CHAPTER 4

PRACTICE SET

Questions

Q4-1. 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.

Q4-3. 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.

Q4-5. The number of virtual circuits is $2^8 = 256$.

Q4-7. None of these services are implemented for the IP protocol in order to make it simple.

Q4-9. The throughput is the smallest transmission rate, or 140 Kbps. The bottleneck is now the link between the source host and R1.

Q4-11. The identification numbers need to be contiguous. The identification number of the last datagram should be 1024 + 100 – 1 = 1123.

Q4-13. 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).

$\boldsymbol{N}$ **= (last address) − (first address) + 1**
$\boldsymbol{n}$ **= 32 − log**$_2$$\boldsymbol{N}$
Block: (first address)/n

Q4-15. 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**

Q4-17. 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).

Q4-19. The header length is 6 × 4 = 24. The option length is then 24 – 20 = 4 bytes.

Q4-21. 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).

Q4-23. 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.

Q4-25. 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.

Q4-27. According to the principle we mentioned in the text, the shortest path is the inverse of the original one. The shortest path is G → E → B → A.

Q4-29. 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.

Q4-31. 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.

Q4-33. 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.

Q4-35. 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.

Q4-37. 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.

Q4-39. 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.

Q4-41. 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.

Q4-43. It cannot. A link needs to be advertised in a router link LSP; a network needs to be advertised in a network link LSP.

Q4-45. 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.

Q4-47. 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.

Q4-49. 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.

Q4-51. 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.

Q4-53. The group list is the union of the individual lists; it is {G1, G2, G3, G4}.

Q4-55.

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.

Q4-57. 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.

Q4-59. 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.

Q4-61. 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.

Q4-63. 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.

Q4-65. The three protocols IGMP, ICMP, and ARP in IPv4 have been combined into a single protocol, ICMPv6.

Problems

P4-1. The total length of the datagram is $(00A0)_{16}$ = 160 bytes. The header length is 5 × 4 = 20. The size of the payload is then 160 – 20 = 140. The efficiency = 140 / 160 = 87.5%.

P4-3. 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

P4-5. 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.

P4-7. We change each byte to the corresponding binary representation:

a. **01101110 00001011 00000101 01011000**

b. **00001100 01001010 00010000 00010010**

c. **11001001 00011000 00101100 00100000**

P4-9. 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.

P4-11. 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 $N = 2^{32}$.

P4-13. The prefix can be found as $n = 32 - \log_2 N$:

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

P4-15. 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**

P4-17. 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

P4-19. 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.

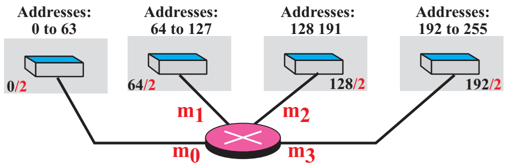
P4-21. The total number of addresses is $2^8 = 256$. This means we have 64 addresses for each network. We can divide the whole address space into four blocks (blocks 0 to 3), each of 64 addresses. The addresses in each block are allocated as (0 to 63), (64 to 127), (128 to 191), and (192 to 255). It 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

P4-23. 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:

$\boldsymbol{N_0}$: 0 / 64 = 0 $\boldsymbol{N_1}$: 64 / 256 = 0.25 $\boldsymbol{N_2}$: 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

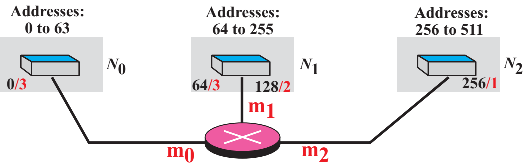
$$\boldsymbol{n_0 = 9 - \log_2 64 = 3} \qquad \boldsymbol{n_{1\text{-}1} = 9 - \log_2 64 = 3}$$

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

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 *N*1 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 *N*1 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

P4-25. 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.

P4-27.

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

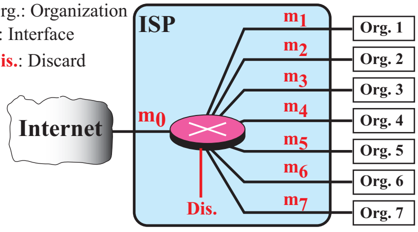
First address: 80.70.56.0/21 **Last address: 80.70.63.255/21**

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.

Org.: Organization
I.: Interface
Dis.: Discard



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.**

P4-29. 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.

P4-31. We have

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

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

P4-33.

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.

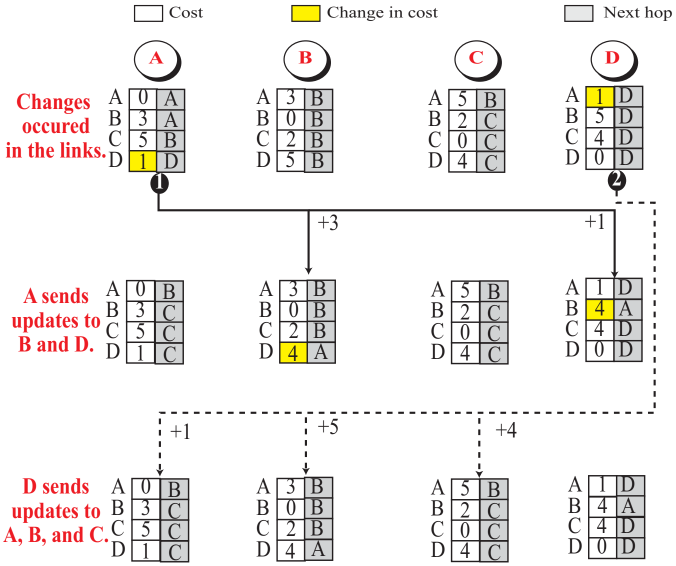
b. The *data description message* (type 2) is sent in response to a hello message. A router sends its full LSDB to the newly joined router.

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.

P4-35. 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.



P4-37. 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.

P4-39. 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).

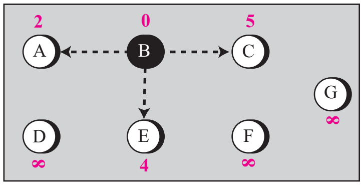
d. From C to A: (B, 8, D), (C, 0, C), (D, 6, C).

P4-41. The forwarding table for node A can be made using the least-cost tree, as shown below:

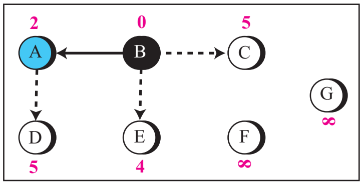
Forwarding table for node A

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

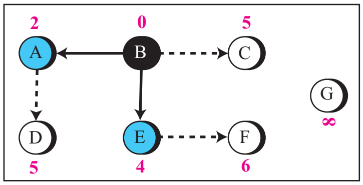
P4-43. The following shows the steps to create the shortest path tree for node B and the forwarding table for this node.



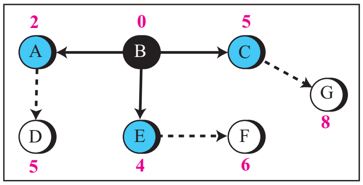
Initialization



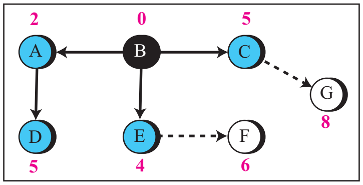
Iteration 1



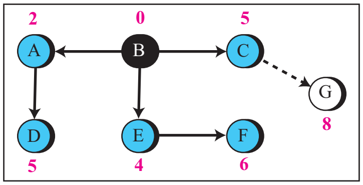
Iteration 2



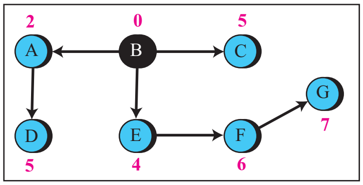
Iteration 3



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

P4-45. 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

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

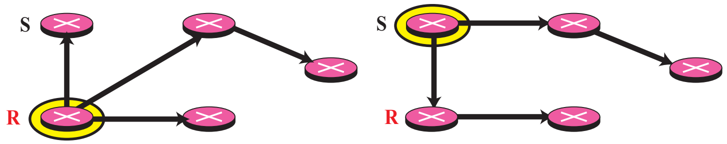
The series can be calculated if it is written twice: once in order and once in the reverse order. We then have *n* items, each of value (*n* + 1), which results in *n*(*n*

+ 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.

P4-47. 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.

P4-49. 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.

P4-51. 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.

P4-53. 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			

P4-55.

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`

e.

P4-57.

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`

P4-59. 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**