MODULE III

Network Layer Protocol

Lecture Notes

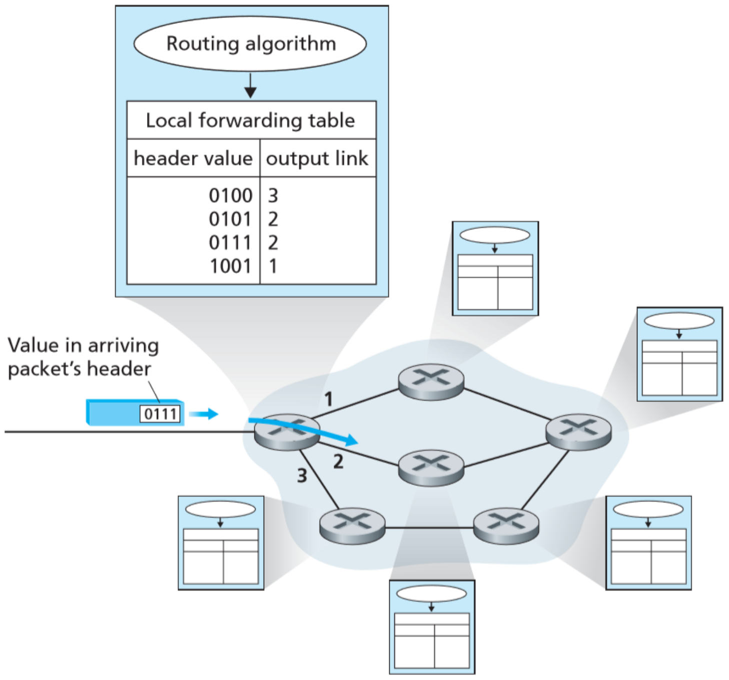
Aparna S Balan

2021

VIDYA ACADEMY OF SCEINCE AND TECHNOLOGY

Introduction

The role of the network layer is to move packets from a sending host to a receiving host. Two important functions that help network-layer to achieve are – forwarding and routing. When a packet arrives at a network component's input link, it must move the packet to the appropriate output link. This is called **forwarding**. The network layer must determine the route or path taken by packets as they flow from a sender to a receiver. This process is called **routing**.



Every router has a **forwarding table**. A router forwards a packet by examining the value of a field in the arriving packet's header, and then uses this header value to search through the router's forwarding table. The value stored in the forwarding table entry for that header indicates the router's outgoing link interface to which that packet is to be forwarded. A router receives routing protocol messages, which are used to configure its forwarding table. The routing algorithm may be **centralized** (with an algorithm executing on a central site and downloading routing information to each of the routers) or **decentralized** (with a piece of the distributed routing algorithm running in each router).

The **network service model** defines the characteristics of end-to-end transport of packets between sending and receiving end systems. Services that could be provided by the network layer include : Guaranteed delivery, Guaranteed delivery with bounded delay, In-order packet delivery, Guaranteed minimal bandwidth, Guaranteed maximum jitter and Security services. The different service models are Best Effort, Constant Bit Rate (CBR) and Available Bit Rate(ABR).

The Internet's network layer provides **best-effort** service. In best-effort service, timing between packets is not guaranteed to be preserved, packets are not guaranteed to be received in the order in which they were sent, nor is the eventual delivery of transmitted packets guaranteed. ATM provides **CBR (Constant bit rate)** network service model. With CBR service, a flow of ATM cells is carried across the network in such a way that a cell's end-to-end delay, the variability in a cell's end-to-end delay (jitter), and the fraction of cells that are lost or delivered late are all guaranteed to be less than specified values. ATM use another service model, that is **ABR (available bit rate)**. ABR is slightly-better-than-best-effort service. Cells may be lost under ABR service. Cells cannot be reordered in ABR and a minimum cell transmission rate (MCR) is guaranteed for a connection using ABR service.

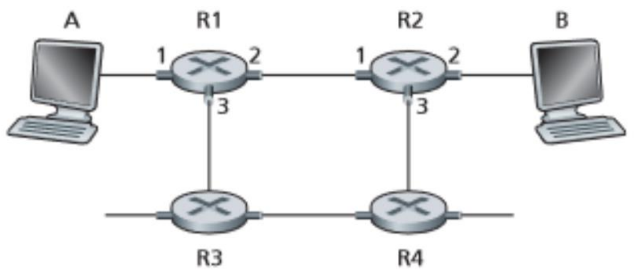
Virtual Circuits and Datagrams

A network-layer can provide connectionless service or connection service between two hosts, which is similar to the transport-layer. Virtual-circuit and datagram networks are two fundamental classes of computer networks.

Virtual Circuit Network

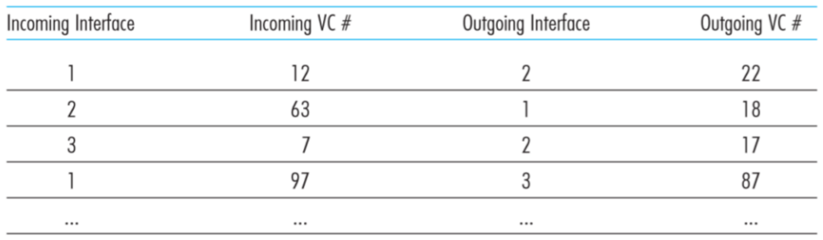
A Virtual Circuit consists of (1) a path (that is, a series of links and routers) between the source and destination hosts, (2) VC numbers, one number for each link along the path, and (3) entries in the forwarding table in each router along the path.

A packet belonging to a virtual circuit will carry a VC number in its header. Because a virtual circuit may have a different VC number on each link, each intervening router must replace the VC number of each traversing packet with a new VC number. The new VC number is obtained from the forwarding table.



The numbers next to the links of R1 are the link interface numbers. Suppose that Host A requests that the network establish a VC between itself and Host B. The network chooses the path A-R1-R2-B and assigns VC numbers 12, 22, and 32 to the three links in this path for this virtual circuit. In this case, when a packet in this VC leaves Host A, the value in the VC number field in the packet header is 12; when it leaves R1, the value is 22; and when it leaves R2, the value is 32.

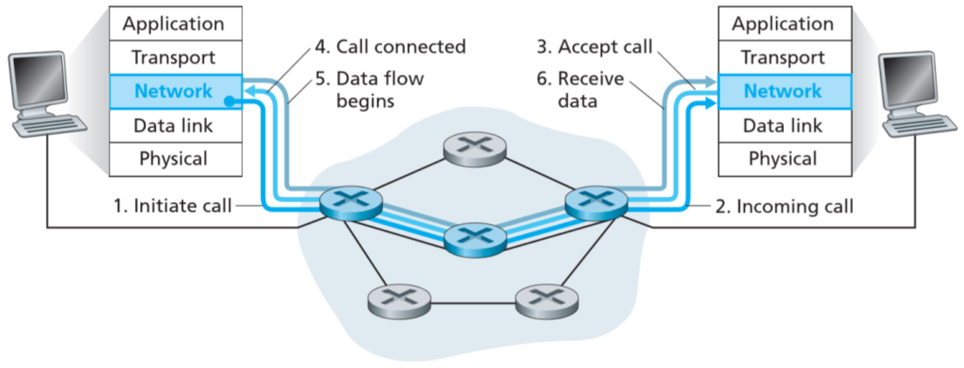
For a VC network, each router's forwarding table includes VC number translation. This is the forwarding table in R1.



Incoming Interface	Incoming VC #	Outgoing Interface	Outgoing VC #
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

The VC number in the packet is updated because of two reasons. First, replacing the number from link to link reduces the length of the VC field in the packet header. Second, VC setup is considerably simplified by permitting a different VC number at each link along the path of the VC.

In a VC network, the network's routers must maintain **connection state information** for the ongoing connections. Each time a new connection is established across a router, a new connection entry must be added to the router's forwarding table; and each time a connection is released, an entry must be removed from the table.

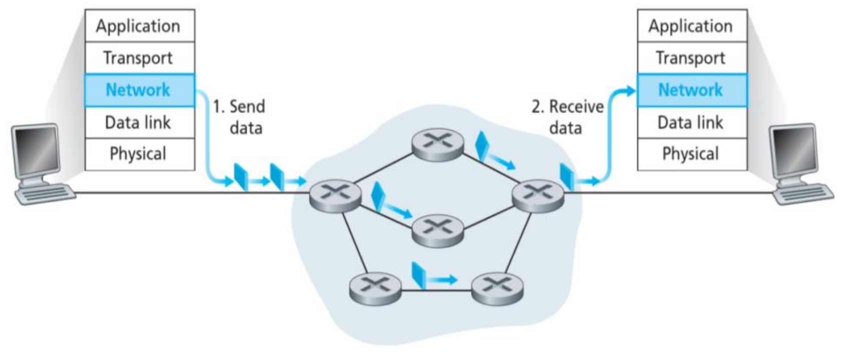


There are **three phases in virtual circuit**: VC setup, Data transfer and VC teardown. During the **VC setup phase**, the sending transport layer contacts the network layer, specifies the receiver's address, and waits for the network to set up the VC. The network layer determines the path between sender and receiver. The network layer also determines the VC number for each link along the path. Finally, the network layer adds an entry in the forwarding table in each router along the path. During VC setup, the network layer may also reserve resources along the path of the VC. In the **data transfer phase**, once the VC has been established, packets can begin to flow along the VC. The **VC teardown phase** is initiated when the sender (or receiver) informs the network layer of its desire to terminate the VC. The network layer will then typically inform the end system on the other side of the network of the call termination and update the forwarding tables in each of the packet routers on the path to indicate that the VC no longer exists.

The messages that the end systems send into the network to initiate or terminate a VC, and the messages passed between the routers to set up the VC are known as **signaling messages** and the protocols used to exchange these messages are often referred to as **signaling protocols**.

Datagram Network

In a datagram network, each time an end system wants to send a packet; it stamps the packet with the address of the destination end system and then pops the packet into the network. There is no VC setup and routers do not maintain any VC state information.



When a packet is transmitted from source to destination, it passes through a series of routers. Each of these routers uses the packet's destination address to forward the packet. Specifically, each router has a forwarding table that maps destination addresses to link interfaces; when a packet arrives at the router, the router uses the packet's destination address to look up the appropriate output link interface in the forwarding table. The router then intentionally forwards the packet to that output link interface.

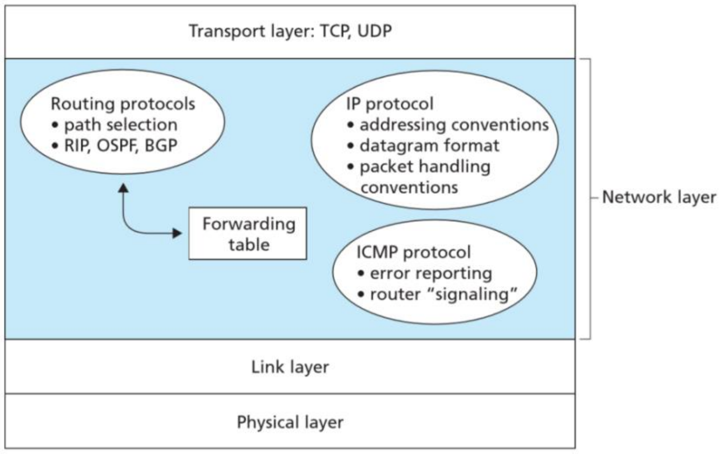
Prefix Match	**Link Interface**
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

The forwarding table stores the prefix of the ip address with the link interface number. The router matches a prefix of the packet's destination address with the entries in the table; if there's a match, the router forwards the packet to a link associated with the match. In the given table, if prefix doesn't match any of the first three entries, then the router forwards the packet to interface 3. When there are multiple matches, the router uses the **longest prefix matching rule**; that is, it finds the longest matching entry in the table and forwards the packet to the link interface associated with the longest prefix match.

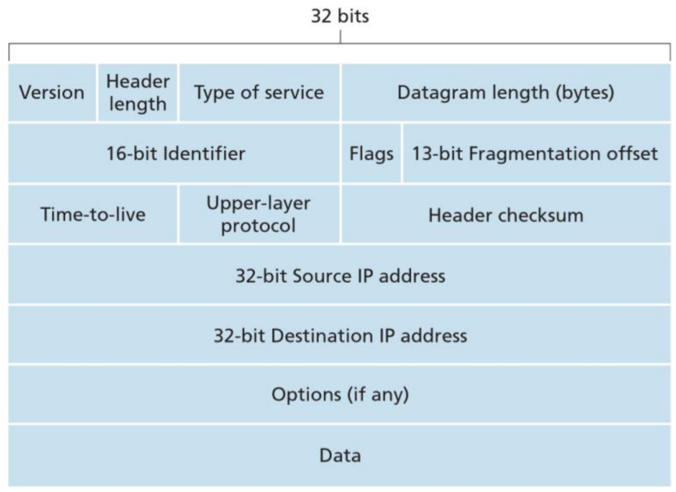
Because forwarding tables in datagram networks can be modified at any time, a series of packets sent from one end system to another may follow different paths through the network and may arrive out of order.

IPv4

The Internet's network layer has three major components. The first component is the IP protocol. The second major component is the routing component, which determines the path a datagram follows from source to destination. The final component of the network layer is a facility to report errors in datagrams and respond to requests for certain network-layer information.



Internet addressing and forwarding are important components of the Internet Protocol (IP). There are two versions of IP: IPv4 and IPv6. A network-layer packet is referred to as a datagram. An IP datagram has a total of 20 bytes of header (assuming no options). The IPv4 datagram format is given below.



Version number: These 4 bits specify the IP protocol version of the datagram. By looking at the version number, the router can determine how to interpret the remainder of the IP datagram. Different versions of IP use different datagram formats.

Header length: Because an IPv4 datagram can contain a variable number of options; these 4 bits are needed to determine where in the IP datagram the data actually begins. Most IP datagrams do not contain options, so the typical IP datagram has a 20-byte header.

Type of service: The type of service (TOS) bits were included in the IPv4 header to allow different types of IP datagrams (for example, datagrams particularly requiring low delay, high throughput, or reliability) to be distinguished from each other.

Datagram length: This is the total length of the IP datagram (header plus data), measured in bytes. This field is 16 bits long.

Identifier, flags, fragmentation offset: These three fields are used in IP fragmentation. The new version of IP, IPv6, does not allow for fragmentation at routers.

Time-to-live: The time-to-live (TTL) field is included to ensure that datagrams do not circulate forever in the network. This field is decremented by one each time the datagram is processed by a router. If the TTL field reaches 0, the datagram must be dropped.

Protocol: This field is used only when an IP datagram reaches its final destination. The value of this field indicates the specific transport-layer protocol to which the data portion of this IP datagram should be passed. A value of 6 indicates that the data portion is passed to TCP, while a value of 17 indicates that the data is passed to UDP.

Header checksum: The header checksum aids a router in detecting bit errors in a received IP datagram. The header checksum is computed by treating each 2 bytes in the header as a number and summing these numbers using 1s complement arithmetic.

Source and destination IP addresses: When a source creates a datagram, it inserts its IP address into the source IP address field and inserts the address of the ultimate destination into the destination IP address field.

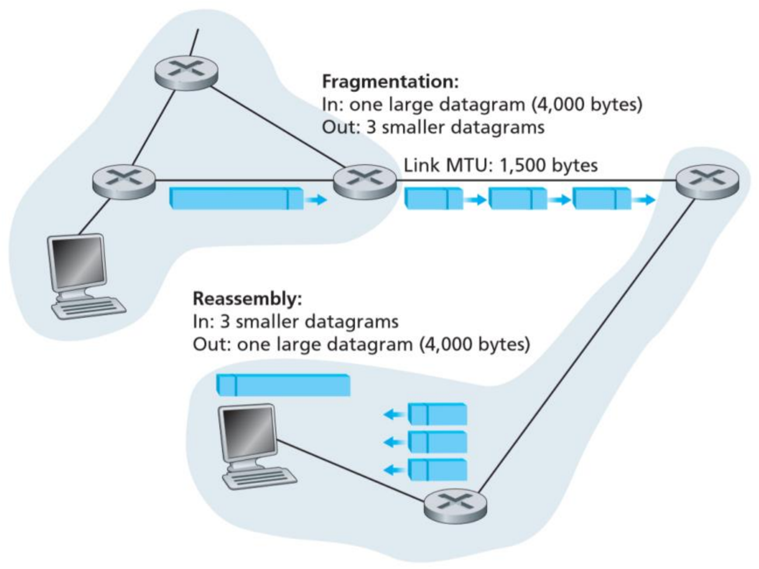
Options: The options fields allow an IP header to be extended.

Data (payload): The data field of the IP datagram contains the transport-layer segment (TCP or UDP) to be delivered to the destination.

Not all link-layer protocols can carry network-layer packets of the same size. The maximum amount of data that a link-layer frame can carry is called the **maximum transmission unit (MTU)**. Because each IP datagram is encapsulated within the link-layer frame, the MTU of the link-layer protocol places a hard limit on the length of an IP datagram. The data in the IP datagram is fragmented into two or smaller IP datagrams, each of these smaller IP datagrams are encapsulated in a separate link-layer frame; and send over the outgoing link. Each of these smaller datagrams is referred to as a **fragment**. This process is called **fragmentation**.

Fragments need to be reassembled before they reach the transport layer at the destination. The job of datagram reassembly is done in the end systems rather than in network routers. When a destination host receives a series of datagrams from the same source, it needs to determine whether any of these datagrams are fragments of some original, larger datagram and how the fragments it has received should be pieced back together to form the original datagram. The IP (version 4) uses identification, flag, and fragmentation offset fields in the IP datagram header for the reassembly purpose.

When a router needs to fragment a datagram, each resulting datagram is stamped with the source address, destination address, and identification number of the original datagram. When the destination receives a series of datagrams from the same sending host, it can examine the identification numbers of the datagrams to determine which of the datagrams actually fragments of the same larger datagram.



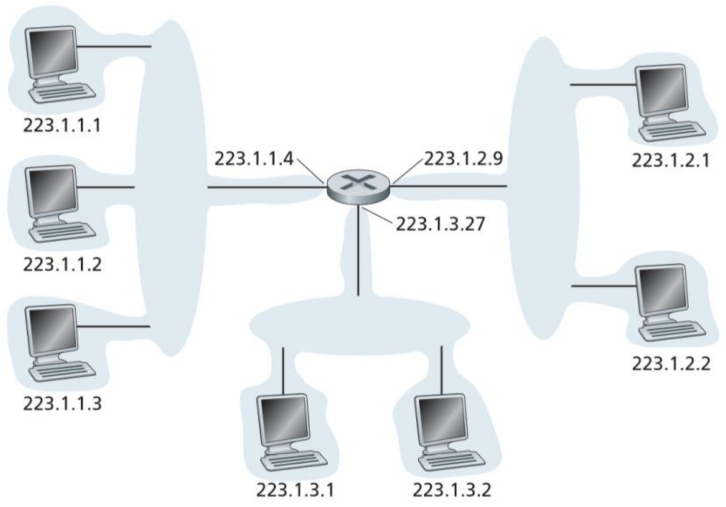
The last fragment of a datagram has a flag bit set to 0, whereas all the other fragments have this flag bit set to 1. It helps the destination host to make sure that it has received the last fragment of the original datagram. In order for the destination host to determine whether a fragment is missing, the offset field is used to specify where the fragment fits within the original IP datagram.

Consider an example where a datagram of 4,000 bytes (20 bytes of IP header plus 3,980 bytes of IP payload) arrives at a router and must be forwarded to a link with an MTU of 1,500 bytes. This implies that the 3,980 data bytes in the original datagram must be allocated to three separate fragments. Suppose that the original datagram is stamped with an identification number of 777. The values in table reflect the requirement that the amount of original payload data in all but the last fragment be a multiple of 8 bytes, and that the offset value be specified in units of 8-byte chunks. At the destination, the payload of the datagram is passed to the transport layer only after the IP layer has fully reconstructed the original IP datagram. If one or more of the fragments does not arrive at the destination, the incomplete datagram is discarded and not passed to the transport layer.

Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in the data field of the IP datagram	identification = 777	offset = 0 (meaning the data should be inserted beginning at byte 0)	flag = 1 (meaning there is more)
2nd fragment	1,480 bytes of data	identification = 777	offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that 185 · 8 = 1,480)	flag = 1 (meaning there is more)
3rd fragment	1,020 bytes (= 3,980–1,480–1,480) of data	identification = 777	offset = 370 (meaning the data should be inserted beginning at byte 2,960. Note that 370 · 8 = 2,960)	flag = 0 (meaning this is the last fragment)

A host typically has only a single link into the network; when IP in the host wants to send a datagram, it does so over this link. The boundary between the host and the physical link is called an **interface**. Now consider a router and its interfaces. A router has two or more links to which it is connected. The boundary between the router and any one of its links is also called an interface. A router has multiple interfaces, one for each of its links. Each host and router interface has its own IP address. Thus, an IP address is technically associated with an interface, rather than with the host or router containing that interface.

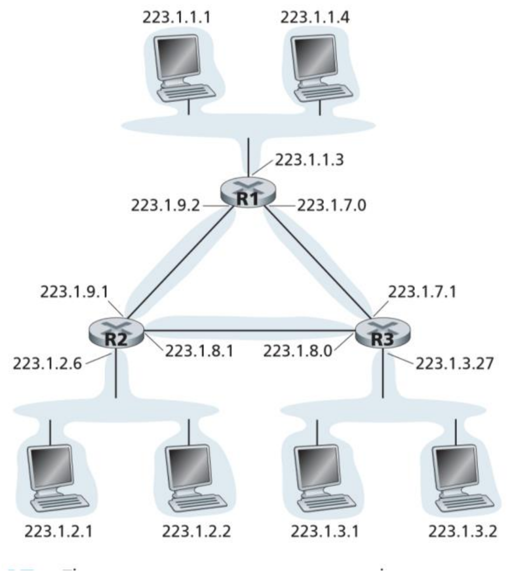
Each IP address is 32 bits long (4 bytes), and there are a total of 2^{32} possible IP addresses. These addresses are typically written in **dotted-decimal notation**, in which each byte of the address is written in its decimal form and is separated by a period (dot) from other bytes in the address. A portion of an interface's IP address will be determined by the subnet to which it is connected.



In this figure one router (with three interfaces) is used to interconnect seven hosts. The three hosts in the upper-left portion and the router interface, to which they are connected, all have an IP address of the form 223.1.1.xxx. That is, they all have the same leftmost 24 bits in their IP address. The four interfaces are also interconnected to each other by a network that contains no routers. This network interconnecting three host interfaces and one router interface forms a **subnet**. All the devices on a given subnet will have the same subnet address. A subnet is also called an IP network.

IP addressing assigns an address to this subnet: 223.1.1.0/24, where the /24 notation, sometimes known as a **subnet mask**, indicates that the leftmost 24 bits of the 32-bit quantity define the subnet address. The subnet 223.1.1.0/24 consists of the three host interfaces (223.1.1.1, 223.1.1.2, and 223.1.1.3) and one router interface (223.1.1.4). Any additional hosts attached to the 223.1.1.0/24 subnet would be required to have an address of the form 223.1.1.xxx.

The following figure shows three routers that are interconnected with each other by point-to-point links. Each router has three interfaces, one for each point-to-point link and one for the broadcast link that directly connects the router to a pair of hosts. Each interface is assigned with a different IP address.



This network has three subnets, 223.1.1.0/24, 223.1.2.0/24, and 223.1.3.0/24. There are three subnet masks also: one subnet, 223.1.9.0/24, for the interfaces that connect routers R1 and R2; another subnet, 223.1.8.0/24, for the interfaces that connect routers R2 and R3; and a third subnet, 223.1.7.0/24, for the interfaces that connect routers R3 and R1

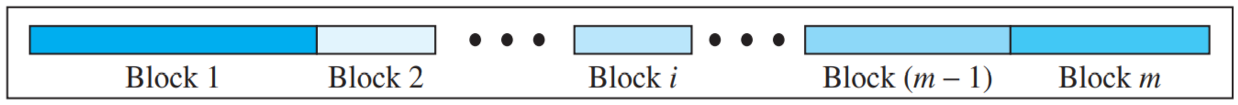
With subnet addressing, the 32-bit IP address is divided into two parts and has the dotted-decimal form a.b.c.d/x, where x indicates the number of bits in the first part of the address. The remaining 32-x bits of an address can be thought of as distinguishing among the devices within the organization. The 'x' most significant bits of an address of the form a.b.c.d/x constitute the network portion of the IP address, and are often referred to as the prefix (or network prefix) of the address. The IP addresses of devices within the organization will share the common prefix. Only x leading prefix bits are considered by routers outside the organization's network. This considerably reduces the size of the forwarding table in these routers, since a single entry of the form a.b.c.d/x will be sufficient to forward packets to any destination within the organization.

In **classful addressing,** the network portions of an IP address were constrained to be 8, 16, or 24 bits in length. Subnets with 8-, 16-, and 24-bit subnet addresses were known as class A, B, and C networks. Class B networks can only accommodate 254 hosts, too small for many organizations, whereas class C networks can accommodate 65,634 hosts, too large for an organization. This leads to poor utilization of the assigned address space.

The IP 255.255.255.255 is known **broadcast address**. When a host sends a datagram with destination address 255.255.255.255, the message is delivered to all hosts on the same subnet.

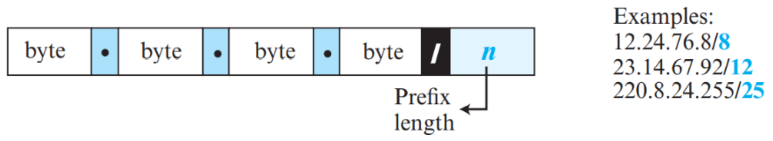
Classless Interdomain Routing (CIDR)

In 1996, the Internet authorities announced a new architecture called classless addressing. In classless addressing, variable-length blocks are used that belong to no classes. In classless addressing, the whole address space is divided into variable length blocks. The prefix in an address defines the block (network id); the suffix defines the node (device id). There can be a block of 2^0, 2^1, 2^2, . . . , 2^{32} addresses. The number of addresses in a block needs to be a power of 2.

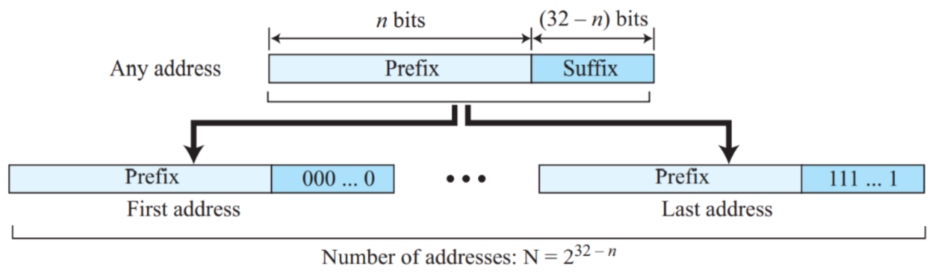


Address space

Unlike classful addressing, the prefix length in classless addressing is variable. Prefix length ranges from 0 to 32. The size of the network is inversely proportional to the length of the prefix. A small prefix means a larger network; a large prefix means a smaller network.



Since the prefix length is not inherent in the classless address, prefix length needs to be specified explicitly. The prefix length, n, is added to the address, separated by a slash. The notation is informally referred to as slash notation and formally as **classless interdomain routing** or **CIDR** strategy.



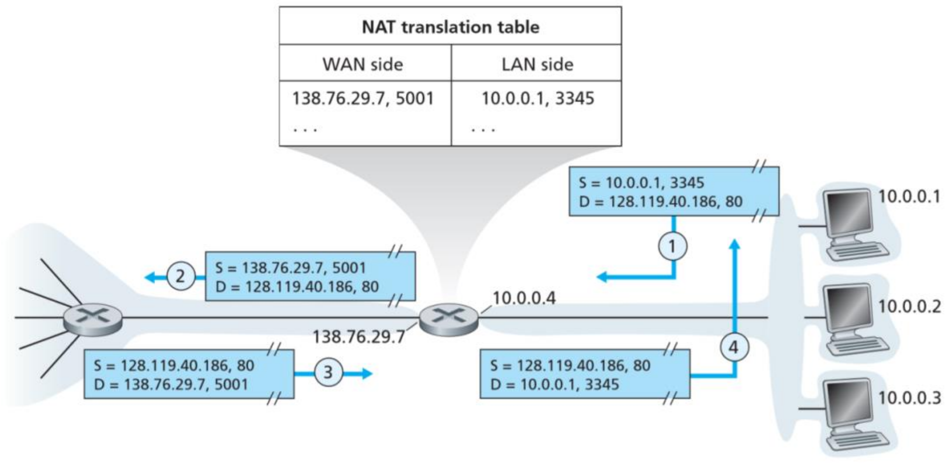
The number of addresses in the block is found as N = 2^{32-n}. To find the first address, we keep the n leftmost bits and set the (32 − n) rightmost bits all to 0s. To find the last address, we keep the n leftmost bits and set the (32 − n) rightmost bits all to 1s.

IP addresses are managed under the authority of the **Internet Corporation for Assigned Names and Numbers (ICANN)** based on guidelines set forth in RFC 2050. The role of the non-profit ICANN organization is not only to allocate IP addresses, but also to manage the DNS root servers. It also has the very contentious job of assigning domain names and resolving domain name disputes. The ICANN allocates addresses to regional Internet and handle the allocation/management of addresses within their regions.

ISP's block	200.23.16.0/20	11001000 00010111 00010000 00000000
Organization 0	200.23.16.0/23	11001000 00010111 00010000 00000000
Organization 1	200.23.18.0/23	11001000 00010111 00010010 00000000
Organization 2	200.23.20.0/23	11001000 00010111 00010100 00000000
. . .	. . .	. . .
Organization 7	200.23.30.0/23	11001000 00010111 00011110 00000000

In order to obtain a block of IP addresses for use within an organization's subnet, a network administrator might first contact its ISP, which would provide addresses from a larger block of addresses that had already been allocated to the ISP. For example, the ISP may itself have been allocated the address block 200.23.16.0/20. The ISP, in turn, could divide its address block into eight equal-sized contiguous address blocks and give one of these address blocks out to each of up to eight organizations that are supported by this ISP.

Network address translation (NAT) is an approach to address allocation. The NAT-enabled router, residing in the home, has an interface that is part of the home network. The NAT-enabled router does not look like a router to the outside world. Instead the NAT router behaves to the outside world as a single device with a single IP address. In essence, the NAT-enabled router is hiding the details of the home network from the outside world. A **NAT translation table** is maintained at the NAT router, and it includes port numbers as well as IP addresses in the table entries.



A user sitting in a home network behind host 10.0.0.1 requests a Web page on some Web server (port 80) with IP address 128.119.40.186. The host 10.0.0.1 assigns the (arbitrary) source port number 3345 and sends the datagram into the LAN. The NAT router receives the datagram, generates a new source port number 5001 for the datagram, replaces the source IP address with its WAN-side IP address 138.76.29.7, and replaces the original source port number 3345 with the new source port number 5001. NAT in the router also adds an entry to its NAT translation table. The Web server responds with a datagram whose destination address is the IP address of the NAT router, and whose destination port number is 5001. When this datagram arrives at the NAT router, the router indexes the NAT translation table using the destination IP address and destination port number to obtain the appropriate IP address (10.0.0.1) and destination port number (3345) for the browser in the home network. The router then rewrites the datagram's destination address and destination port number, and forwards the datagram into the home network.

Introduction to IPv6

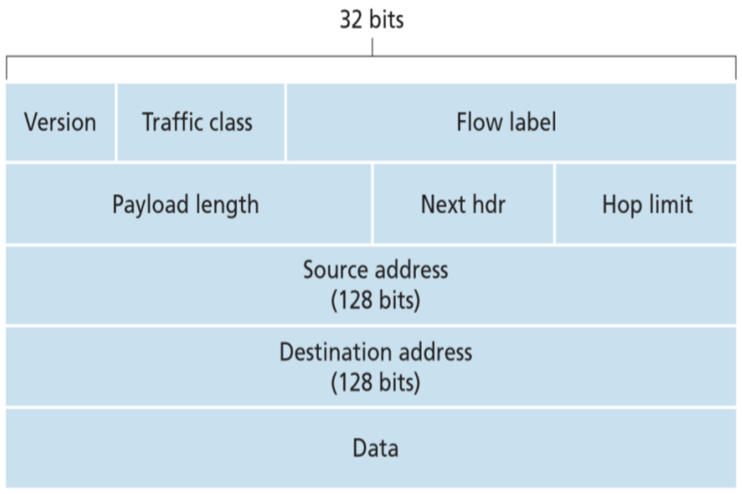
IPv6 was developed to respond to the need for a large IP address space. The most important changes introduced in IPv6 are

- ***Expanded addressing capabilities***: IPv6 increases the size of the IP address from 32 to 128 bits. In addition to unicast and multicast addresses, IPv6 has introduced a new type of address, called an **anycast address**, which allows a datagram to be delivered to any one of a group of hosts.
- ***A streamlined 40-byte header***: A number of IPv4 fields have been dropped or made optional. The resulting 40-byte fixed-length header allows for faster processing of the IP datagram.
- ***Flow labelling and priority***: IPv6 has an elusive definition of a flow. This allows labelling of packets belonging to particular flows for which the sender requests special handling. IPv6 foresee the eventual need to be able to differentiate among the flows, even if the exact meaning of a flow has not yet been determined. The IPv6 header also has an 8-bit traffic class field. This field can be used to give priority to certain datagrams within Traffic class. a flow, or it can be used to give priority to datagrams from certain applications.

The IPv6 datagram format is given below. The IPv6 header doesn't have many fields in it, when compared with IPv4 packet. The IPv6 header is 40 bytes long where as IPv4 header is minimum 20 bytes long.

- ***Version***: This 4-bit field identifies the IP version number.
- ***Traffic class***: This 8-bit field is similar in spirit to the TOS field we saw in IPv4.
- ***Flow label***: 20-bit field is used to identify a flow of datagrams.
- ***Payload length***: This 16-bit value is treated as an unsigned integer giving the number of bytes in the IPv6 datagram following the fixed-length, 40-byte datagram header.
- ***Next header***: This field identifies the protocol to which the contents of this datagram will be delivered.
- ***Hop limit***: The contents of this field are decremented by one by each router that forwards the datagram. If the hop limit count reaches zero, the datagram is discarded.
- ***Source and destination addresses***: These fields contain the source and destination address.

- ***Data***: This is the payload portion of the IPv6 datagram. When the datagram reaches its destination, the payload will be removed from the IP datagram and passed on to the protocol specified in the next header field.



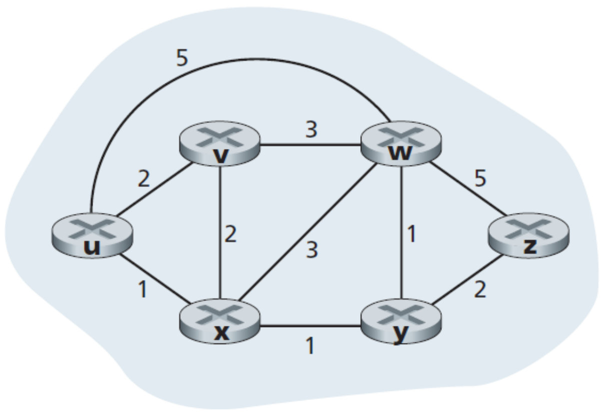
IPv6 does not allow for fragmentation and reassembly at intermediate routers; these operations can be performed only by the source and destination. If an IPv6 datagram received by a router is too large to be forwarded over the outgoing link, the router simply drops the datagram and sends a “Packet Too Big” ICMP error message back to the sender. The sender can then resend the data, using a smaller IP datagram size.

Principles of Routing

In order to transfer packets from a sending host to the destination host, the network layer must determine the path or route that the packets are to follow. This is the job of the network layer routing protocol. At the heart of any routing protocol is the algorithm (the "routing algorithm") that determines the path for a packet. The purpose of a routing algorithm is simple: given a set of routers, with links connecting the routers, a routing algorithm finds a "good" path from source to destination.

When a packet arrives to a router, the router indexes a forwarding table and determines the link interface to which the packet is to be directed. Routing algorithms, operating in network routers, exchange and compute the information that is used to configure these forwarding tables.

A host is attached directly to one router, the **default router** for the host (**first-hop router**). Whenever a host sends a packet, the packet is transferred to its default router. We refer to the default router of the source host as the **source router** and the default router of the destination host as the **destination router**.



A graph is used to formulate routing problems. A graph G = (N,E) is a set N of nodes and a collection E of edges, where each edge is a pair of nodes from N. In the context of network-layer routing, the nodes in the graph represent routers and the edges connecting these nodes represent the physical links between these routers. The costs are assigned to the various edges in the graph abstraction; a natural goal of a routing algorithm is to identify the **least-cost paths** between sources and destinations. The least-cost path between source node u and destination node w is (u, x, y, w) with a path cost of 3. Note that if all edges in the graph have the same cost, the least-cost path is also the **shortest path**.

Routing algorithms can be classified as global or decentralized. A **global routing algorithm** computes the least-cost path between a source and destination using complete, global knowledge about the network. This algorithm takes the connectivity between all nodes and all link costs as inputs. The calculation itself can be run at one site or replicated at multiple sites. The key distinguishing feature is that a global algorithm has complete information about connectivity and link costs. Algorithms with global state information are often referred to as link-state (LS) algorithms, since the algorithm must be aware of the cost of each link in the network. In a **decentralized routing algorithm**, the calculation of the least-cost path is carried out in an iterative, distributed manner. No node has complete information about the costs of all network links. Instead, each node begins with only the knowledge of the costs of its own directly attached links. Then, through an iterative process of calculation and exchange of information with its neighbouring nodes, a node gradually calculates the least-cost path to a destination or set of destinations. An example of the decentralized routing algorithm is called a distance-vector (DV) algorithm, because each node maintains a vector of estimates of the costs (distances) to all other nodes in the network.

A second broad way to classify routing algorithms is according to whether they are static or dynamic. In **static routing algorithms**, routes change very slowly over time, often as a result of human intervention. **Dynamic routing algorithms** change the routing paths as the network traffic loads or topology change. A dynamic algorithm can be run either periodically or in direct response to topology or link cost changes.

A third way to classify routing algorithms is according to whether they are load sensitive or load-insensitive. In a **load-sensitive algorithm**, link costs vary dynamically to reflect the current level of congestion in the underlying link. If a high cost is associated with a link that is currently congested, a routing algorithm will tend to choose routes around such a congested link. Today's Internet routing algorithms (such as RIP, OSPF, and BGP) are **load-insensitive**, as a link's cost does not explicitly reflect its current level of congestion.

Link-State Routing

In a link-state algorithm, the network topology and all link costs are known, that is, available as input to the LS algorithm. This is accomplished by having each node broadcast link-state packets to all other nodes in the network, with each link-state packet containing the identities and costs of its attached links. In practice this is often accomplished by a link-state broadcast algorithm.

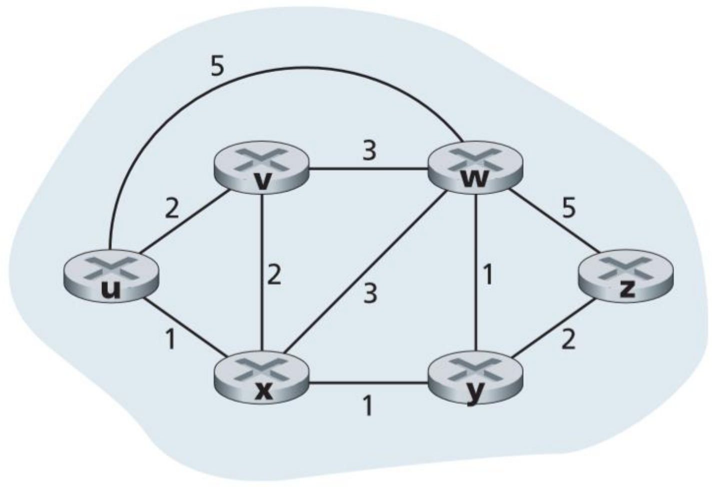
The link-state routing algorithm is also known as Dijkstra's algorithm. It computes the least-cost path from one node (the source, which we will refer to as u) to all other nodes in the network. Dijkstra's algorithm is iterative and has the property that after the k^{th} iteration of the algorithm, the least-cost paths are known to k destination nodes, and among the least-cost paths to all destination nodes, these k paths will have the k smallest costs. Consider the following notation

- D(v): cost of the least-cost path from the source node to destination v as of this iteration of the algorithm.
- p(v): previous node (neighbour of v) along the current least-cost path from the source to v.
- N': subset of nodes; v is in N' if the least-cost path from the source to v is definitively known.

The global routing algorithm consists of an initialization step followed by a loop. The number of times the loop is executed is equal to the number of nodes in the network. Upon termination, the algorithm will have calculated the shortest paths from the source node u to every other node in the network. Given n nodes the LS algorithm has worst-case complexity of order n squared: O(n2).

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v is a neighbor of u
5        then D(v) = c(u,v)
6      else D(v) = ∞
7

8  Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for each neighbor v of w and not in N':
12        D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    least path cost to w plus cost from w to v */
15 until N'= N
```

Now consider the network in following figure and compute the least-cost paths from u to all possible destinations.



- In the initialization step, the currently known least-cost paths from u to its directly attached neighbours, v, x, and w, are initialized to 2, 1, and 5, respectively. The cost to w is set to 5, since this is the cost of the direct link from u to w. The costs to y and z are set to infinity because they are not directly connected to u.
- In the first iteration, look among those nodes not yet added to the set N' and find that node with the least cost as of the end of the previous iteration. That node is x, with a cost of 1, and thus x is added to the set N'. Line 12 of the LS algorithm is then performed to update D(v) for all nodes v, yielding the results shown in the second line in following table. The cost of the path to v is unchanged. The cost of the path to w through node x is found to have a cost of 4. Hence this lower-cost path is selected and w's predecessor along the shortest path from u is set to x. Similarly, the cost to y (through x) is computed to be 2, and the table is updated accordingly.
- In the second iteration, nodes v and y are found to have the least-cost paths (2), and add y to the set N' so that N' now contains u, x, and y. The cost to the

remaining nodes not yet in N', that is, nodes v, w, and z, are updated via line 12 of the LS algorithm, yielding the results shown in the third row in the table and so on.

step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

Distance Vector Routing

It is also called as **Bellman-Ford routing**. The distance vector (DV) algorithm is iterative, asynchronous, and distributed. It is **distributed** in that each node receives some information from one or more of its directly attached neighbours, performs a calculation, and then distributes the results of its calculation back to its neighbours. It is **iterative** in that this process continues on until no more information is exchanged between neighbours. The algorithm is **asynchronous** in that it does not require all of the nodes to operate in lockstep with each other.

The DV algorithm, each node x maintains the following routing information:

- For each neighbour v, the cost c(x,v) from x to directly attached neighbour, v
- Node x's distance vector, that is, $D_x = [D_x(y)\text{: } y \text{ in } N]$, containing x's estimate of its cost to all destinations, y, in N
- The distance vectors of each of its neighbours, that is, $D_v = [D_v(y)\text{: } y \text{ in } N]$ for each neighbour v of x

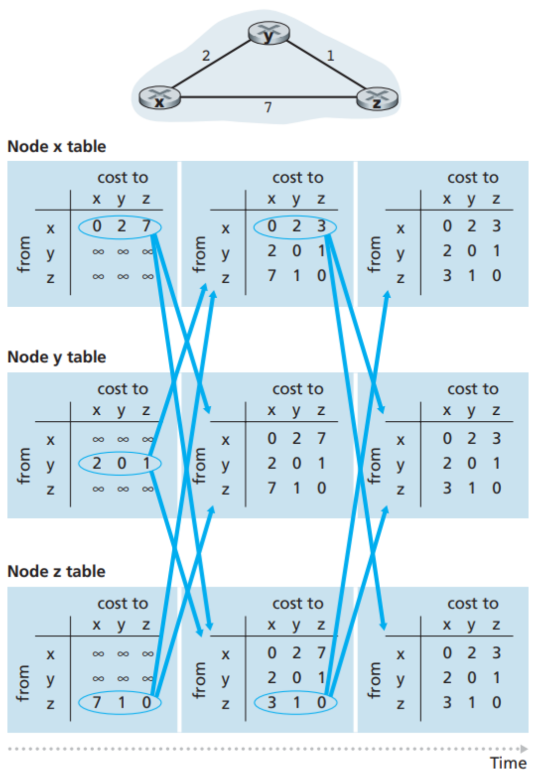
In the distributed, asynchronous algorithm, from time to time, each node sends a copy of its distance vector to each of its neighbours. When a node x receives a new distance vector from any of its neighbours 'v', it saves v's distance vector, and then uses the Bellman-Ford equation to update its own distance vector as follows:

$D_x(y)=\min_v\{c(x,v) + D_v(y)\}$ for each node y in N

If node x's distance vector has changed as a result of this update step, node x will then send its updated distance vector to each of its neighbours, which can in turn update their own distance vectors. As long as all the nodes continue to exchange their distance vectors in an asynchronous fashion, each cost estimate $D_x(y)$ converges to $d_x(y)$, the actual cost of the least-cost path from node x to node y.

```
Initialization:
   for all destinations y in N:
      $D_x(y)$ = c(x,y)   /* if y is not a neighbor then c(x,y) = ∞ */
   for each neighbor w
      $D_w(y)$ = ? for all destinations y in N
   for each neighbor w
      send distance vector $\mathbf{D}_x$ = [$D_x(y)$: y in N] to w

loop
   wait (until I see a link cost change to some neighbor w or
         until I receive a distance vector from some neighbor w)

   for each y in N:
      $D_x(y) = \min_v\{c(x,v) + D_v(y)\}$

   if $D_x(y)$ changed for any destination y
      send distance vector $\mathbf{D}_x$ = [$D_x(y)$: y in N] to all neighbors

forever
```

In the DV algorithm, a node x updates its distance-vector estimate when it either sees a cost change in one of its directly attached links or receives a distance vector update from some neighbour. To update its own forwarding table for a given destination y, what node x really needs to know is not the shortest-path distance to y but instead the neighbouring node v*(y) that is the next-hop router along the shortest path to y.



Node x table

from	cost to x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Node y table

from	cost to x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from	cost to x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Node z table

from	cost to x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from	cost to x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

The leftmost column of the figure displays three initial routing tables for each of the three nodes. For example, the table in the upper-left corner is node x's initial routing table. Within a specific routing table, each row is a distance vector specifically, each node's routing table includes its own distance vector and that of each of its neighbours. Thus, the first row in node x's initial routing table is $D_x = [D_x(x), D_x(y), D_x(z)] = [0, 2, 7]$. The second and third rows in this table are the most recently received distance vectors from nodes y and z, respectively. Because at initialization node x has not received anything from node y or z, the entries in the second and third rows are initialized to infinity.

After initialization, each node sends its distance vector to each of its two neighbours. This is illustrated in the figure by the arrows from the first column of tables to the second column of tables. For example, node x sends its distance vector $D_x = [0, 2, 7]$ to both nodes y and z. After receiving the updates, each node recomputes its own distance vector. For example, node x computes

$Dx(x) = 0$

$Dx(y) = \min\{c(x,y) + Dy(y), c(x,z) + Dz(y)\} = \min\{2 + 0, 7 + 1\} = 2$

$Dx(z) = \min\{c(x,y) + Dy(z), c(x,z) + Dz(z)\} = \min\{2 + 1, 7 + 0\} = 3$

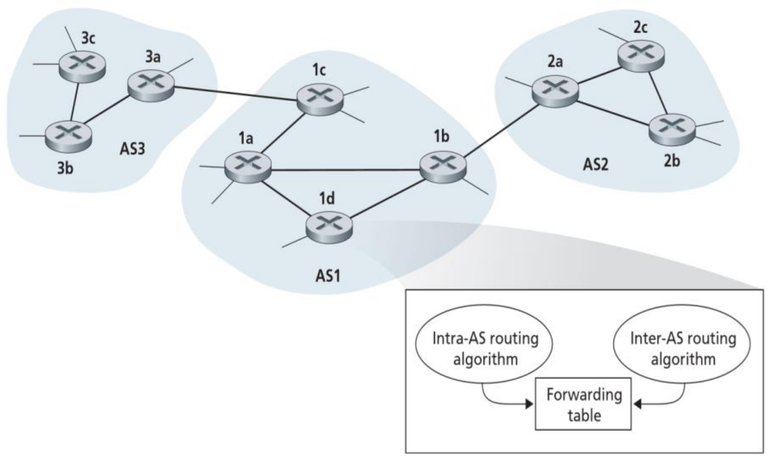
The second column displays, for each node, the node's new distance vector along with distance vectors just received from its neighbours. Node x's estimate for the least cost to node z, Dx(z), has changed from 7 to 3.

After the nodes recompute their distance vectors, they again send their updated distance vectors to their neighbours (if there has been a change). This is illustrated in figure by the arrows from the second column of tables to the third column of tables. Only nodes x and z send updates: node y's distance vector didn't change so node y doesn't send an update. After receiving the updates, the nodes then recompute their distance vectors and update their routing tables, which are shown in the third column.

The process of receiving updated distance vectors from neighbours, recomputing routing table entries, and informing neighbours of changed costs of the least-cost path to a destination continues until no update messages are sent. At this point, since no update messages are sent, no further routing table calculations will occur and the algorithm will enter a quiescent state

The LS algorithm is a global algorithm in the sense that it requires each node to first obtain a complete map of the network before running the Dijkstra algorithm. The DV algorithm is decentralized and does not use such global information. The only information a node will have is the costs of the links to its directly attached neighbours and information it receives from these neighbours. Each node waits for an update from any neighbour, calculates its new distance vector when receiving an update, and distributes its new distance vector to its neighbours.

Routing on the Internet

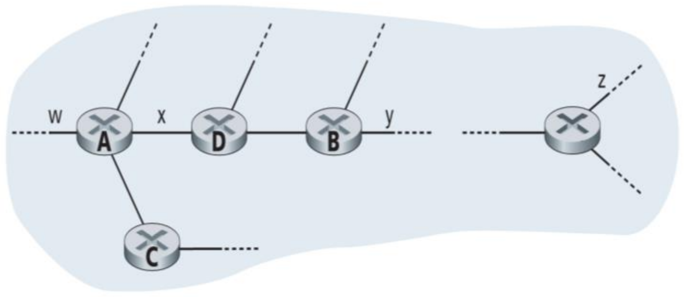


An **autonomous system (AS)** is a collection of routers under the same administrative and technical control, and that all run the same routing protocol among them. The routing algorithm running within an autonomous system is called an **intra-autonomous system routing protocol**. One or more of the routers in an AS will have the added task of being responsible for forwarding packets to destinations outside the AS, to connect ASs to each other; these routers are called **gateway routers**. The **inter-AS routing protocol** involves communication between two ASs, the two communicating ASs must run the same inter-AS routing protocol. Two intra-AS routing protocols (RIP and OSPF) and the inter-AS routing protocol (BGP) that are used in today's Internet.

RIP (Routing Information Protocol)

An **intra-AS routing protocol** is used to determine how routing is performed within an autonomous system (AS). Intra-AS routing protocols are also known as **interior gateway protocols**.

RIP is a distance-vector protocol. In RIP, costs are from source router to a destination subnet. RIP uses the term **hop**, which is the number of subnets traversed along the shortest path from source router to destination subnet, including the destination subnet. In RIP the maximum cost of a path is limited to 15, thus limiting the use of RIP to autonomous systems that are fewer than 15 hops in diameter. In RIP, routing updates are exchanged between neighbours approximately every 30 seconds using a **RIP response message**. The response message sent by a router or host contains a list of up to 25 destination subnets within the AS, as well as the sender's distance to each of those subnets. Response messages are also known as **RIP advertisements**. If a router does not hear from its neighbour at least once every 180 seconds, that neighbour is considered to be no longer reachable. Each router maintains a RIP table known as a **routing table**. A router's routing table includes both the router's distance vector and the router's forwarding table.

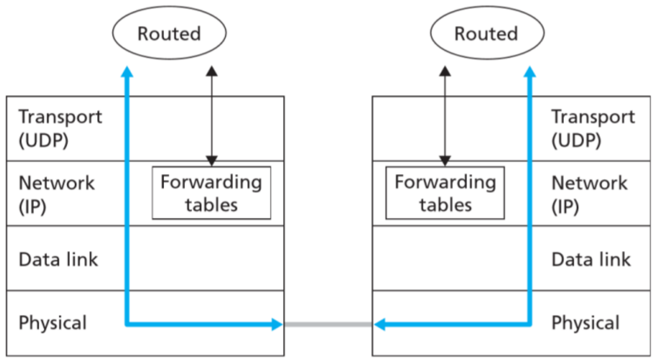


In this figure, lines connecting the routers denote subnets. Only selected routers (A, B, C, and D) and subnets (w, x, y, and z) are labelled. Number of hops from source router A to various subnets is given in the table. The routing table for router D is given below.

Destination Subnet	Next Router	Number of Hops to Destination
w	A	2
y	B	2
z	B	7
x	—	1
. . . .	. . . .	. . . .

The routing table has three columns. The first column is for the destination subnet, the second column indicates the identity of the next router along the shortest path to the destination subnet, and the third column indicates the number of hops (that is, the number of subnets that have to be traversed, including the destination subnet) to get to the destination subnet along the shortest path.

A router can also request information about its neighbour's cost to a given destination using RIP's request message. Routers send RIP request and response messages to each other over UDP using port number 520.



This figure shows how RIP is implemented in a UNIX system. A process called routed executes RIP, that is, maintains routing information and exchanges messages with routed processes running in neighbouring router. Because RIP is implemented as an application-layer process it can send and receive messages over a standard socket and use a standard transport protocol. RIP is implemented as an application-layer protocol running over UDP.

OSPF (Open Shortest Path First)

OSPF routing is used for intra-AS routing in the Internet. OSPF is deployed in upper-tier ISPs whereas RIP is deployed in lower-tier ISPs and enterprise networks. The Open in OSPF indicates that the routing protocol specification is publicly available. OSPF is a link-state protocol that uses flooding of link-state information and a Dijkstra least-cost path algorithm. With OSPF, a router constructs a complete topological map of the entire autonomous system. The router then locally runs Dijkstra's shortest-path algorithm to determine a shortest-path tree to all subnets, with itself as the root node. Individual link costs are configured by the network administrator. The administrator might choose to set all link costs to 1, thus achieving minimum-hop routing, or might choose to set the link weights to be inversely proportional to link capacity in order to discourage traffic from using low-bandwidth links.

In OSPF, a router broadcasts routing information to all other routers in the autonomous system, not just to its neighbouring routers. A router broadcasts link-state information whenever there is a change in a link's state. It also broadcasts a link's state periodically, even if the link's state has not changed. OSPF advertisements are contained in OSPF messages that are carried directly by IP, with an upper-layer protocol of 89 for OSPF. Thus, the OSPF protocol must itself implement functionality such as reliable message transfer and link-state broadcast. The OSPF protocol also checks that links are operational (via a HELLO) and allows an OSPF router to obtain a neighbouring router's database of network-wide link state. Advanced feature in OSPF are

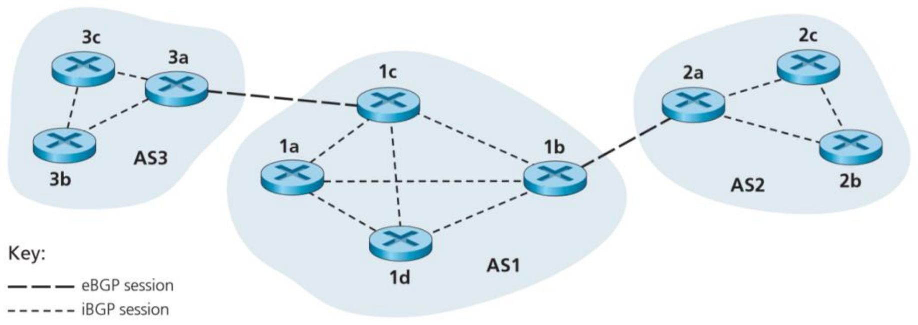
- **Security**: Exchanges between OSPF routers can be authenticated. By default, OSPF packets between routers are not authenticated and could be forged. Two types of authentication can be configured— simple and MD5. In **simple authentication**, the same password is configured on each router. When a router sends an OSPF packet, it includes the password in plaintext. Simple authentication is not very secure. **MD5 authentication** is based on shared secret keys that are configured in all the routers. For each OSPF packet that it sends, the router computes the MD5 hash of the content of the OSPF packet appended with the secret key. Then the router includes the resulting hash value in the OSPF packet. The receiving router, using the preconfigured secret key, will compute an MD5 hash of the packet and compare it with the hash value that the packet carries, thus verifying the packet's authenticity.

- **Multiple same-cost paths**: When multiple paths to a destination have the same cost, OSPF allows multiple paths to be used.
- **Integrated support for unicast and multicast routing**: Multicast OSPF (MOSPF) provides extensions to OSPF to provide for multicast routing. MOSPF uses the existing OSPF link database and adds a new type of link-state advertisement to the existing OSPF link-state broadcast mechanism.
- **Support for hierarchy within a single routing domain**: The most significant advance in OSPF is the ability to structure an autonomous system hierarchically, so that hierarchical routing can be implemented.

An OSPF autonomous system can be configured hierarchically into areas. Each area runs its own OSPF link-state routing algorithm, with each router in an area broadcasting its link state to all other routers in that area. Within each area, one or more **area border routers** are responsible for routing packets outside the area. Exactly one OSPF area in the AS is configured to be the **backbone area**. The primary role of the backbone area is to route traffic between the other areas in the AS. Inter-area routing within the AS requires that the packet be first routed to an area border router, then routed through the backbone to the area border router that is in the destination area, and then routed to the final destination.

BGP (Border Gateway Protocol)

Border Gateway Protocol is the de facto standard inter-AS routing protocol in today's Internet. It is commonly referred to as BGP4. As an inter-AS routing protocol, BGP provides each AS a means to obtain subnet reachability information from neighbouring ASs, propagate the reachability information to all routers internal to the AS, determine "good" routes to subnets based on the reachability information and on AS policy and allows each subnet to advertise its existence to the rest of the Internet.



In BGP, pairs of routers exchange routing information over semi-permanent TCP connections using port 179. There is typically one such BGP TCP connection for each link that directly connects two routers in two different ASs. There is a TCP connection between gateway routers 3a and 1c and another TCP connection between gateway routers 1b and 2a. There are also semi-permanent BGP TCP connections between routers within an AS. For each TCP connection, the two routers at the end of the connection are called **BGP peers**, and the TCP connection along with all the BGP messages sent over the connection is called a **BGP session**. A BGP session that spans two ASs is called an **external BGP (eBGP)** session, and a BGP session between routers in the same AS is called an **internal BGP (iBGP)** session.

BGP allows each AS to learn which destinations are reachable via its neighbouring ASs. In BGP, destinations are not hosts but instead are CIDRized prefixes, with each prefix representing a subnet or a collection of subnets.

Using the eBGP session between the gateway routers 3a and 1c, AS3 sends AS1 the list of prefixes that are reachable from AS3; and AS1 sends AS3 the list of prefixes that are reachable from AS1. Similarly, AS1 and AS2 exchange prefix reachability information through their gateway routers 1b and 2a. When a gateway router receives eBGP-learned prefixes, the gateway router uses its iBGP sessions to distribute the prefixes to the other routers in the AS. Thus, all the routers in AS1 learn about AS3 prefixes, including the gateway router 1b. When a router (gateway or not) learns about a new prefix, it creates an entry for the prefix in its forwarding table.

In BGP, an autonomous system is identified by its globally unique **autonomous system number (ASN)**. When a router advertises a prefix across a BGP session, it includes with the prefix a number of **BGP attributes**. A prefix along with its attributes is called a **route**. Two of the more important attributes are AS-PATH and NEXT-HOP. **AS-PATH** attribute contains the ASs through which the advertisement for the prefix has passed. When a prefix is passed into an AS, the AS adds its ASN to the AS-PATH attribute. Routers also use the AS-PATH attribute in choosing among multiple paths to the same prefix. The **NEXT-HOP** is the router interface that begins the AS-PATH. When the gateway router 3a in AS3 advertises a route to gateway router 1c in AS1 using eBGP, it includes the NEXT-HOP, which is the IP address of the router 3a interface that leads to 1c. The NEXT-HOP attribute is used by routers to properly configure their forwarding tables.

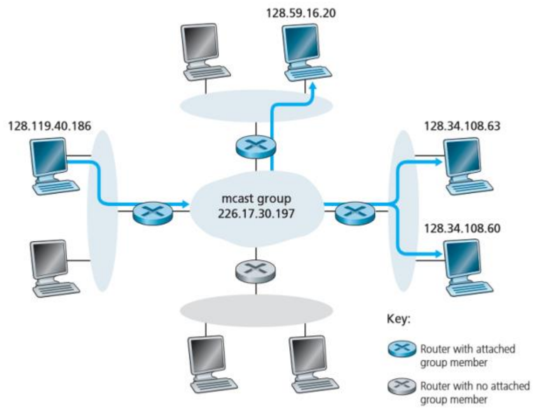
BGP uses eBGP and iBGP to distribute routes to all the routers within ASs. If there is multiple routes, the router must select one of the possible routes. The input into this route selection process is the set of all routes that have been learned and accepted by the router. If there are two or more routes to the same prefix, then BGP sequentially invokes the following elimination rules until one route remains:

- Routes are assigned a local preference value as one of their attributes. The local preference of a route could have been set by the router or could have been learned by another router in the same AS. *The routes with the highest local preference values are selected.*
- From the remaining routes, the route with the *shortest AS-PATH is selected*.
- From the remaining routes, the route with the *closest NEXT-HOP router* is selected.
- If more than one route still remains, the router uses *BGP identifiers* to select the route.

Multicast Routing

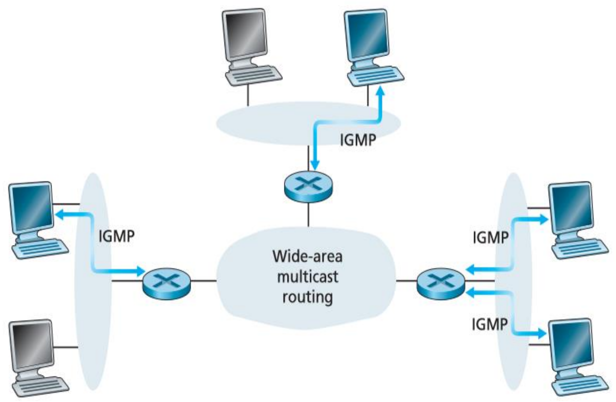
In **multicast** service, a multicast packet is delivered to only a subset of network nodes. Applications that need bulk data transfer, streaming continuous media, shared data applications, data feeds, Web cache updating, and interactive gaming makes use of multicast routing.

In the Internet architecture, a multicast packet is addressed using **address indirection**.



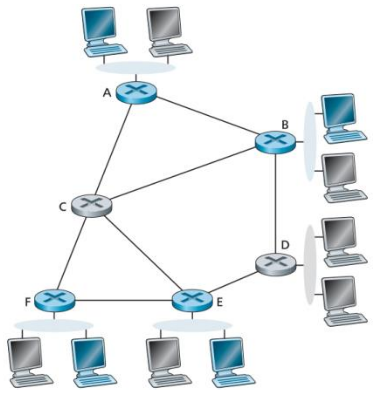
That is, a single identifier is used for the group of receivers, and a copy of the packet that is addressed to the group using this single identifier is delivered to all of the multicast receivers associated with that group. The single identifier that represents a group of receivers is a **class D multicast IP address**. The group of receivers associated with a class D address is referred to as a **multicast group**. In the figure, four hosts are associated with the multicast group address of 226.17.30.197 and will receive all datagrams addressed to that multicast address.

Internet Group Management Protocol



The IGMP3 operates between a host and its directly attached router. The figure has three first-hop multicast routers, each connected to its attached hosts via one outgoing local interface. This local interface is attached to a LAN, and while each LAN has multiple attached hosts, at most a few of these hosts will typically belong to a given multicast group at any given time. **IGMP** allows a host to inform its attached router that an application running on the host wants to join a specific multicast group. **Multicast routing protocols** coordinates the multicast routers throughout the Internet, so that multicast datagrams are routed to their final destinations. Network-layer multicast in the Internet thus consists of two complementary components: IGMP and multicast routing protocols.

IGMP messages are carried within an IP datagram, with an IP protocol number of 2. IGMP has only three message types. The **membership_query message** is sent by a router to all hosts on an attached interface to determine the set of all multicast groups that have been joined by the hosts on that interface. Hosts respond to a membership_query message with an IGMP **membership_report message**. Membership_report messages can also be generated by a host when an application first joins a multicast group without waiting for a membership_query message from the router. The final type of IGMP message is the **leave_group message**, which is optional. The router infers that a host is no longer in the multicast group if it no longer responds to a membership_query message with the given group address.



A multicast routing problem is in this figure. Only a subset of routers in the network actually needs to receive the multicast traffic. In this example only routers A, B, E, and F need to receive the multicast traffic. The goal of multicast routing is to find a tree of links that connects all of the routers that have attached hosts belonging to the multicast group. Multicast packets will then be routed along this tree from the sender to all of the hosts belonging to the multicast tree. Two approaches have been adopted for determining the multicast routing tree

- **Multicast routing using a group-shared tree:** Multicast routing over a group-shared tree is based on building a tree that includes all edge routers with attached hosts belonging to the multicast group. A center-based approach is used to construct the multicast routing tree, with edge routers with attached hosts belonging to the multicast group sending join messages addressed to the center node. All routers along the path that the join message follows will then forward received multicast packets to the edge router that initiated the multicast join.

- **Multicast routing using a source-based tree:** This approach constructs a multicast routing tree for each source in the multicast group. In practice, an RPF algorithm is used to construct a multicast forwarding tree for multicast datagrams originating at source x. A multicast router that receives multicast packets and has no attached hosts joined to that group will send a prune message to its upstream router. If a router receives prune messages from each of its downstream routers, then it can forward a prune message upstream.

The first multicast routing protocol used in the Internet was the **Distance-Vector Multicast Routing Protocol (DVMRP)**. DVMRP implements source-based trees with reverse path forwarding and pruning. DVMRP uses an RPF(Reverse Path Forwarding) algorithm with pruning. The most widely used Internet multicast routing protocol is the **Protocol-Independent Multicast (PIM) routing protocol**. It has two modes. In **dense mode**, multicast group members are densely located; that is, many or most of the routers in the area need to be involved in routing multicast datagrams. In **sparse mode**, the number of routers with attached group members is small with respect to the total number of routers; group members are widely dispersed. In **source-specific multicast (SSM)**, only a single sender is allowed to send traffic into the multicast tree.