transmission rate, or 140 Kbps. The bottleneck is now the link between the source host and R1.

Q4-8. 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.

Q4-9. The identification numbers need to be contiguous. The identification number of the last datagram should be 1024 + 100 − 1 = 1123.

Q4-10. 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.

Q4-11. None of these services are implemented for the IP protocol in order to make it simple.

Q4-12. Four types of delays are transmission delay, propagation delay, processing delay, and queuing delay.

Q4-13. Many blocks can have the same prefix length. The prefix length only determines the number of addresses in the block, not the block itself. Two blocks can have the same prefix length but start in two different points in the address space. For example, the following two blocks have the same prefix length, but they are definitely two different blocks. The length of the blocks is the same, but the blocks are different.

127.15.12.32/27 **174.18.19.64/27**

Q4-14. We cannot find the prefix length because we don’t know the length of the block. The second given address can be an address in the middle. We need the last address, or the length of the block to find the prefix length.

Q4-15. If the first and the last addresses are known, the block is fully defined. We can first find the number of addresses in the block (*N*) and then find the prefix length (*n*).

$N = (\text{last address}) - (\text{first address}) + 1$

$n = 32 - \log_2 N$

Block: (first address)/n

Q4-16. If the first and the number of addresses (*N*) are known, the block is fully defined. We can find the prefix length (*n*) using the number of addresses.

$n = 32 - \log_2 N$

Block: (first address)/n

Q4-17. 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).

Q4-18. Two fields, source IP address and the identification, are needed to uniquely define fragments belonging to the same datagram. The value of the identification field is not enough because two sources may start with the same identification number.

Q4-19. 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.

Q4-20. MPLS adds an extra header to an IP datagram. This means that MPLS implicitly creates a new layer in which a datagram is encapsulated. This layer is between the network layer and the data-link layer.

Q4-21. There is no need for a router and subnetting. Each customer can be directly connected to the ISP server. In this case, the set of addresses assigned to customers can be thought of as belonging to a single block with the prefix length *n* (the prefix length assigned to the ISP).

Q4-22. The three auxiliary protocols are ICMP, IGMP, and ARP.

Q4-23. The header length is $6 \times 4 = 24$. The option length is then $24 - 20 = 4$ bytes.

Q4-24. It can be 23 or 1. It cannot be 0 because it means the packet cannot travel at all. It cannot be 301, because the length of the value field is 8 bits, which means the maximum value is 255.

Q4-25. According to the principle we mentioned in the text, the shortest path is the inverse of the original one. The shortest path is $G \rightarrow E \rightarrow B \rightarrow A$.

Q4-26. According to the principle we mention in the text, the shortest path from A to N can be found in two steps. We first use the shortest path from A to H to move to node H. We then use the shortest path from node H to N. The result is $A \rightarrow B \rightarrow H \rightarrow G \rightarrow N$.

Q4-27. 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.

Q4-28. The source IP address is the IP address of the router interface from which the original IP datagram is received. The destination IP address is the IP address of the original source host that sent the original datagram. In other words, the reporting router in this case acts as a source host. This proves that a router needs an IP address for each of its interfaces.

Q4-29. The source and destination IP addresses in datagrams carrying payloads between the hosts are the IP addresses of the hosts; the IP addresses carrying routing update packets between routers are IP addresses of the routing interfaces from which the packets are sent or received. This shows that a router needs as many IP addresses as it has interfaces.

Q4-30. Each datagram has a different source IP address: the IP address of the interface from which it is sent out (a router can have only one immediate neighbor on each interface). Each datagram also has a different destination IP address: the IP address of the router interface at which it arrives.

Q4-31. Although RIP is running as a process using the service of the UDP, the process is called a *daemon* because it is running all the time in the background. Each router acts both as a client and a server; it acts as a client when there is a message to send; it acts as a server when a message arrives.

Q4-32. RIP messages are short with clear message boundaries. It is not efficient to use the service of TCP with all of the connection establishment and connection teardown overhead.

Q4-33. Link-state routing uses Dijkstra’s algorithm to first create the shortest-path tree before creating the forwarding table. The algorithm needs to have the complete LSDB to start.

Q4-34. The path-vector routing algorithm is actually distance-vector routing using the best path instead of the shortest distance as the metric. Each node first creates a forwarding table, assuming it can only reach immediate neighbors. The forwarding table is gradually improved as path vectors arrive from the immediate neighbors.

Q4-35. The three ASs described in the text are *stub*, *multihomed*, and *transient.* The first two do not allow transient traffic; the third does. The stub and multihomed ASs are similar in that they are either the sink or source of traffic; the first is connected to only one other AS, but the second is connected to more than one ASs.

Q4-36. We can say that a number of hops in RIP is the number of networks a packet travels to reach its final destination. The first network, in which the original host is located, is normally not counted in this calculation because the source host does not take part in routing. To reduce the traffic of exchanging routing updates, the hosts in the Internet do not take part in this process. This is done because the number of hosts in the Internet is much larger than the number of routers. Including hosts in this process makes the routing-update traffic unbearable.

Q4-37. In RIP, each router just needs to share its distance vector with its neighbor. Since each router has one type of distance vector, we need only one update message. In OSPF, each router needs to share the state of its links with every other router. Since a router can have several types of links (a router link, a network link, …), we need several update messages.

Q4-38. We need to have OSPF processes that run all the time because we never know when an OSPF message will arrive. These processes are running at the network layer, not at the application layer. They are normally referred to as daemons.

Q4-39. The type of payload can be determined from the value of the protocol field. The protocol field value for ICMP is 01; for OSPF, it is 89.

Q4-40.

a. router link **b.** router link **c.** network link

Q4-41. OSPF divides an AS into areas, in which routing in each area is independent from the others; the areas only exchange a summary of routing information between them. RIP, on the other hand, considers the whole AS as one single entity.

Q4-42. If the AS is small, it is normally recommended to consider it as only one area (the backbone area) to reduce the overhead of information exchange between areas.

Q4-43. BGP is designed to create semi-permanent communication between two BGP speakers; this requires the service of TCP. A connection is made between the two speakers and remains open, while the messages are exchanged between them. UDP cannot provide such a service.

Q4-44. The intradomain routing routes the packet inside an autonomous system that is totally in the control of the organization. On the other hand, the interdomain routing routes the packet through an autonomous system that is out of the control of the organization; the organization needs to apply a policy to decide through which AS the packet should pass.

Q4-45. The following shows the use of each attribute:

a. The LOCAL-PREF is used to implement the organization policy.

b. The AS-PATH defines the list of autonomous systems through which the destination can be reached.

c. The NEXT-HOP defines the next router to which the data packet should be forwarded.

Q4-46. In multicasting, the sender host sends only one copy of the message, but it is multiplied at the routers if needed; all multiplied copies have the same destination address. In multiple-unicasting, the sender host sends one copy for each destination; each copy has its own destination address.

Q4-47. It cannot. A link needs to be advertised in a router link LSP; a network needs to be advertised in a network link LSP.

Q4-48. Each AS is independent, which means that it can run one of the two common intradomain routing protocols (RIP or OSPF). On the other hand, the whole Internet is considered as one entity, which means that we must run only one interdomain routing protocol (the common one is BGP).

Q4-49. In each case, we find the corresponding block to be able to find the group

a. 224.0.1.7 belongs to the block 224.0.1.0/24; it belongs to the internetwork control block.

b. 232.7.14.8 belongs the block 232.0.0.0/8; it belongs to the SSM block.

c. 239.14.10.12 belongs the block 239.0.0.0/8; it belongs to the administratively scoped block.

Q4-50. If a host is a member of *N* multicast group, it will have *N* multicast addresses.

Q4-51. Sending a multiple-recipient e-mail is a case of multiple unicasting. An email message needs recipient addresses at the application layer, which cannot be translated to a multicast address at the network layer. The recipients of an email address do not necessarily belong to the same group. In other words, we a one-to-many communication at the application layer, which should not be confused to one-to-many communication at the network layer.

Q4-52. The multicast address block is 224.0.0.0/4. In other words, a multicast address is between 224.0.0.0 and 239.255.255.255. Based on this criteria we have

a. A multicast   b. A multicast   c. Not a multicast

Q4-53.

a. In the source-based tree approach, we need 20 × 4 = 80 shortest-path trees.

b. In the group-shared tree, we need only 4 shortest-path trees, one for each group.

Q4-54. DVMRP allows a router to create the shortest path-tree whenever it receives a multicast packet (on demand). The number of shortest-path trees in DVMRP that use the source-based approach is huge. This means if each router created all of the required multicast shortest-path trees, it would be a huge overhead.

Q4-55. Each router using DVMRP creates the shortest-path three in three steps:

- **a.** In the first step, the router uses the RPF algorithm to keep only packets that have arrived from the source using the shortest-path three. In other words, the first part of the tree is made using the RPF algorithm.
- **b.** In the second step, the router uses the RPB algorithm to create a broadcast tree.
- **c.** In the third step, the router use the RPM algorithm to change the broadcast tree created in the second step to a multicast tree.

Q4-56. MOSPF uses Dijkstra's algorithm to create the whole broadcast path tree in one shot, but DVMRP needs to use three steps because it does not have the LSDB to use Dijkstra's algorithm.

Q4-57. Every multicast routing algorithm needs to somehow use a unicast protocol in its operation. For example, DVMRP needs to use RIP and MOSPF needs to use OSPF. Although PIM also needs to use a unicast protocol, the protocol can be either RIP or OSPF.

Q4-58. PIM-DM is very similar to DVMRP, but it does not care about controlling the broadcast step of DVMRP because it assumes that most networks have a loyal member in each group. It only uses the first step (RPF) and the third step (RPM).

Q4-59. In PIM-DM, it is assumed that most networks have a loyal member in each group, so it does not matter if the first packet reaches all networks. In PIM-SM, it is assumed that a few networks has a loyal member in each group, so broadcasting is wasting the bandwidth.

Q4-60. We can say (1) larger address space, (2) better header format, (3) new options, (4) allowance for extension, (5) support for resource allocation, and (6) support for more security.

Q4-61. The group list is the union of the individual lists; it is {G1, G2, G3, G4}.

Q4-62.

- **a.** In unicast communication, the destination is only one of the leaves of the tree in each transmission.
- **b.** In multicast communication, the destination may be one or more leaves of the tree in each transmission.

Q4-63. The three protocols IGMP, ICMP, and ARP in IPv4 have been combined into a single protocol, ICMPv6.

Q4-64. The IP header is included because it contains the IP address of the original source. The first 8 bytes of the data are included because they contain the first section of the TCP or UDP header which contains information about the port numbers (TCP and UDP) and sequence number (TCP). This information allows the source to direct the ICMP message to the correct application.

Q4-65. The flow field can be used in several ways. It allows IPv6 to be used as a connection-oriented protocol. It also allows IPv6 to give priority to different payloads, such as giving high priority to real-time multimedia applications.

Q4-66. A compatible address is an address of 96 bits of 0s followed by 32 bits of an IPv4 address. A mapped address is an address of 80 bits of 0s followed by 16 bits of 1s and followed by 32 bits of an IPv4 address. A compatible address is used when a computer using IPv6 wants to send a packet to another computer using IPv6. A mapped address is used when a computer using IPv6 wants to send a packet to a computer still using IPv4.

Problems

P4-1. The following fields can be changed from one router to another:

a. HLEN: If there is option change

b. Total length: If fragmented or options change

c. Flags: If fragmented

d. Fragmentation Offset: If fragmented

e. Time-to-Live; Decremented at each router

f. Header Checksum: Need to change because of other changes

P4-2. 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.

P4-3. 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.

P4-4. The size of the address in each case is the base to the power of the number of digits:

a. The size of the address space is $2^{16} = 65{,}536$.

b. The size of the address space is $16^6 = 16{,}777{,}216$.

c. The size of the address space is $8^4 = 4096$.

P4-5. 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%.

P4-6. We analyze each byte or group of bytes to answer the questions:

a. The second hex digit in the first byte is 5 (HLEN), which means that the header length is only $5 \times 4 = 20$ bytes.

b. There are no options because the header size is only 20 bytes.

c. The total length of the packet is $(0054)_{16}$ or 84 bytes. Since the header is 20 bytes, it means the packet is carrying 64 bytes of data.

d. Since the flags field fragmentation offset bit is all 0s, the packet is not fragmented.

e. The value of the TTL field is $(20)_{16}$ or 32 in decimal, which means the packet may visit up to 32 more routers.

f. The value of the protocol field is 6, which means that the packet is carrying a segment from the TCP protocol.

P4-7. The class can be defined by looking at the first byte (see figure 4.31):

a. Since the first byte is between 128 and 191, the class is B.

b. Since the first byte is between 192 and 223, the class is C.

c. Since the first byte is between 240 and 255, the class is E.

P4-8. The class can be defined by checking the first few bits (see figure 4.31). We need to stop checking if we find a 0 bit or four bits have already been checked.

a. Since the first bit is 0, the Class is A.

b. Since the first four bits are 1110, the class is D.

c. Since the first three bits are 110, the class is C.

P4-9. We change each byte to the corresponding binary representation:

a. `01101110 00001011 00000101 01011000`

b. `00001100 01001010 00010000 00010010`

c. `11001001 00011000 00101100 00100000`

P4-10. We change each 8-bit section to the corresponding decimal value and insert dots between the bytes.

a. 94.176.117.21 **b.** 137.142.208.49 **c.** 87.132.55.15

P4-11. We first write the mask in binary notation and then count the number of leftmost 1s.

a. 11111111 11100000 00000000 00000000 → ***n*: 11**

b. 11111111 11110000 00000000 00000000 → ***n*: 12**

c. 11111111 11111111 11111111 10000000 → ***n*: 25**

P4-12. We first write each potential mask in binary notation and then check if it has a contiguous number of 1s from the left followed by 0s.

a. 11111111 11100001 00000000 00000000 → **Not a mask**

b. 11111111 11000000 00000000 00000000 → **A mask**

c. 11111111 11111111 11111111 00000110 → **Not a mask**

P4-13. We can write the address in binary. Set the last 32 – *n* bits to 0s to get the first address; set the last 32 – *n* bits to 1s to get the last address. You can use one of the applets at the book website to check the result.

a.

Given:	00001110 00001100 01001000 00001000	14.12.72.8/24
First:	00001110 00001100 01001000 00000000	14.12.72.0/24
Last:	00001110 00001100 01001000 11111111	14.12.72.255/24

b.

Given:	11001000 01101011 00010000 00010001	200.107.16.17/18
First:	11001000 01101011 00000000 00000000	200.107.0.0/18
Last:	11001000 01101011 00111111 11111111	200.107.63.255/18

c.

Given:	01000110 01101110 00010011 00010001	70.110.19.17/16
First:	01000110 01101110 00000000 00000000	70.110.0.0/16
Last:	01000110 01101110 11111111 11111111	70.110.255.255/16

P4-14. We write the address in binary and then keep only the leftmost *n* bits.

a. 10101010 00101000 00001011

b. 01101110 00101000 111100

c. 01000110 00001110 00

P4-15. The whole block can be represented as 0.0.0.0/0. The first address in the class is 0.0.0.0. The prefix is 0 because no bits define the block; all bits define the address itself. Another test to prove that the prefix is 0 is that the number of addresses in the block can be found as 2^{32-n}. The value of n should be zero to make the number of addresses $N = 2^{32}$.

P4-16. The size of the block can be found as $N = 2^{32-n}$:

a. $2^{32-0} = 4{,}294{,}967{,}296$ b. $2^{32-14} = 262{,}144$ c. $2^{32-32} = 1$

P4-17. The prefix can be found as $n = 32 - \log_2 N$:

a. $n = 32 - \log_2 1 = 32$ b. $n = 32 - \log_2 1024 = 22$ c. $n = 32 - \log_2 2^{32} = 0$

P4-18. We can first write the prefix in binary and then change each 8-bit chunk to decimal:

a. `00000000 00000000 00000000 00000000` → mask: 0.0.0.0

b. `11111111 11111100 00000000 00000000` → mask: 255.252.0.0

c. `11111111 11111111 11111111 11111100` → mask: 255.252.255.252

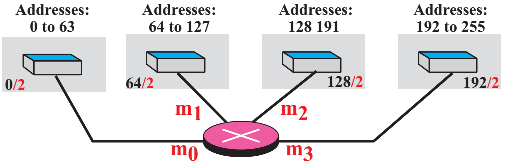
P4-19. The total number of addresses is $2^8 = 256$. This means we have 64 addresses for each network. We can divide the whole address space into four blocks (blocks 0 to 3), each of 64 addresses. The addresses in each block are allocated as (0 to 63), (64 to 127), (128 to 191), and (192 to 255). It can be checked that each block is allocated according to the two restrictions needed for the proper operation of CIDR. First, the number of addresses in each block is a power of 2. Second, the first address in each block is divisible by the number of addresses in the block, as shown below:

Block 0: $0/64 = 0$ Block 1: $64/64 = 1$ Block 2: $128/64 = 2$ Block 3: $192/64 = 3$

The prefix length for each group is $n_i = 8 - \log_2 64 = 2$. We can then write the ranges in binary to find the prefix for each block.

Block	Range	Range in binary	n	Prefix
0	0 to 63	00000000 to 00111111	2	00
1	64 to 127	01000000 to 01111111	2	01
2	128 to 191	10000000 to 10111111	2	10
3	192 to 255	11000000 to 11111111	2	11

The following shows the outline and the forwarding table. Note that each interface can use one of the addresses in the corresponding block.



Forwarding table

Prefix	Interface
00	m_0
01	m_1
10	m_2
11	m_3

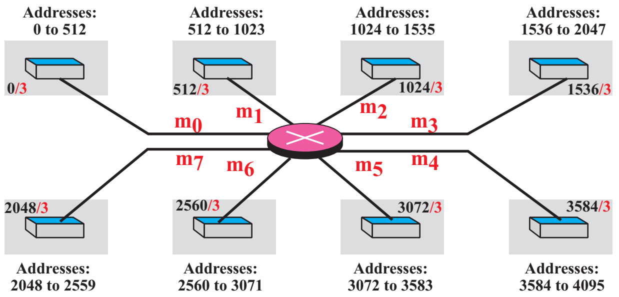
P4-20. The total number of addresses is 2^{12} = 4096. This means that there are 512 addresses for each network. We can divide the whole address space into eight blocks (blocks 0 to 7), each of 512 addresses. The addresses in each block are allocated as (0 to 511), (512 to 1023), (1024 to 1535), (1536 to 2047), …, (3584 to 4095). It can be checked that each block is allocated according to the two restrictions needed for the proper operation of CIDR. First, the number of addresses in each block is a power of 2. Second, the first address is divisible by the number of addresses as shown below:

Block 0: 0 / 512 = 0   **Block 1**: 512 / 512 = 1   **Block 2**: 1024 / 512 = 2   …

The prefix length for each group is $n_i = 12 - \log_2 512 = 3$. We can then write the ranges in binary to find the prefix for each block.

Block	Range	Range in binary	*n*	Prefix
0	0 to 511	000000000000 to 000111111111	3	000
1	512 to 1023	001000000000 to 001111111111	3	001
2	1024 to 1535	010000000000 to 010111111111	3	010
3	1536 to 2047	011000000000 to 011111111111	3	011
4	2048 to 2559	100000000000 to 100111111111	3	100
5	2560 to 3071	101000000000 to 101111111111	3	101
6	3072 to 3583	110000000000 to 110111111111	3	110
7	3584 to 4095	111000000000 to 111110000000	3	111

The following figure shows the outline and the forwarding table. Note that each interface can use one of the addresses in the corresponding block. The addresses are written in decimal (not dotted-decimal) because of the address space size.



Forwarding table

Prefix	Interface
000	m_0
001	m_1
010	m_2
011	m_3
000	m_4
000	m_5
000	m_6
000	m_7

P4-21. The total number of addresses is 2^9 = 512. We need, however, to check whether address allocation is done according to the restrictions for CIDR's proper operation. The address allocations to the networks are (N_0: 0 to 63),

(N_1: 64 to 255), and (N_2: 256 to 511). Each range is a power of 2, which means that the first restriction is fulfilled. The second restriction (the first address in the block should divide the number of addresses in the block) is fulfilled for N_0 and N_2, but not for N_1:

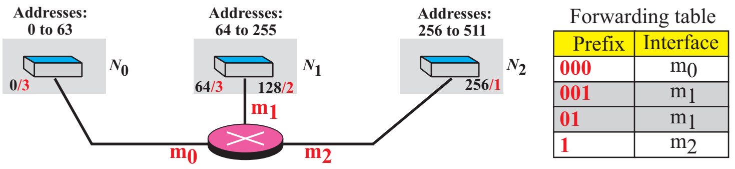
N_0: 0 / 64 = 0 N_1: 64 / 256 = 0.25 N_2: 256 / 256 = 1

One solution would be to think of the addresses in N_1 as the aggregation of two contiguous blocks (64 to 127) and (128 to 256) connected to the same interface. We call these blocks 1-1 and 1-2. The prefixes for blocks are

$$n_0 = 9 - \log_2 64 = 3 \qquad n_{1\text{-}1} = 9 - \log_2 64 = 3$$
$$n_{1\text{-}2} = 9 - \log_2 128 = 2 \qquad n_2 = 9 - \log_2 256 = 1$$

Block	Range	Range in binary	*n*	Prefix
0	0 to 63	000000000 → 000111111	3	000
1-1	64 to 127	001000000 → 001111111	3	001
1-2	128 to 255	010000000 → 011111111	2	01
2	256 to 511	100000000 → 111111111	1	1

Based on the above table, we can show the outline of the internet and addresses and the forwarding table, as shown below. Note that the address aggregation in $N1$ is transparent to the user as long as the router forwards the packet according to its forwarding table. If we need to be fair, we should say that $N1$ actually has two network addresses because it is made of two blocks. The administration can easily divide the block into two subblocks with a router.



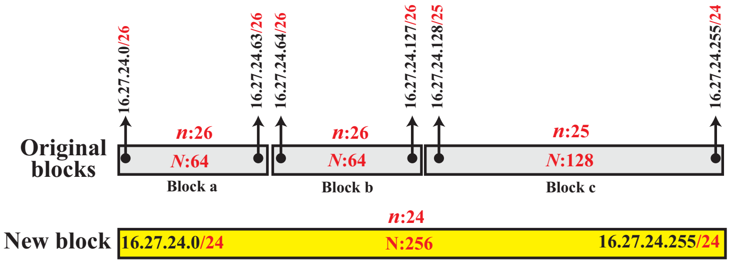
Forwarding table

Prefix	Interface
000	m_0
001	m_1
01	m_1
1	m_2

P4-22. One way to do this is to first find the size of each block. We can then add the size to the first address in the block to find the last address. Next, we can put the blocks together to find whether they can be combined into a larger block.

Block	Size	First address		Last address
a	$N = 2^{32-26} = 64$	16.27.24.0/26	→	16.27.24.63/26
b	$N = 2^{32-26} = 64$	16.27.24.64/26	→	16.27.24.127/26
c	$N = 2^{32-25} = 128$	16.27.24.128/25	→	16.27.24.255/26

Since the blocks are contiguous, we can combine the three blocks into a larger one. The new block has 256 addresses and $n = 32 - \log_2 256 = 24$.



P4-23. The organization is granted $2^{32-21} = 2^{11} = 2048$ addresses. The medium-size company has $2^{32-22} = 2^{10} = 1024$ addresses. Each small organization has $2^{32-23} = 2^9 = 512$ addresses. We can plot the range of addresses for each organization as shown below:

Large organization:	**12.44.184.0/21**	**to**	**12.44.191.255/21**
Medium organization:	**12.44.184.0/22**	**to**	**12.44.187.255/22**
Small organization 1:	**12.44.188.0/23**	**to**	**12.44.189.255/23**
Small organization 2:	**12.44.190.0/23**	**to**	**12.44.191.255/23**

The company install a router whose forwarding table is based on the longest-prefix match first principle as shown below.

Network address /mask	Next hop	Interface
00001100 00101100 1011110	…	Small organization 1
00001100 00101100 1011111	…	Small organization 2
00001100 00101100 101110	…	Medium organization

Let us use three cases to show that the packets are forwarded correctly.

a. Assume a packet with the destination address 12.44.185.110 is arrived. The router first extracts the first 23 bits (00001100 00101100 1011100) and check to see if it matches with the first row of the table, which does not. It then checks with the second row, which does not match either. The router next extracts the first 22 bits (00001100 00101100 101110), which matches with the last entry. The packet is correctly forwarded to the interface of the medium organization.

b. Assume a packet with the destination address 12.44.190.25 is arrived. The router first extracts the first 23 bits (00001100 00101100 1011111) and check to see if it matches with the first row of the table, which does not. It then checks with the second row, which does. The packet is correctly forwarded to the interface of second small organization.

c. Assume a packet with the destination address 12.44.189.24 is arrived. The router first extracts the first 23 bits (00001100 00101100 1011110) and check to see if it matches with the first row of the table, which does The packet is correctly forwarded to the interface of first small organization.

P4-24.

a. The number of addresses in the ISP block is $N = 2^{32-20} = 4096$. We can add 4095 (which is $N - 1$) to the first address to find the last one (note that the addition can be done in base 256, as described in Appendix B. In base 256, 4095 is (15.255). We have

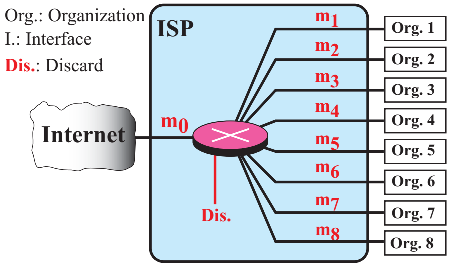
First address: 16.12.64.0/20	**Last address: 16.12.79.255/20**

The prefix length for each organization is $n_i = 32 - \log_2 256 = 24$. We assume that the addresses are allocated from the beginning of the ISP block with each organization consuming 256 addresses. The following shows how addresses are allocated. Note that the prefix for each block is 24 bits.

Block	First address		Last address	*n*
1	16.12.64.0/24	→	16.12.64.255/24	24
2	16.12.65.0/24	→	16.12.65.255/24	24
3	16.12.66.0/24	→	16.12.66.255/24	24
4	16.12.67.0/24	→	16.12.67.255/24	24
5	16.12.68.0/24	→	16.12.68.255/24	24
6	16.12.69.0/24	→	16.12.69.255/24	24
7	16.12.70.0/24	→	16.12.70.255/24	24
8	16.12.71.0/24	→	16.12.71.255/24	24
Unassigned	16.12.72.0/21	→	16.12.79.255/21	21

The unallocated addresses, which can be reserved for the future use of the ISP, are 16.12.72.0/21 to 16.12.79.255/21, for a total of 2048 addresses.

b. The simplified outline is given below. Note that packets having destination addresses with the last prefix in the figure are discarded until these addresses are assigned.



Forwarding table

Prefix	Interface
Default	m_0
0001000 00001100 01000000	m_1
0001000 00001100 01000001	m_2
0001000 00001100 01000010	m_3
0001000 00001100 01000011	m_4
0001000 00001100 01000100	m_5
0001000 00001100 01000101	m_6
0001000 00001100 01000110	m_7
0001000 00001100 01000111	m_8
0001000 00001100 01001	Dis.

P4-25.

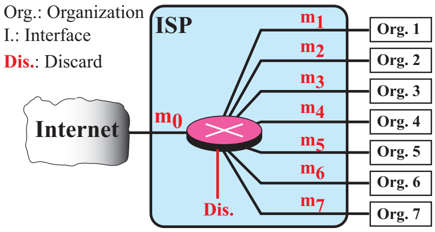
a. The number of addresses in the ISP block is $N = 2^{32-21} = 2048$. We can add 2047 (which is $N - 1$) to the first address to find the last one (note that the addition can be done in base 256, as described in Appendix B. In base 256, 2047 is (7.255). We have

First address: 80.70.56.0/21	Last address: 80.70.63.255/21

b. To make the number of addresses in each block a power of 2 (first CIDR restriction), we assign the following ranges to each organization. The prefix length for each organization is $n_i = 32 - \log_2 N_i$ where N_i is the number of addresses assigned to that organization. Note that the unused addresses cannot fit in a single block (second CIDR restriction).

Block	Size	First address		Last address	*n*
1	512	80.70.56.0/23	→	80.70.57.255/23	23
2	512	80.70.58.0/23	→	80.70.59.255/23	23
3	256	80.70.60.0/24	→	80.70.60.255/24	24
4	256	80.70.61.0/24	→	80.70.61.255/24	24
5	64	80.70.62.0/26	→	80.70.62.63/26	26
6	64	80.70.62.64/26	→	80.70.62.127/26	26
7	64	80.70.62.128/26	→	80.70.62.191/24	26
Unused	320	80.70.62.192	→	80.70.63.255	

c. The simplified outline is given below. Note that to make the forwarding table operable, we need to divide the unused addresses into two blocks. Packets with destination addresses matching the last two prefixes are discarded by the router.



Forwarding table

Prefix	I.
Default	m_0
01010000 01000110 0011100	m_1
01010000 01000110 0011101	m_2
01010000 01000110 00111100	m_3
01010000 01000110 00111101	m_4
01010000 01000110 00111110 00	m_5
01010000 01000110 00111110 01	m_6
01010000 01000110 00111110 10	m_7
01010000 01000110 00111110 11 01010000 01000110 00111111	**Dis.**

P4-26. The total number of addresses in the organization is $N = 2^{32-16} = 65{,}536$.

a. Each subnet can have $N_{sub} = 65{,}536 / 1024 = 64$ addresses.

b. The subnet prefix for each subnet is $n_{sub} = 32 - \log_2 N_{sub} = 32 - 6 = 26$.

c. Now we can calculate the first and the last address in the first subnet. The first address is the beginning address of the block; the last address is the first address plus 63.

First address: 130.56.0.0/26	**Last address: 130.56.0.63/26**

d. To find the first address in subnet 1024, we need to add 65,472 (1023 × 64) in base 256 (0.0.255.192) to the first address in subnet 1. The last address can then be found by adding 63 to the first.

First address: 130. 56.255.192/26	**Last address: 130. 56.255.255/26**

P4-27. The administration can use DHCP to dynamically assign addresses when a host needs to access the Internet. This is possible because, in most organizations, not all of the hosts need to access the Internet at the same time.

P4-28. Both NAT and DHCP can be used for this purpose. DHCP dynamically assigns one of the assigned addresses when a host needs to access the Internet; NAT permanently assigns a set of private addresses to the host, but maps the private address to the global address when a host needs to use the Internet.

P4-29.

a. The *hello message* (type 1) is used by a router to introduce itself to neighboring routers and to introduce already-known neighboring routers to other neighbors.

b. The *data description message* (type 2) is sent in response to a hello message. A router sends its full LSDB to the newly joined router.

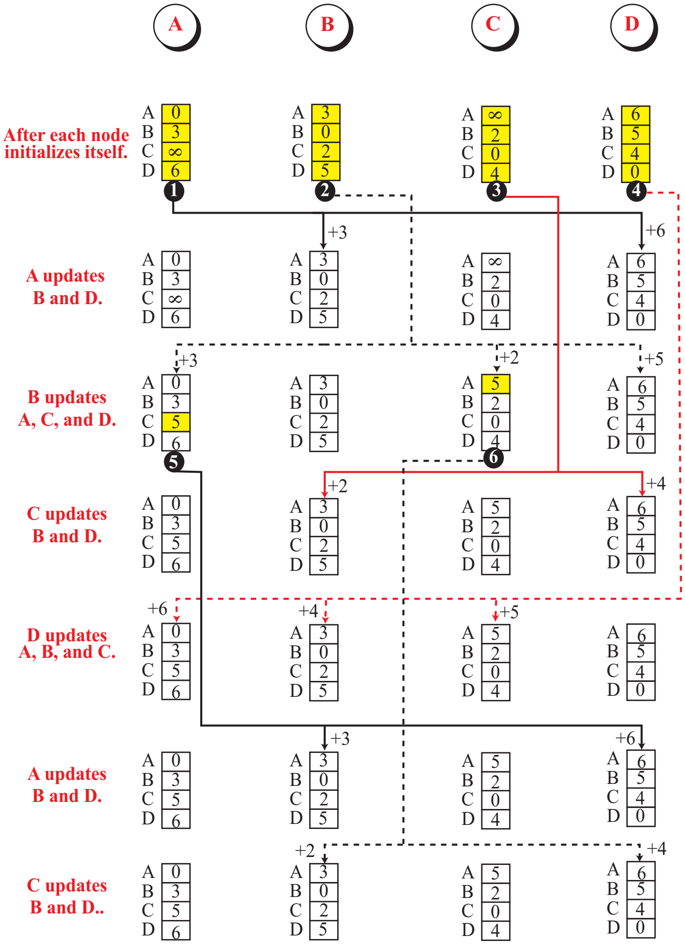
c. The *link-state request message* (type 3) is sent by a router that needs information about a specific LS.

d. The *link-state update message* (type 4) is sent by a router to other routers for building the LSDB. There are five different versions of this message to announce different link states.

e. The *link-state acknowledge message* (type 5) is sent by a router to announce the receiving of a link-state update message. This message is used to provide reliability for the main message used in OSFP.

P4-30. The following shows the initialization and two rounds of updates. Although the process is asynchronous, which means that a node can initialize itself and fire updates to its neighbors at the any time, we have assumed the updates takes place in an orderly way (A, B, C, D). After all nodes has sent their updates, a new round starts and those nodes that have seen any change, will fire updates again. The result should be the same using any other order. The process should stop after there is no change in any node.

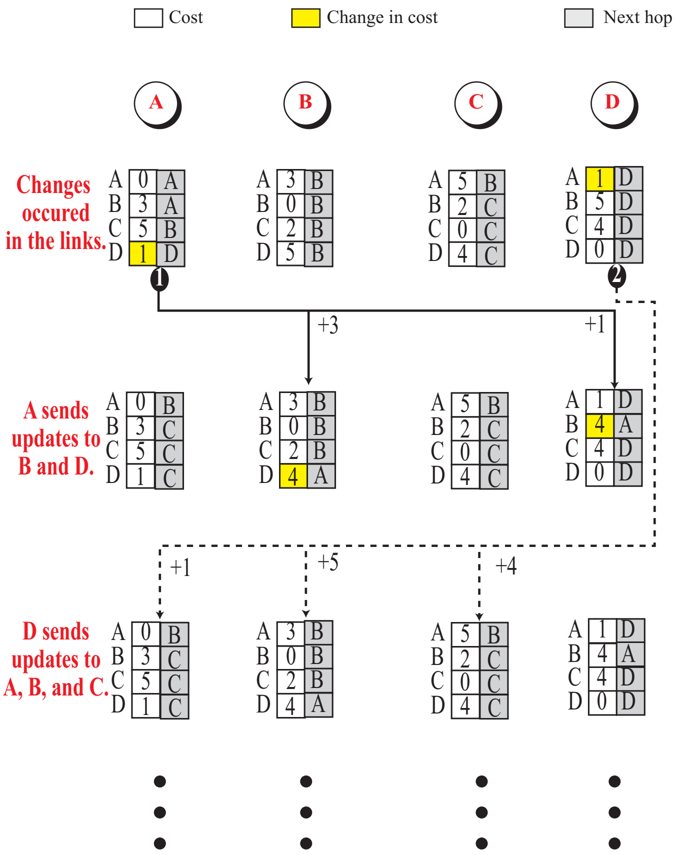
Note: Colored boxes show the changes in the metric.



	A	B	C	D
After each node initializes itself.	A 0 B 3 C ∞ D 6	A 3 B 0 C 2 D 5	A ∞ B 2 C 0 D 4	A 6 B 5 C 4 D 0
A updates B and D.	A 0 B 3 C ∞ D 6	A 3 B 0 C 2 D 5	A ∞ B 2 C 0 D 4	A 6 B 5 C 4 D 0
B updates A, C, and D.	A 0 B 3 C 5 D 6	A 3 B 0 C 2 D 5	A 5 B 2 C 0 D 4	A 6 B 5 C 4 D 0
C updates B and D.	A 0 B 3 C 5 D 6	A 3 B 0 C 2 D 5	A 5 B 2 C 0 D 4	A 6 B 5 C 4 D 0
D updates A, B, and C.	A 0 B 3 C 5 D 6	A 3 B 0 C 2 D 5	A 5 B 2 C 0 D 4	A 6 B 5 C 4 D 0
A updates B and D.	A 0 B 3 C 5 D 6	A 3 B 0 C 2 D 5	A 5 B 2 C 0 D 4	A 6 B 5 C 4 D 0
C updates B and D..	A 0 B 3 C 5 D 6	A 3 B 0 C 2 D 5	A 5 B 2 C 0 D 4	A 6 B 5 C 4 D 0

No more updates need to be sent; the system is stable.

P4-31. Two nodes, A and D, see the changes (see Table 4.4, line 16). These two nodes update their vectors immediately. We assume that changes in each round are fired in the order A, B, C, D. The following shows that the internet is actually stable after two rounds of updates, but more updates are fired to assure the system is stable. We have shown only three columns for each forwarding table, but RIP usually uses more than columns. Also note that we have used the yellow color to show the changed in cost field, which triggers updates. The cost and the next hop fields participate in updating.



Changes occured in the links.

	A		B		C		D	
A	0	A	3	B	5	B	1	D
B	3	A	0	B	2	C	5	D
C	5	B	2	B	0	C	4	D
D	1	D	5	B	4	C	0	D

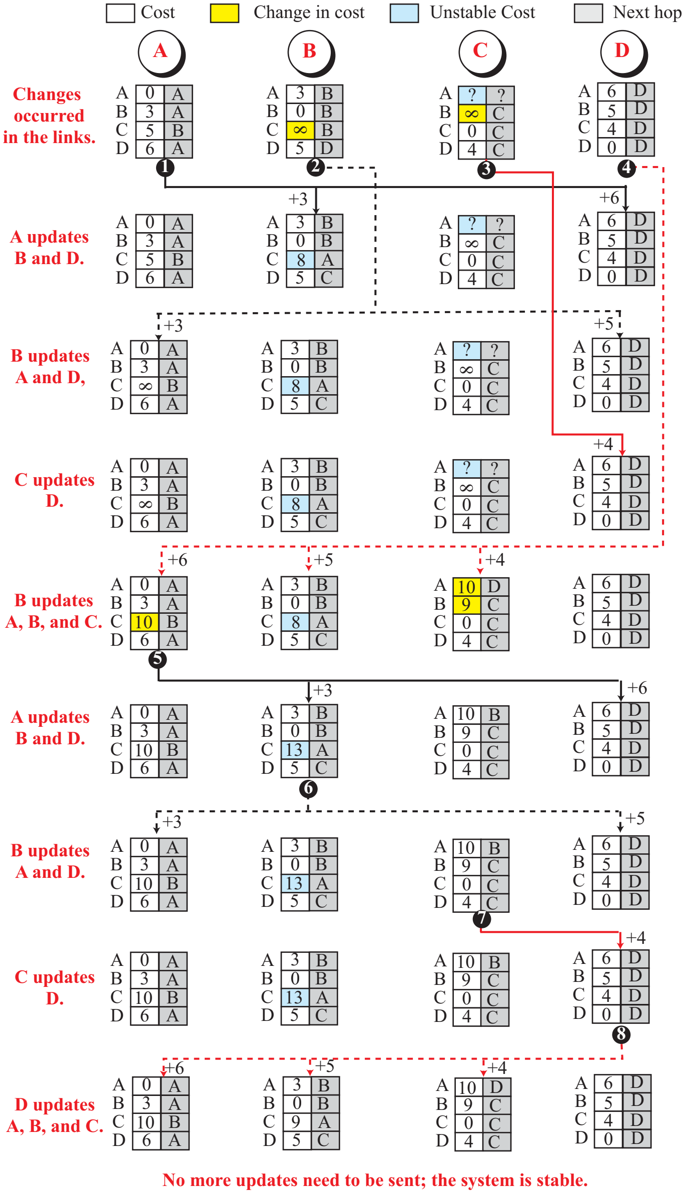
A sends updates to B and D.

	A		B		C		D	
A	0	B	3	B	5	B	1	D
B	3	C	0	B	2	C	4	A
C	5	C	2	B	0	C	4	D
D	1	C	4	A	4	C	0	D

D sends updates to A, B, and C.

	A		B		C		D	
A	0	B	3	B	5	B	1	D
B	3	C	0	B	2	C	4	A
C	5	C	2	B	0	C	4	D
D	1	C	4	A	4	C	0	D

P4-32. The following shows how the forwarding tables will be changed.



Changes occurred in the links.

	A		B		C		D	
A	0	A	3	B	?	?	6	D
B	3	A	0	B	∞	C	5	D
C	5	B	∞	B	0	C	4	D
D	6	A	5	D	4	C	0	D

A updates B and D.

	A		B		C		D	
A	0	A	3	B	?	?	6	D
B	3	A	0	B	∞	C	5	D
C	5	B	8	A	0	C	4	D
D	6	A	5	C	4	C	0	D

B updates A and D,

	A		B		C		D	
A	0	A	3	B	?	?	6	D
B	3	A	0	B	∞	C	5	D
C	∞	B	8	A	0	C	4	D
D	6	A	5	C	4	C	0	D

C updates D.

	A		B		C		D	
A	0	A	3	B	?	?	6	D
B	3	A	0	B	∞	C	5	D
C	∞	B	8	A	0	C	4	D
D	6	A	5	C	4	C	0	D

B updates A, B, and C.

	A		B		C		D	
A	0	A	3	B	10	D	6	D
B	3	A	0	B	9	C	5	D
C	10	B	8	A	0	C	4	D
D	6	A	5	C	4	C	0	D

A updates B and D.

	A		B		C		D	
A	0	A	3	B	10	B	6	D
B	3	A	0	B	9	C	5	D
C	10	B	13	A	0	C	4	D
D	6	A	5	C	4	C	0	D

B updates A and D.

	A		B		C		D	
A	0	A	3	B	10	B	6	D
B	3	A	0	B	9	C	5	D
C	10	B	13	A	0	C	4	D
D	6	A	5	C	4	C	0	D

C updates D.

	A		B		C		D	
A	0	A	3	B	10	B	6	D
B	3	A	0	B	9	C	5	D
C	10	B	13	A	0	C	4	D
D	6	A	5	C	4	C	0	D

D updates A, B, and C.

	A		B		C		D	
A	0	A	3	B	10	D	6	D
B	3	A	0	B	9	C	5	D
C	10	B	9	A	0	C	4	D
D	6	A	5	C	4	C	0	D

No more updates need to be sent; the system is stable.

Note that there are some unstable cost values that are not finalized. These wrong pieces of information may create looping in the system; the packet may bound back and forth until the system becomes stable. Eight updates are needed to stabilize the system.

P4-33. Router R1 has four interfaces. Let us investigate the possibility of a packet with destination 140.24.7.194 from each of these interfaces and see how it is routed.

a. The packet can arrive from one of the interfaces m0, m1, and m2, because one of the computers in organization 1, 2, or 3 could have sent this packet. The prefix /26 is applied to the address, resulting in the network address 140.24.7.192/26. Since none of the network addresses/masks matches this result, the packet is sent to the default router R2.

b. The packet cannot arrive at router R1 from interface m3 because this means that the packet must have arrived from interface m0 of router R2, which is impossible because if this packet arrives at router R2 (from any interface), the prefix length /26 is applied to the destination address, resulting in the network address/mask of 140.24.7.192/26. The packet is sent out from interface m1 and directed to organization 4 and never reaches router R1.

P4-34. The packet is sent to router R1 and eventually to organization 1 as shown below:

a. Router R2 applies the mask /26 to the address (or it extracts the leftmost 26 bits) resulting in the network address/mask of 140.24.7.0/26, which does not match with the first entry in the forwarding table.

b. Router R2 applies the mask /24 to the address (or it extracts the leftmost 24 bits) resulting in the network address/mask of 140.24.7.0/24, which matches with the second entry in the forwarding table. The packet is sent out from interface m0 to router R1.

c. Router R1 applies the mask /26 to the address (or it extracts the leftmost 26 bits) resulting in the network address/mask of 140.24.7.0/26, which matches with the first entry in the forwarding table. The packet is sent out from interface m0 to organization 1.

P4-35. We have

$$\mathbf{D}_{xy} = \mathbf{min}\ \{(\mathbf{c}_{xa} + \mathbf{D}_{ay}), (\mathbf{c}_{xb} + \mathbf{D}_{by}), (\mathbf{c}_{xc} + \mathbf{D}_{cy}), (\mathbf{c}_{xd} + \mathbf{D}_{dy})\}$$

$$\mathbf{D}_{xy} = \mathbf{min}\ \{(2 + 5), (1 + 6), (3 + 4), (1 + 3)\} = \mathbf{min}\ \{7, 7, 7, 4\} = 4$$

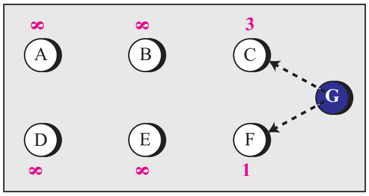
P4-36. At time t_1, we have one periodic timer, ten expiration timers, and no garbage collection timer. An expiration timer becomes invalid after 180 seconds. This means, at time t_2, we have one periodic timer, nine expiration timers, and one garbage collection timer (for the one which has become invalid).

P4-37. The forwarding table for node A can be made using the least-cost tree, as shown below:

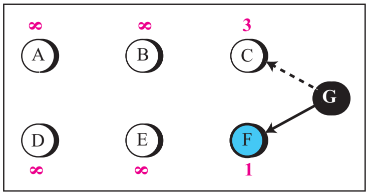
Destination	Cost	Next hop
A	0	—
B	2	—
C	7	B
D	3	—
E	6	B
F	8	B
G	9	B

Forwarding table for node A

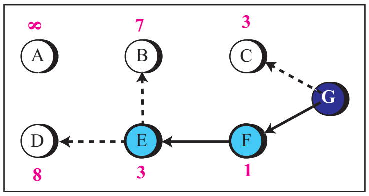
P4-38. The following shows the steps to create the shortest-path tree for node G and the forwarding table for this node.



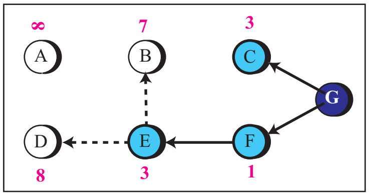
Initialization



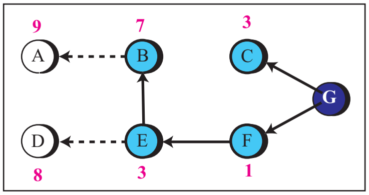
Iteration 1



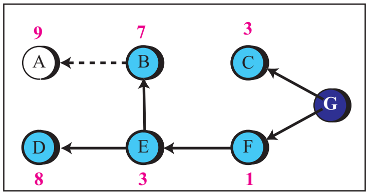
Iteration 2



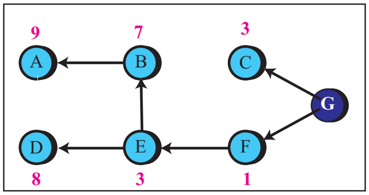
Iteration 3



Iteration 4



Iteration 5

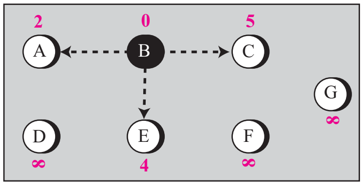


Iteration 6

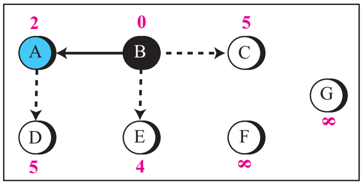
Destination	Cost	Next hop
A	9	F
B	7	F
C	3	—
D	8	F
E	3	F
F	1	—
G	0	—

Forwarding table for node G

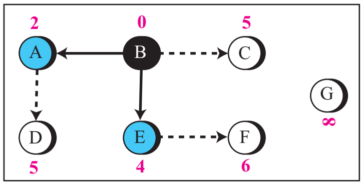
P4-39. The following shows the steps to create the shortest path tree for node B and the forwarding table for this node.



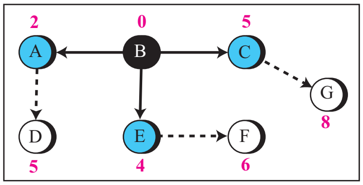
Initialization



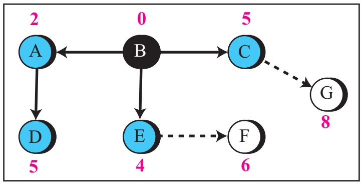
Iteration 1



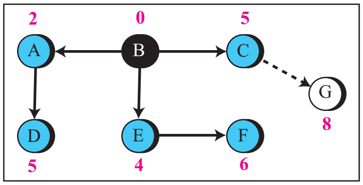
Iteration 2



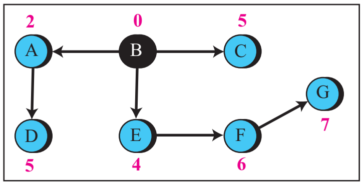
Iteration 3



Iteration 4



Iteration 5

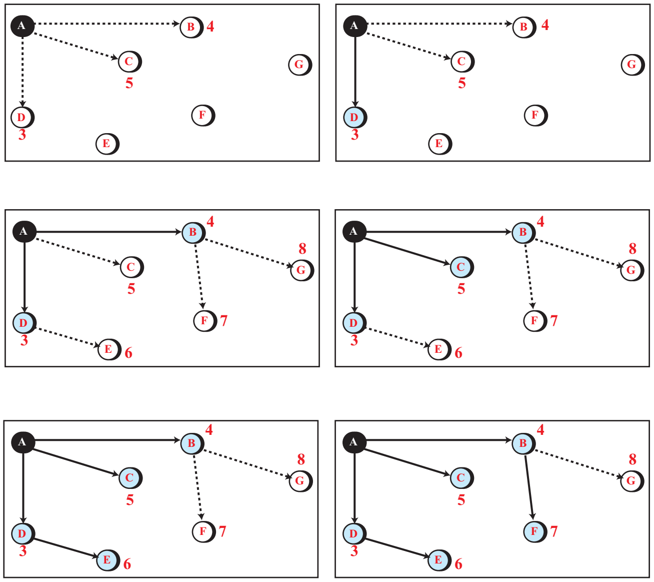


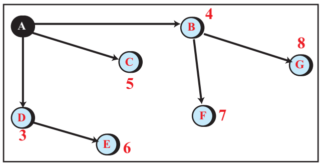
Iteration 6

Destination	Cost	Next hop
A	2	———
B	0	———
C	5	———
D	5	A
E	4	———
F	6	E
G	8	E

Forwarding table for node B

P4-40. The following shows the shortest-path tree and the forwarding table for node A.





Destination	Cost	Next hop
A	0	———
B	4	———
C	5	———
D	3	———
E	6	D
F	7	B
G	8	B

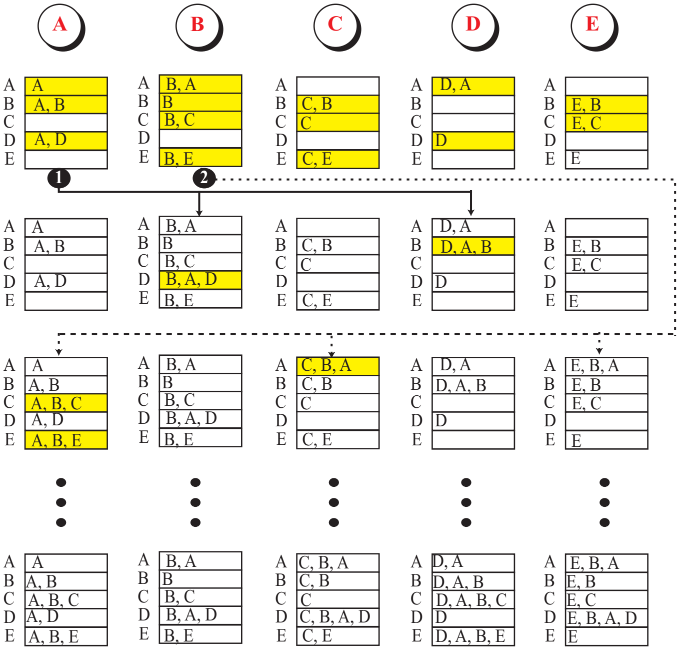
Forwarding table for node A

P4-41. The number of searches in each iteration of Dijkstra's algorithm is different. In the first iteration, we need *n* number of searches, in the second iteration, we need (*n* – 1), and finally in the last iteration, we need only one. In other words, the total number of searches for each node to find its own shortest-path tree is

Number of searches $= n + n - 1 + n - 2 + n - 3 + \ldots + 3 + 2 + 1 = n\,(n + 1) / 2$

The series can be calculated if it is written twice: once in order and once in the reverse order. We then have *n* items, each of value (*n* + 1), which results in *n*(*n* + 1). However, we need to divide the result by 2. In computer science, this complexity is written as $O(n^2)$ and is referred to as Big-O notation.

P4-42. The following shows the initialization and updates.



	A	B	C	D	E
A	A	B, A		D, A	
B	A, B	B	C, B		E, B
C		B, C	C		E, C
D	A, D			D	
E		B, E	C, E		E

	A	B	C	D	E
A	A	B, A		D, A	
B	A, B	B	C, B	D, A, B	E, B
C		B, C	C		E, C
D	A, D	B, A, D		D	
E		B, E	C, E		E

	A	B	C	D	E
A	A	B, A	C, B, A	D, A	E, B, A
B	A, B	B	C, B	D, A, B	E, B
C	A, B, C	B, C	C		E, C
D	A, D	B, A, D		D	
E	A, B, E	B, E	C, E		E

	A	B	C	D	E
A	A	B, A	C, B, A	D, A	E, B, A
B	A, B	B	C, B	D, A, B	E, B
C	A, B, C	B, C	C	D, A, B, C	E, C
D	A, D	B, A, D	C, B, A, D	D	E, B, A, D
E	A, B, E	B, E	C, E	D, A, B, E	E

P4-43. The number of operations in each iteration of the algorithm is *n*, in which *n* is the number of nodes in the network. In computer science, this complexity is written as O(*n*) and is referred to as Big-O notation.

P4-44. The following shows the advertisement in each case (a triplet defines the destination, cost, and the next hop):

a. From A to B: (A, 0, A), (B, ∞, A), (C, 4, A), (D, ∞, B).

b. From C to D: (A, 4, C), (B, ∞, D), (C, 0, C), (D, ∞, C).

c. From D to B: (A, ∞, B), (B, ∞, D), (C, 6, D), (D, 0, D).

d. From C to A: (A, ∞, C), (B, 8, D), (C, 0, C), (D, 6, C).

P4-45. The following shows the advertisement in each case (a triplet defines the destination, cost, and the next hop):

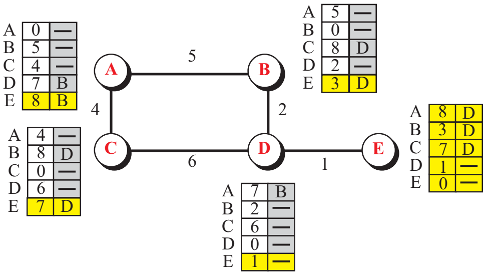
a. From A to B: (A, 0, A), (C, 4, A).

b. From C to D: (A, 4, C), (C, 0, C).

c. From D to B: (C, 6, D), (D, 0, D).

d. From C to A: (B, 8, D), (C, 0, C), (D, 6, C).

P4-46. We can guess the new routing table because the only way each node can reach node E is via D. The following shows the new network and the forwarding tables. Note that we only add an entry to each forwarding table for nodes A, B, C, and D. The forwarding table for node E is totally new.



A	0	—
B	5	—
C	4	—
D	7	B
E	8	B

A	5	—
B	0	—
C	8	D
D	2	—
E	3	D

A	4	—
B	8	D
C	0	—
D	6	—
E	7	D

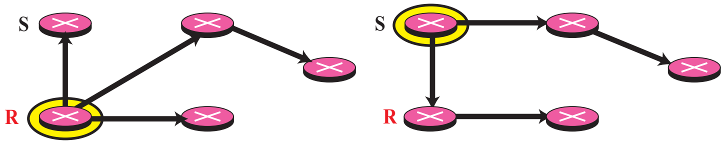
A	7	B
B	2	—
C	6	—
D	0	—
E	1	—

A	8	D
B	3	D
C	7	D
D	1	—
E	0	—

P4-47. The packet that has arrived through m2 should be forwarded because if R3 wants to send a packet to the source, S, in the reverse path, it will send it through the interface m2.

P4-48. This is where IGMP can help. Using IGMP, the router should collect information about the interests of networks. When a packet with a multicast destination address G arrives, the router should send the packet only from those interfaces connected to the interested networks.

P4-49. The following shows the two cases. In the first case, router R is the root of the unicast communication. In the second case, router S is the root of the multicast communication.



a. OSPF: R is the root.

b. MOSPF: S is the root.

P4-50. See Figure 4.72.

a. The size of a RIP message that advertises a single network is 24 [4 + (20 × 1)] bytes.

b. The size of a RIP message that advertises *n* networks is [4 + (20 × *n*)] bytes.

P4-51. Router R1, using its OSPF forwarding table, knows how to forward a packet destined for N4. R1 announces this reachability to R5 using an eBGP session. R5 adds an entry to its RIP forwarding table that shows R1 as the next router for any packet destined for N4.

P4-52. Router R9 knows how to reach N13 through its RIP forwarding table. R9 advertises this reachability to R4 using an eBGP session. R4 in turn advertises this reachability to R2 using an iBGP session. R2 then advertises this reachability to R6 using an eBGP session. R6 advertises its reachability to R8. R8 adds an entry to its RIP forwarding table to show that any packet destined for N13 should be forwarded to R6.

P4-53.

a. 0000:0000:0000:0000:5555:5555:5555:5555
b. 0000:0000:0000:0000:AAAA:AAAA:AAAA:AAAA
c. 5555:5555:5555:5555:5555:5555:5555:5555
d. 7777:7777:7777:7777:7777:7777:7777:7777

P4-54.

a. 0:FFFF:FFFF::
b. 1234:2346:3456::FFFF
c. 0:1::FFFF:1200:1000
d. ::FFFF:FFFF:24.123.12.6

P4-55.

a. 0000:0000:0000:0000:0000:0000:0000:2222
b. 1111:0000:0000:0000:0000:0000:0000:0000
c. 000B:000A:00CC:0000:0000:0000:1234:000A

P4-56.

a. 0000:0000:0000:0000:0000:0000:0000:0002
b. 0000:0023:0000:0000:0000:0000:0000:0000
c. 0000:000A:0000:0000:0000:0000:0000:0003

P4-57. One way to solve the problem is to write the binary notation in each case and keep the block prefix:

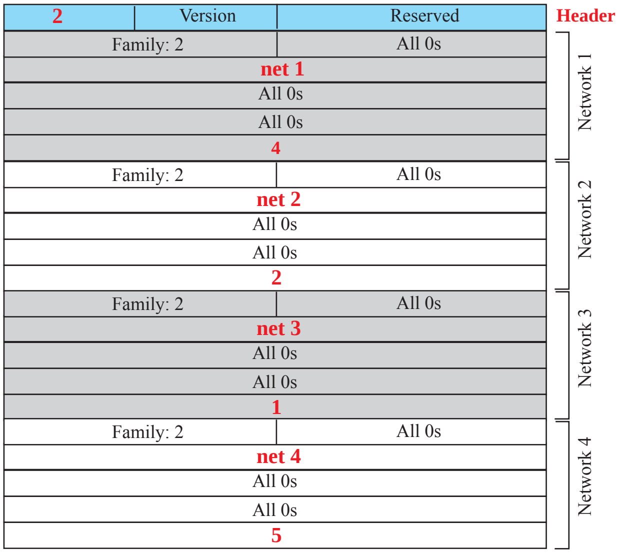
a. 1111 1110 1000 0000 ... 0001 0010 **Link local**
b. 1111 1101 0010 0011 ... 0000 0000 **Unique local unicast**
c. 0011 0010 0000 0000 ... 0000 0000 **Global unicast**

P4-58. Both subnets keep the given *global routing prefix*. Each subnet adds a 16-bit subnet identifier. We assume the subnet identifiers start from $(0001)_{16}$, but they can also start from $(0000)_{16}$.

a. The first subnet block is 2000:1234:1423:**0001**/64.

b. The second subnet block is 2000:1234:1423:**0002**/64.

P4-59. The following shows the contents of the RIP message (See Figure 4.72).



2	Version	Reserved	Header
Family: 2		All 0s	Network 1
net 1			
All 0s			
All 0s			
4			
Family: 2		All 0s	Network 2
net 2			
All 0s			
All 0s			
2			
Family: 2		All 0s	Network 3
net 3			
All 0s			
All 0s			
1			
Family: 2		All 0s	Network 4
net 4			
All 0s			
All 0s			
5			

P4-60. The following table shows the comparison:

Field	*IPv4*	*IPv6*
VER	√	√
HLEN	√	
Service (or traffic class)	√	√
Flow label		√
Total length	√	
Payload length		√
Identification	√	
Flags	√	
Flag offset	√	
TTL (or hop limit)	√	√
Protocol	√	
Checksum	√	
Source Address	√	√
Destination Address	√	√