Computer Network

Lecture-37

Dharmendra Kumar (Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research,
Prayagraj

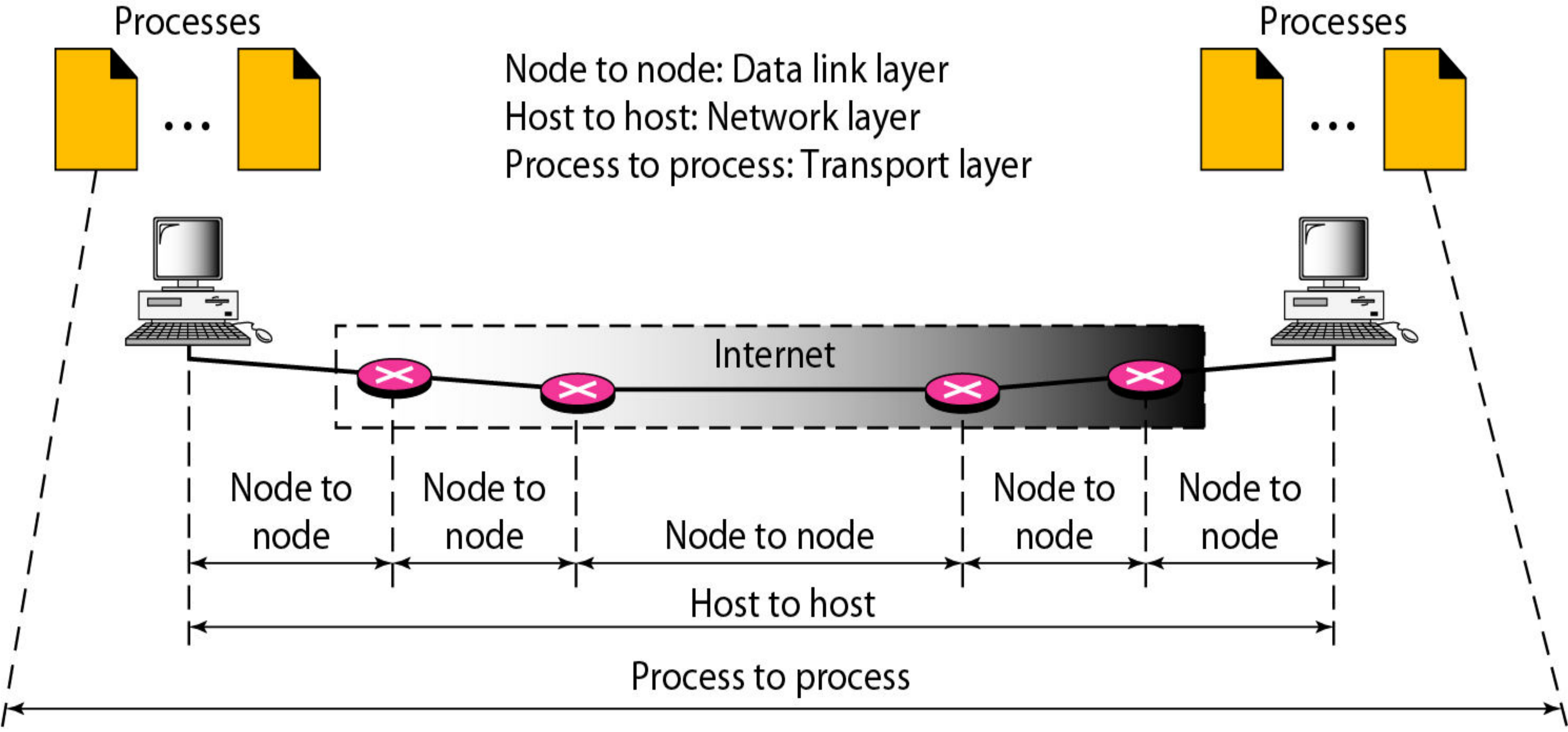
Unit-4

Transport Layer

Process-to-Process Delivery

Process-to-Process Delivery

The transport layer is responsible for process-to process delivery-the delivery of a packet, part of a message, from one process to another.



Process-to-Process Delivery

Client/Server Paradigm

- ❖A process on the local host, called a client, needs services from a process usually on the remote host, called a server.
- ❖Both processes (client and server) have the same name. For example, to get the day and time from a remote machine, we need a Daytime client process running on the local host and a Daytime server process running on a remote machine.

Process-to-Process Delivery

Addressing

- ❖ At the transport layer, we need a transport layer address, called a **port number**, to choose among multiple processes running on the destination host. The destination port number is needed for delivery; the source port number is needed for the reply.
- ❖ In the Internet model, the port numbers are 16-bit integers between 0 and 65,535.

Process-to-Process Delivery

IANA Ranges

The IANA (Internet Assigned Number Authority) has divided the port numbers into three ranges: well known, registered, and dynamic (or private).

Well-known ports: The ports ranging from 0 to 1023 are assigned and controlled by IANA. These are the well-known ports.

Registered ports: The ports ranging from 1024 to 49,151 are not assigned or controlled by IANA. They can only be registered with IANA to prevent duplication.

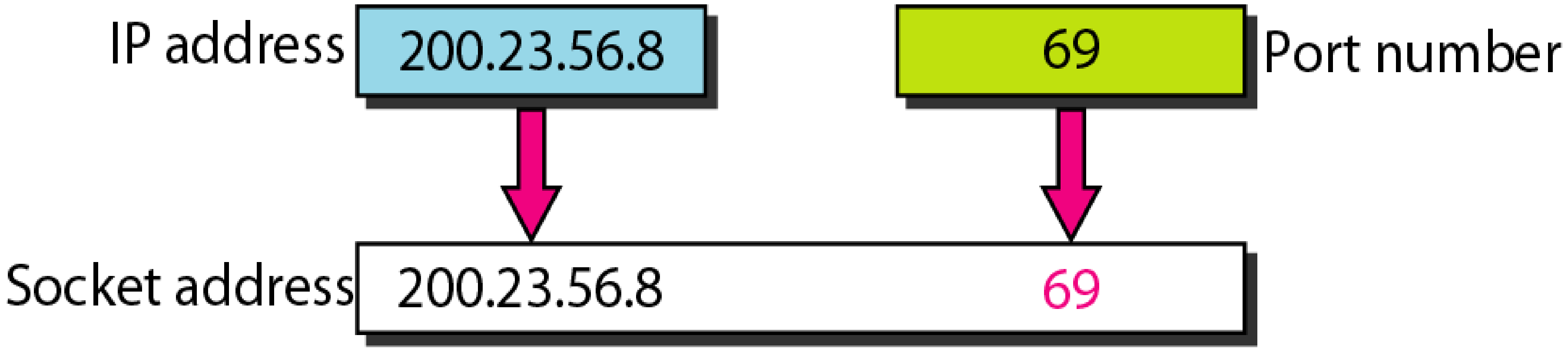
Dynamic ports: The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used by any process. These are the ephemeral ports.

Process-to-Process Delivery

Socket Addresses

Process-to-process delivery needs two identifiers, IP address and the port number, at each end to make a connection. The combination of an IP address and a port number is called a socket address.

The client socket address defines the client process uniquely just as the server socket address defines the server process uniquely.



Process-to-Process Delivery

Connectionless Versus Connection-Oriented Service

A transport layer protocol can either be connectionless or connection-oriented.

Connectionless Service

In a connectionless service, the packets are sent from one party to another with no need for connection establishment or connection release. The packets are not numbered; they may be delayed or lost or may arrive out of sequence. There is no acknowledgment.

Connection-Oriented Service

In a connection-oriented service, a connection is first established between the sender and the receiver. Data are transferred. At the end, the connection is released.

Process-to-Process Delivery

Reliable Versus Unreliable

- ❖ The transport layer service can be reliable or unreliable. If the application layer program needs reliability, we use a reliable transport layer protocol by implementing flow and error control at the transport layer.
- ❖ On the other hand, if the application program does not need reliability because it uses its own flow and error control mechanism or it needs fast service or the nature of the service does not demand flow and error control (real-time applications), then an unreliable protocol can be used.

Process-to-Process Delivery

- In the Internet, there are three different transport layer protocols, UDP, TCP and SCTP.
- UDP is connectionless and unreliable;
- TCP and SCTP are connection-oriented and reliable.

Process-to-Process Delivery

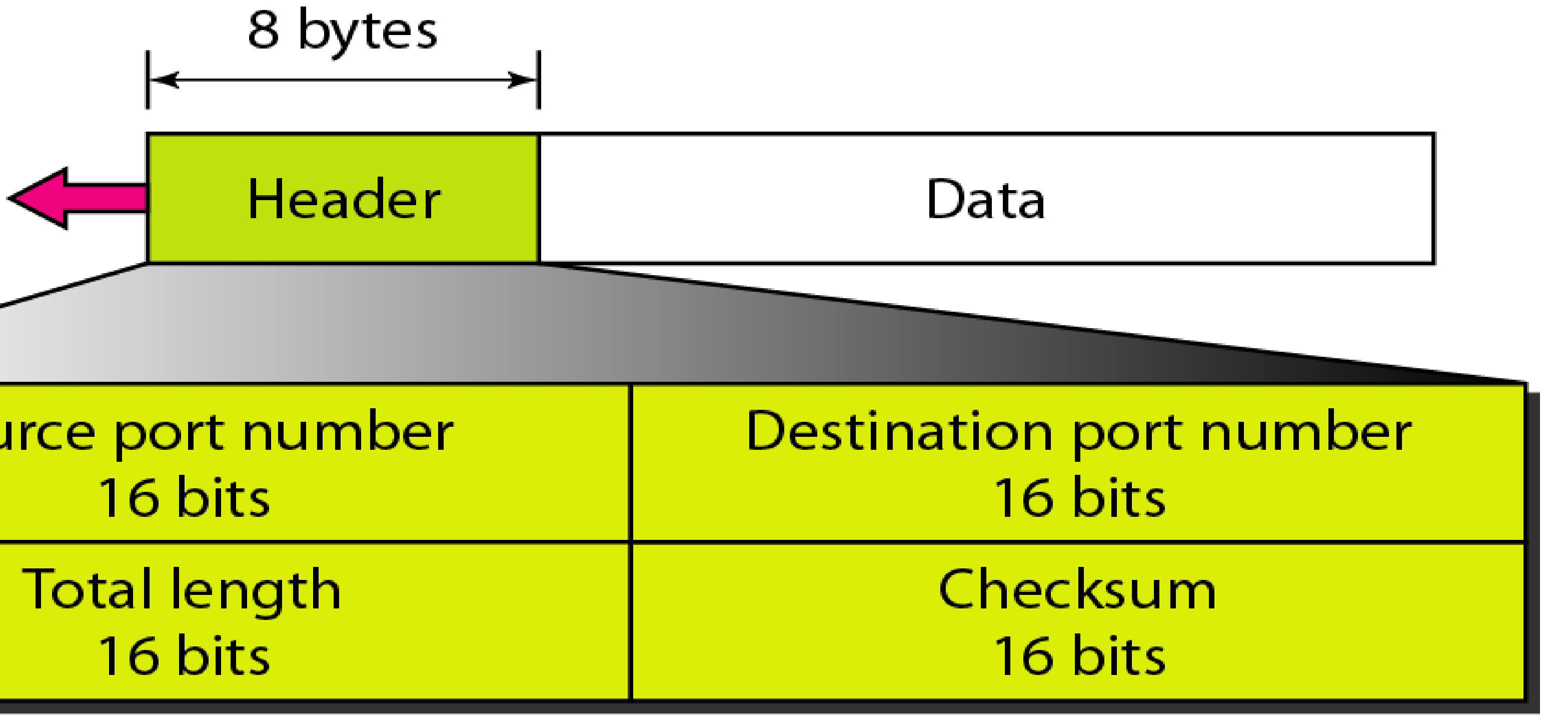
USER DATAGRAM PROTOCOL (UDP)

- The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication. Also, it performs very limited error checking.
- UDP is a very simple protocol using a minimum of overhead. If a process wants to send a small message and does not care much about reliability, it can use UDP. Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP or SCTP.

Process-to-Process Delivery

User Datagram

UDP packets, called user datagrams, have a fixed-size header of 8 bytes. Following figure shows the format of a user datagram.



Process-to-Process Delivery

All the fields are explained as following:-

Source port number:

This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535. If the source host is the client (a client sending a request), the port number, in most cases, is an **ephemeral port number** requested by the process and chosen by the UDP software running on the source host. If the source host is the server (a server sending a response), the port number, in most cases, is a **well-known port number**.

Process-to-Process Delivery

Destination port number: This is the port number used by the process running on the destination host. It is also 16 bits long. If the destination host is the server (a client sending a request), the port number, in most cases, is a well-known port number. If the destination host is the client (a server sending a response), the port number, in most cases, is an ephemeral port number. In this case, the server copies the ephemeral port number it has received in the request packet.

Total length: This is a 16-bit field that defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes.

UDP datagram length = IP datagram length – IP header length

Checksum: This field is used to detect errors over the entire user datagram (header plus data).

Process-to-Process Delivery

UDP Operation

Connectionless Services

- ❖ UDP provides a connectionless service. This means that each user datagram sent by UDP is an independent datagram.
- ❖ There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program.
- ❖ The user datagrams are not numbered.
- ❖ Also, there is no connection establishment and no connection termination, as is the case for TCP. This means that each user datagram can travel on a different path.
- ❖ Only those processes sending short messages should use UDP.

Process-to-Process Delivery

Flow and Error Control

- UDP is a very simple, unreliable transport protocol. There is no flow control and hence no window mechanism. The receiver may overflow with incoming messages.
- There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated. When the receiver detects an error through the checksum, the user datagram is silently discarded.

Process-to-Process Delivery

Use of UDP

The following are some uses of the UDP protocol:

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.
- UDP is suitable for a process with internal flow and error control mechanisms. For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP.
- UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.
- UDP is used for management processes such as SNMP.
- UDP is used for some route updating protocols such as Routing Information Protocol(RIP).