Module II-Network Layer

Network Layer Services

Services Provided to the Transport Layer

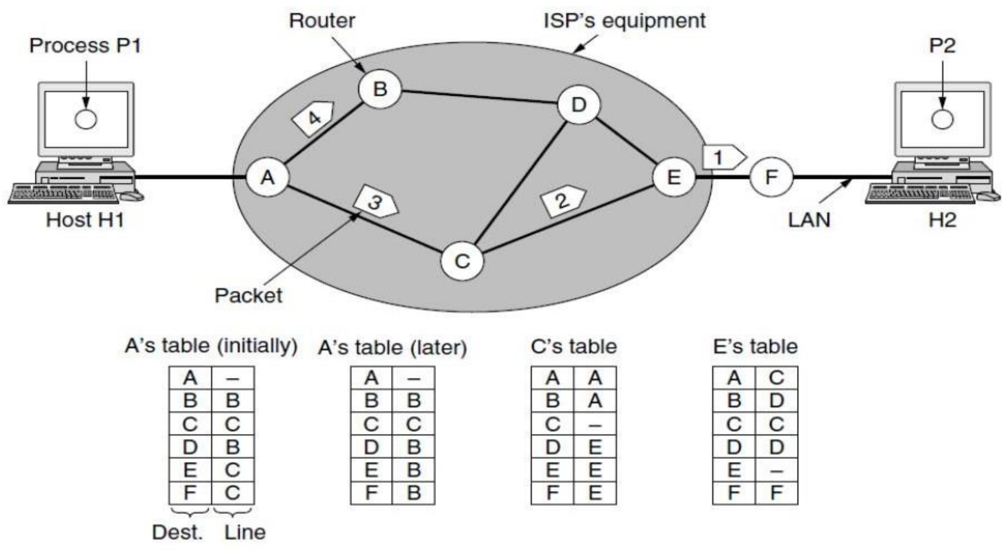
The network layer provides services to the transport layer at the network layer/transport layer interface. The services need to be carefully designed with the following goals in mind:

1. The services should be independent of the router technology.
2. The transport layer should be shielded from the number, type, and topology of the routers present.
3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

Implementation of Connectionless Service

Two different organizations are possible, depending on the type of service offered. If connectionless service is offered, packets are injected into the network individually and routed independently of each other. No advance setup is needed. In this context, the packets are frequently called **datagrams** and the network is called a **datagram network**. If connection-oriented service is used, a path from the source router all the way to the destination router must be established before any data packets can be sent. This connection is called a **VC** (**virtual circuit**), in analogy with the physical circuits set up by the telephone system, and the network is called a **virtual-circuit network**.

Let us now see how a datagram network works. Suppose that the process *P1* in Fig. has a long message for *P2*. It hands the message to the transport layer, with instructions to deliver it to process *P2* on host *H2*. The transport layer code runs on *H1*, typically within the operating system. It prepends a transport header to the front of the message and hands the result to the network layer, probably just another procedure within the operating system.



A's table (initially)

A	–
B	B
C	C
D	B
E	C
F	C

A's table (later)

A	–
B	B
C	C
D	B
E	B
F	B

C's table

A	A
B	A
C	–
D	E
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	–
F	F

Routing within a datagram network.

Routing within a datagram network.

Let us assume for this example that the message is four times longer than the maximum packet size, so the network layer has to break it into four packets, 1, 2,3, and 4, and send each of them in turn to router *A* using some point-to-point protocol, for example, PPP. At this point the ISP takes over. Every router has an internal table telling it where to send packets for each of the possible destinations.

Each table entry is a pair consisting of a destination and the outgoing line to use for that destination. Only directly connected lines can be used. For example, in Fig., *A* has only two outgoing lines—to *B* and to *C*—so every incoming packet must be sent to one of these routers, even if the ultimate destination is to some other router. *A*'s initial routing table is shown in the figure under the label ‘‘initially.’’ At *A*, packets 1, 2, and 3 are stored briefly, having arrived on the incoming link and had their checksums verified. Then each packet is forwarded according to *A*'s table, onto the outgoing link to *C* within a new frame. Packet 1 is then forwarded to *E* and then to *F*. When it gets to *F*, it is sent within a frame over the LAN to *H2*. Packets 2 and 3 follow the same route. However, something different happens to packet 4. When it gets to *A* it is sent to router *B*, even though it is also destined for *F*. For some reason, *A* decided to send packet 4 via a different route than that of the first three packets. Perhaps it has learned of a traffic jam somewhere along the *ACE* path and updated its routing table, as shown under the label ‘‘later.’’ The algorithm that manages the tables and makes the routing decisions is called the **routing algorithm.** IP (Internet Protocol), which is the basis for the entire Internet, is the dominant example of a connectionless network service. Each packet carries a destination IP address that routers use to individually forward each packet. The addresses are 32 bits in IPv4 packets and

128 bits in IPv6 packets.

ROUTING ALGORITHMS

The main function of the network layer is routing packets from the source machine to the destination machine. The algorithms that choose the routes and the data structures that they use are a major area of network layer design.

The **routing algorithm** is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. If the network uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time.

A router as having two processes inside it. One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing tables. This process is **forwarding**.

The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play.

Routing algorithms can be grouped into two major classes: non adaptive and adaptive. **Non adaptive algorithms** do not base their routing decisions on any measurements or estimates of the current topology

Instead, the choice of the route to use to get from *I* to *J* (for all *I* and *J*) is computed in advance, offline, and downloaded to the routers when the network is booted. This procedure is sometimes called **static routing**.

Adaptive algorithms, in contrast, change their routing decisions to reflect changes in the topology, and sometimes changes in the traffic as well.

These **dynamic routing** algorithms differ in where they get their information, when they change the routes, and what metric is used for optimization .

The Optimality Principle

Before we get into specific algorithms, it may be helpful to note that one can make a general statement about optimal routes without regard to network topology or traffic. This statement is known as the **optimality principle**. It states that if router *J* is on the optimal path from router *I* to router *K*, then the optimal path from *J* to *K* also falls along the same route.

Shortest Path Algorithm

These paths are the ones that we want a distributed routing algorithm to find, even though not all routers may know all of the details of the network. The idea is to build a graph of the network, with each node of the graph representing a router and each edge of the graph representing a communication line, or link. To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

Flooding

When a routing algorithm is implemented, each router must make decisions based on local knowledge, not the complete picture of the network. A simple local technique is **flooding**, in which every incoming packet is sent out on every outgoing line except the one it arrived on.

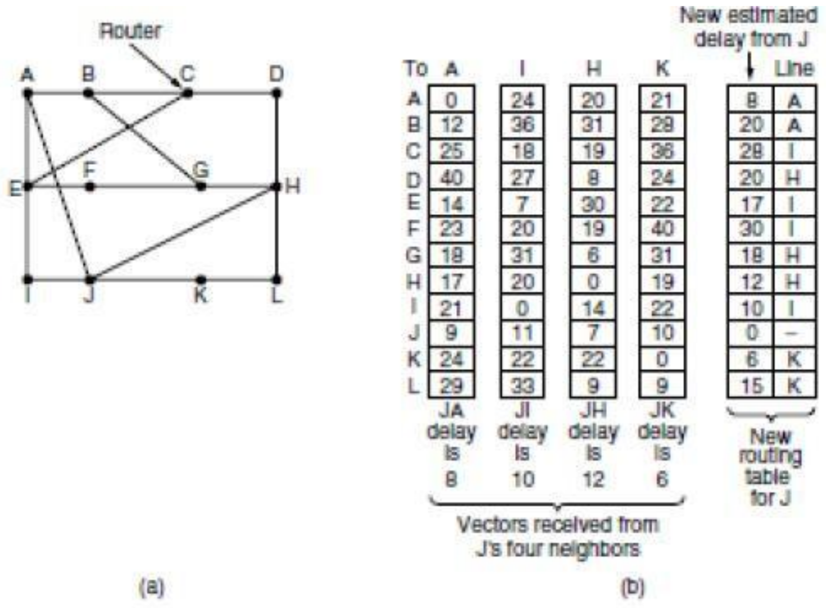
Distance Vector Routing

Computer networks generally use dynamic routing algorithms that are more complex than flooding, but more efficient because they find shortest paths for the current topology.

A **distance vector routing** algorithm operates by having each router maintain a table (i.e., a vector) giving the best known distance to each destination and which link to use to get there. These tables are updated by exchanging information with the neighbors. Eventually, every router knows the best link to reach each destination. The distance vector routing algorithm is sometimes called by other names, most commonly the distributed **Bellman- Ford** routing algorithm,

(a) A network. (b) Input from *A*, *I*, *H*, *K*, and the new routing table for *J*.

Consider how *J* computes its new route to router *G*. It knows that it can get to *A* in 8 m sec, and furthermore *A* claims to be able to get to *G* in 18 m sec, so *J* knows it can count on a delay of 26 m sec to *G* if it forwards packets bound for *G* to *A*. Similarly, it computes the delay to *G* via *I*, *H*, and *K* as 41 (31 + 10), 18 (6 + 12), and 37 (31 + 6) m sec, respectively. The best of these values is 18, so it makes an entry in its routing table that the delay to *G* is 18 m sec and that the route to use is via *H*. The same calculation is performed for all the other destinations, with the new routing table shown in the last column of the figure.



To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K

CONGESTION CONTROL ALGORITHMS

Too many packets present in the network causes packet delay and loss that degrades performance. This situation is called **congestion**. The network and transport layers share the responsibility for handling congestion. Since congestion occurs within the network, it is the network layer that directly experiences it and must ultimately determine what to do with the excess packets. However, the most effective way to control congestion is to reduce the load that the transport layer is placing on the network. This requires the network and transport layers to work together.

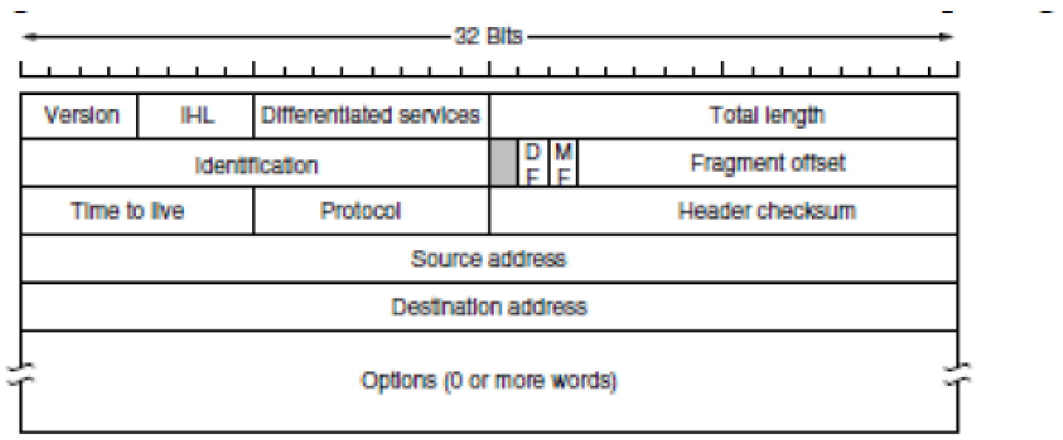
Approaches to Congestion Control

The presence of congestion means that the load is (temporarily) greater than the resources (in a part of the network) can handle. Two solutions come to mind: increase the resources or decrease the load.Timescales of approaches to congestion control.

The most basic way to avoid congestion is to build a network that is well matched to the traffic that it carries.

The IP Version 4 Protocol

An appropriate place to start our study of the network layer in the Internet is with the format of the IP datagrams themselves. An IPv4 datagram consists of a header part and a body or payload part. The header has a 20-byte fixed part and a variable-length optional part. The header format is shown in Fig. 5-46. The bits are transmitted from left to right and top to bottom, with the highorder bit of the *Version* field going first.



The IPv4 (Internet Protocol) header.

The *Version* field keeps track of which version of the protocol the datagram belongs to. Version 4 dominates the Internet today, and that is where we have started our discussion. By including the version at the start of each datagram, it becomes possible to have a transition between versions over a long period of time

Since the header length is not constant, a field in the header, *IHL*, is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15, which limits the header to 60 bytes, and thus the *Options* field to 40 bytes. For some options, such as one that records the route a packet has taken, 40 bytes is far too small, making those options useless. The *Differentiated services* field is one of the few fields that has changed its meaning (slightly) over the years. Originally, it was called the *Type of service* field. It was and still is intended to distinguish between different classes of service.

The *Type of service* field provided 3 bits to signal priority and 3 bits to signal whether a host cared more about delay, throughput, or reliability.

The *Total length* includes everything in the datagram—both header and data. The maximum length is 65,535 bytes. At present, this upper limit is tolerable, but with future networks, larger datagrams may be needed.

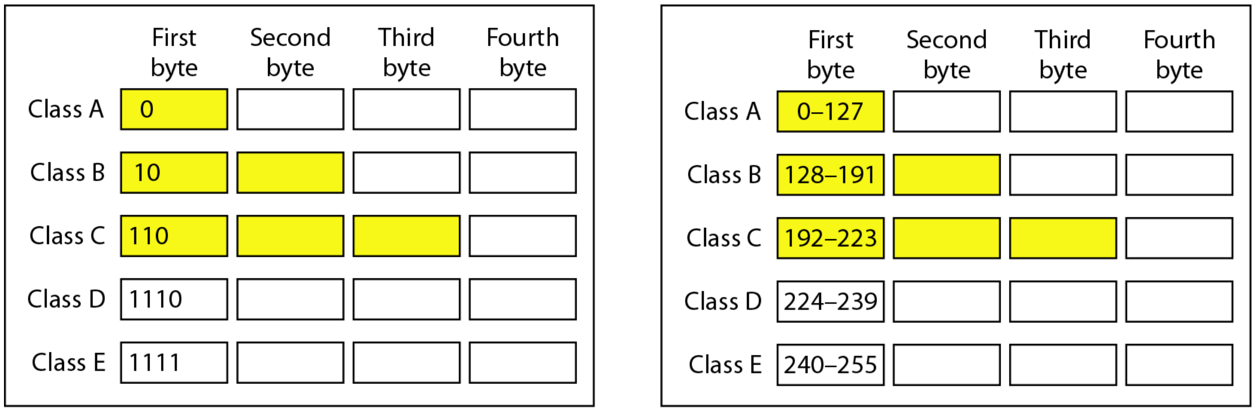
The *Identification* field is needed to allow the destination host to determine which packet a newly arrived fragment belongs to. All the fragments of a packet contain the same *Identification* value.

<u>Classful Addressing</u>

In classful addressing, the address space is divided into five classes:

Class A, B, C, D, and E.

Finding the classes in binary and dotted-decimal notation



	First byte	Second byte	Third byte	Fourth byte
Class A	0			
Class B	10	🟨		
Class C	110	🟨	🟨	
Class D	1110			
Class E	1111			

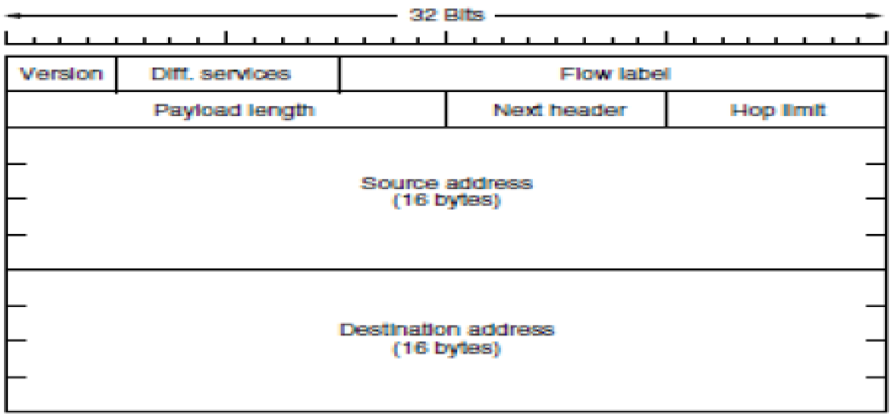
a. Binary notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0–127			
Class B	128–191	🟨		
Class C	192–223	🟨	🟨	
Class D	224–239			
Class E	240–255			

b. Dotted-decimal notation

The Main IPv6 Header

The IPv6 header is shown in Figure The *Version* field is always 6 for IPv6 (and 4 for IPv4). During the transition period from IPv4, which has already taken more than a decade, routers will be able to examine this field to tell what kind of packet they have. For example, the Ethernet *Type* field has different values to indicate an IPv4 or an IPv6 payload.



The IPv6 fixed header (required).

The ***Differentiated services*** field is used to distinguish the class of service for packets with different real-time delivery requirements.

The ***Flow label*** field provides a way for a source and destination to mark groups of packets that have the same requirements and should be treated in the same way by the network, forming a pseudo connection

The *Payload length* field tells how many bytes follow the 40-byte header of Fig. The name was changed from the IPv4 *Total length* field because the meaning was changed slightly: the 40 header bytes are no longer counted as part of the length

The *Next header* field lets the cat out of the bag. The reason the header could be simplified is that there can be additional (optional) extension headers.

If this header is the last IP header, the *Next header* field tells which transport protocol handler to pass the packet to.

The *Hop limit* field is used to keep packets from living forever. It is, the same as the *Time to live* field in IPv4, namely, a field that is decremented on each hop. Next come the *Source address* and *Destination address* fields.

A new notation has been devised for writing 16-byte addresses. They are written as eight groups of four hexadecimal digits with colons between the groups, like this: 8000:0000:0000:0000:0123:4567:89AB: CDEF