Lecture 3.1

Internet protocol – IPv4

Dr. Vandana Kushwaha

Department of Computer Science

Institute of Science, BHU, Varanasi

Internet Protocol version 4(IPv4)

The Internet Protocol version4 (**IPv4**) is a **Network Layer protocol** in **TCP/IP** protocols suit.

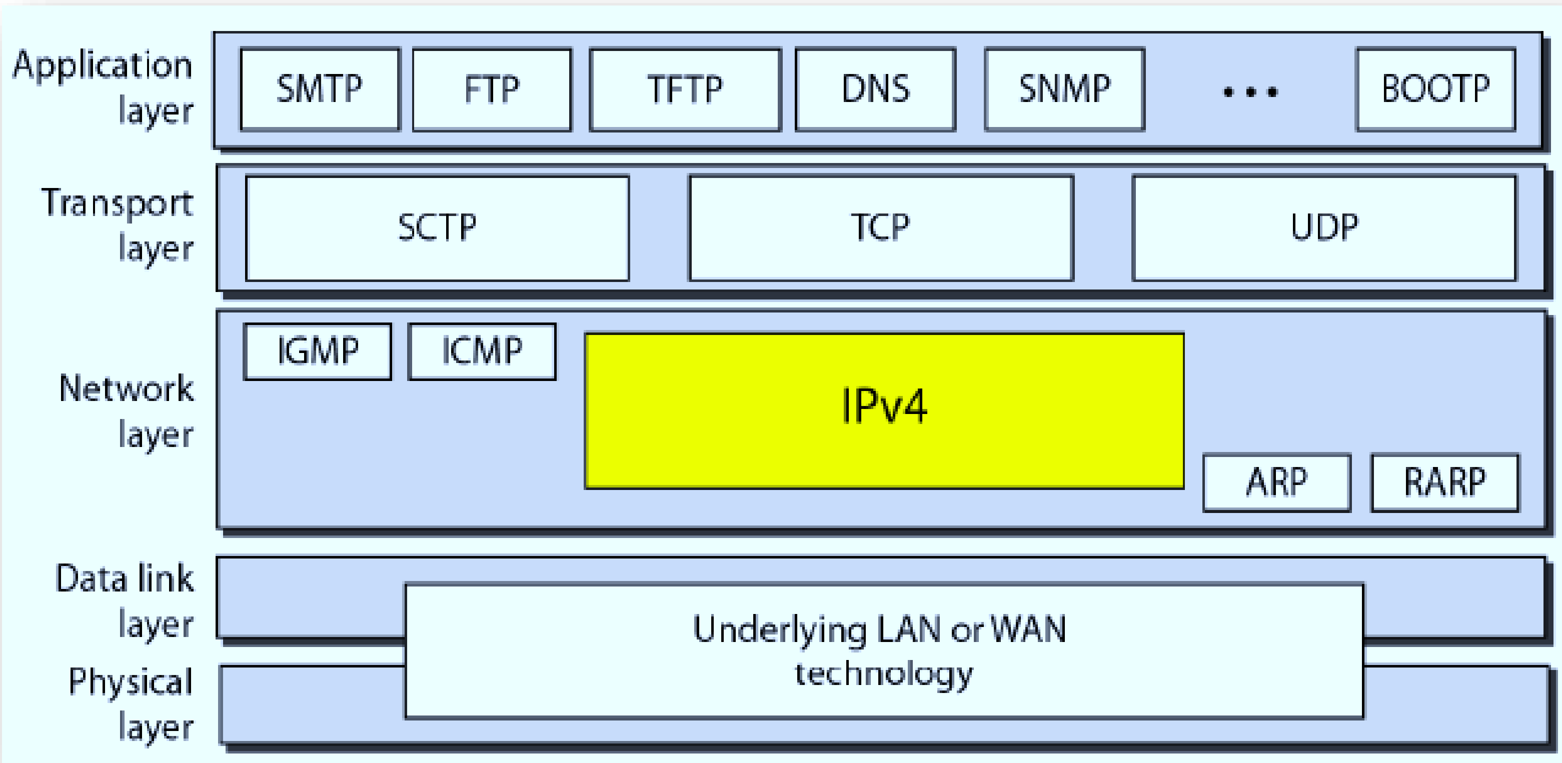
Characteristics of IPv4 protocol

- **IPv4** is an **Unreliable** and **Connectionless Datagram protocol- a best-effort delivery service**.
- The term ***best-effort*** **means** that **IPv4** provides **no error control or flow control** (except for error detection on the header).
- **IPv4 does its best** to get a transmission through to its **destination,** but with **no guarantees**.
- If **reliability is important**, **IPv4** must be **paired with** a **reliable protocol at Transport Layer** such as **TCP**.

Internet Protocol version 4(IPv4)

- **IPv4** is also a **connectionless packet-switching** network that uses the **datagram approach**.
- This means that **each datagram** is **handled independently**, and each **datagram can follow** a **different route** to the **destination**.
- This implies that **datagrams** sent by the **same source** to the **same destination** could **arrive out of order**.
- Also, some could be **lost or corrupted** during **transmission.**
- **IPv4** relies on a **higher-level protocol** like **TCP** to take care of all these problems.

Position of IPv4 in TCP/IP protocol suite



IPv4 Datagram

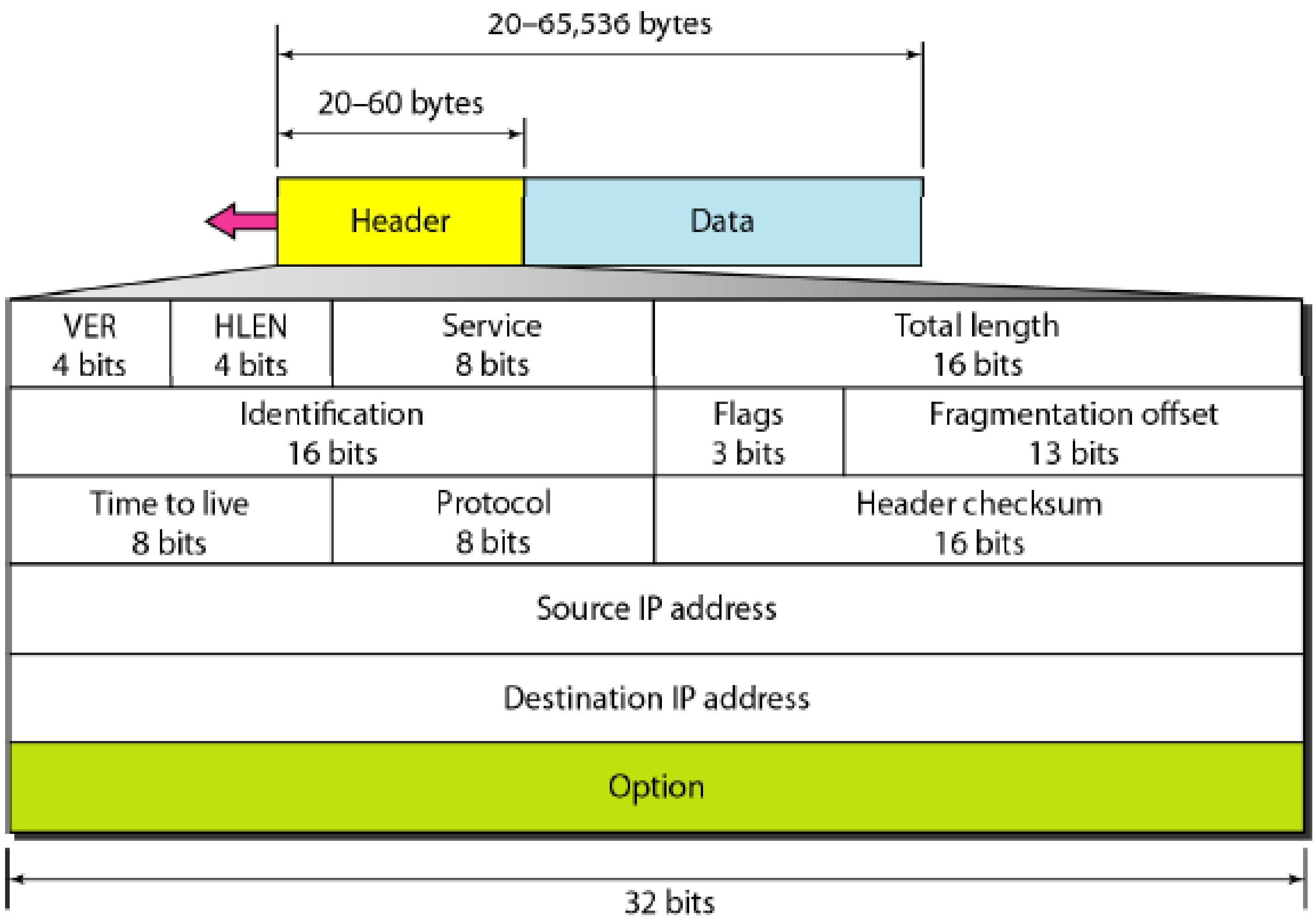
- **Packets** in the **IPv4 layer** are called **datagrams**.
- A **datagram** is a **variable-length** packet consisting of **two parts**: **header** and **data**.
- The **header** is **20 to 60 bytes** in **length** and contains information essential to **routing** and **delivery**.
- It is customary in ***TCP/IP*** to show the header in **4-byte sections**.

Fields in IPv4 Header

a. Version (VER)

- This **4-bit** field defines the **version** of the **IPv4 protocol**.
- Currently the **version** is **4**. However, **version 6** (or **IPv6**) may totally replace **version 4** in the future.

IPv4 Datagram



IPv4 Datagram

- **b. Header length (HLEN)**
- This **4-bit** field defines the **total length** of the **datagram header** in **4-byte** **words**.
- This **field** is needed because the **length** of the **header** is **variable** (between **20** and **60 bytes**).
- When there are **no options**, the header length is **20 bytes**, and the value of this field is **5** (**5 x 4 = 20**).
- When the **option field** is at its **maximum size**, the **value** of this field is **15** (**15 x 4 = 60**).

IPv4 Datagram

c. Services(8 bits): IETF has changed the interpretation and name of this **8-bit field**. This field, previously called **Service type**, is now called **Differentiated services**.

1. Service Type

- In this interpretation, the **first 3 bits** are called **precedence bits**. The **next 4 bits** are called **type of service (TOS) bits**, and the **last bit** is **not used**.
- **Precedence** is a **3-bit** subfield ranging from **0** (**000** in binary) to **7** (**111** in binary).
- The **precedence** defines the **priority** of the **datagram** in issues such as **congestion.**
- If a **router** is congested and needs to **discard** some **datagrams**, those **datagrams** with **lowest precedence** are **discarded first**.
- Some **datagrams** in the Internet are **more important** than others. For **example**, a datagram used for **network management** is much more **urgent and important.**

IPv4 Datagram

- **TOS bits** is a **4-bit** subfield with **each bit** having a **special meaning**.
- Although a **bit** can be either **0 or 1**, one and only one of the **bits** can have the value of **1** in each datagram.
- The **bit patterns** and their interpretations are given in Table below. With only **1 bit** set at a time, we can have **five different types of services**.
- Application programs can request a specific **type of service**. The defaults for some applications are shown in **Table.**

TOS Bits	*Description*
0000	Normal (default)
0001	Minimize cost
0010	Maximize reliability
0100	Maximize throughput
1000	Minimize delay

Default types of service

Protocol	*TOS Bits*	*Description*
ICMP	0000	Normal
BOOTP	0000	Normal
NNTP	0001	Minimize cost
IGP	0010	Maximize reliability
SNMP	0010	Maximize reliability
TELNET	1000	Minimize delay
FTP (data)	0100	Maximize throughput
FTP (control)	1000	Minimize delay
TFTP	1000	Minimize delay
SMTP (command)	1000	Minimize delay
SMTP (data)	0100	Maximize throughput
DNS (UDP query)	1000	Minimize delay
DNS (TCP query)	0000	Normal
DNS (zone)	0100	Maximize throughput

Default types of service

- **Interactive activities**, activities requiring immediate attention, and activities requiring immediate response need **minimum delay**. e.g. **TELNET.**
- Those activities that **send bulk data** require **maximum throughput**. E.g. **FTP(data)**
- **Management activities** need **maximum reliability**. e.g. **SNMP.**
- **Background activities** need **minimum cost**. e.g. **NNTP.**

Services: Differentiated Services

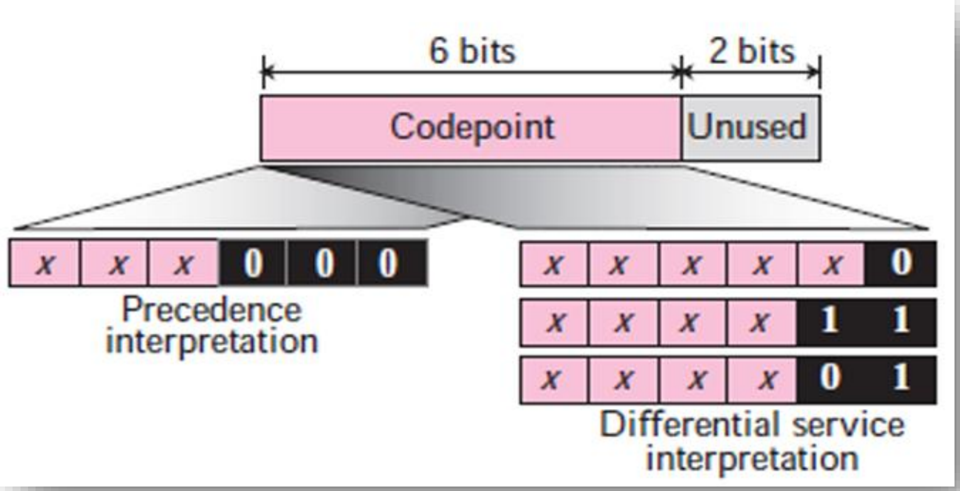
- In this interpretation, the **first 6 bits** make up the **codepoint subfield**, and the **last 2 bits** are **not used**. The **codepoint subfield** can be used in **two** different ways:

  i. When the **3 rightmost bits are 0s**, the **3 leftmost bits are interpreted the same as the precedence bits** in the service type interpretation.

  ii. When the **3 rightmost bits are not all 0s**, the **6 bits** define **64 services**(divided into three categories) :

  - The **first category** contains **32 service types**; the **second** and the **third** each contain **16 service types**.
  - The **first category** is assigned by the **Internet authorities (IETF).**
  - The **second category** be used by **local authorities** (organizations).
  - The **third category** is temporary and can be used for **experimental purposes.**

Interpretation of Services Field



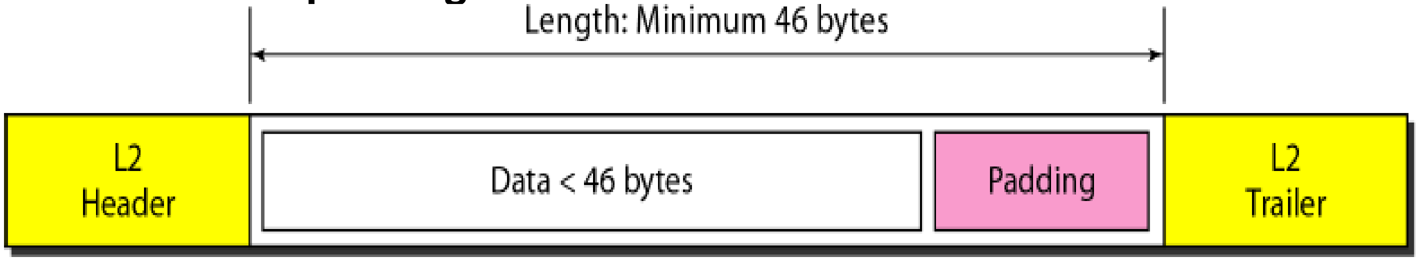
Category	Codepoint	Assigning Authority
1	XXXXX0	Internet
2	XXXX11	Local
3	XXXX01	Temporary or experimental

Datagram

d. Total length. This is a **16-bit** field that defines the **total length** **(header plus data)** of the **IPv4 datagram** in **bytes**.

Length of data* = *total length* - *header length

- Since the field length is **16 bits**, the **total length** of the **IPv4 datagram** is limited to **65,535** **(2^{16} - 1) bytes**, of which **20** to **60 bytes are the header** and the **rest is data** from the **upper layer**.
- If the size of an **IPv4 datagram** is **less than 46 bytes**, some **padding** will be added to meet this requirement. In this case, when a machine **decapsulates** the datagram, it needs to check the **total length** field to determine how much is really **data** and how much is **padding.**



Datagram

e. Identification. This field is used in **fragmentation** (discussed in the next section).

f. Flags. This field is used in **fragmentation** (discussed in the next section).

g. Fragmentation offset. This field is used in **fragmentation** (discussed in the next section).

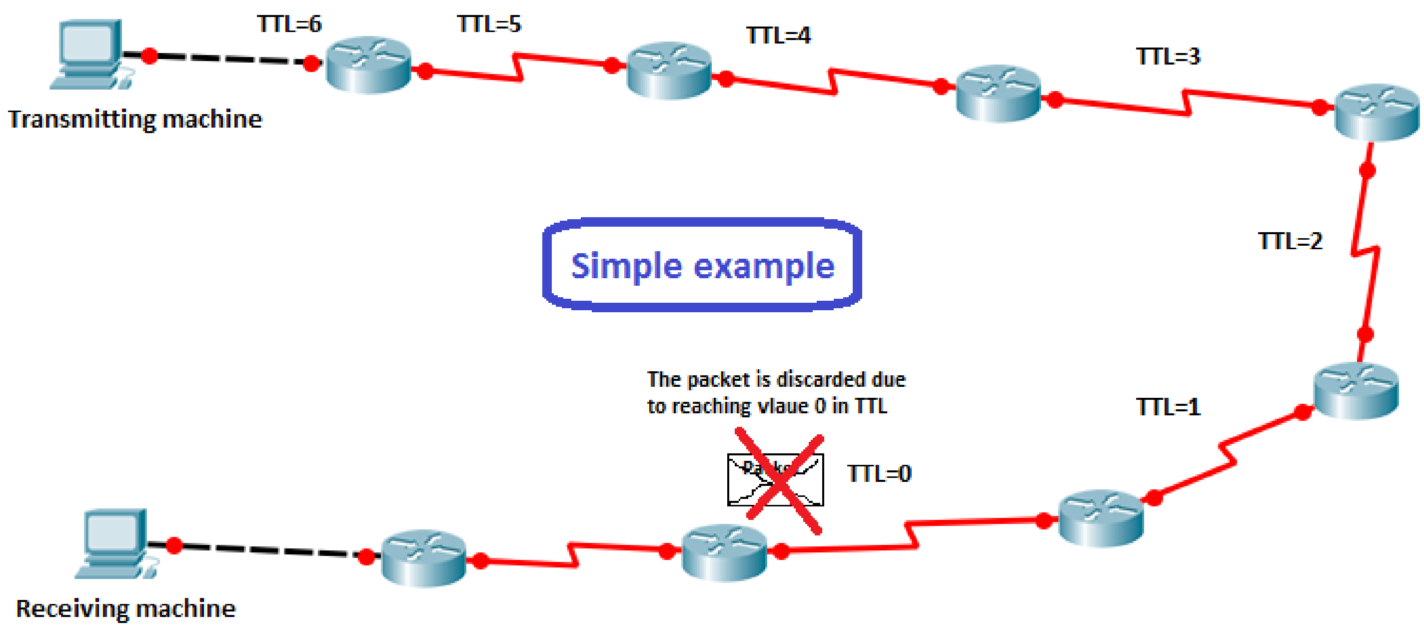
h. Time to live(TTL). A datagram has a **limited lifetime** in its travel through an internet. This field is used mostly to **control the maximum number of hops** (routers) **visited** by the **datagram.**

- When a **source host** sends the datagram, it stores a **number** in this field. This value is approximately **2 times** the **maximum number** of **hops between any two hosts**.
- Each **router** that processes the datagram **decrements** this **number by 1**. If this value, after being decremented, is **zero,** the router **discards** the datagram.

Need for TTL

- This **field** is **needed** because **routing tables** in the Internet can become **corrupted**.
- A **datagram** may travel between two or more **routers** for a **long time** without ever getting **delivered** to the **destination host.**
- This field **limits** the **lifetime** of a **datagram.**
- Another use of this field is to **intentionally limit** the **journey** of the **packet**.
- For **example,** if the **source** wants to **confine** the **packet** to the **local network**, it can store **1** in this field.
- When the **packet** arrives at the **first router**, this value is **decremented** to **0**, and the **datagram** is **discarded.**

Need for TTL


TTL=6
TTL=5
TTL=4
TTL=3
Transmitting machine
TTL=2
Simple example
The packet is discarded due
to reaching vlaue 0 in TTL
TTL=1
TTL=0
Receiving machine

Datagram

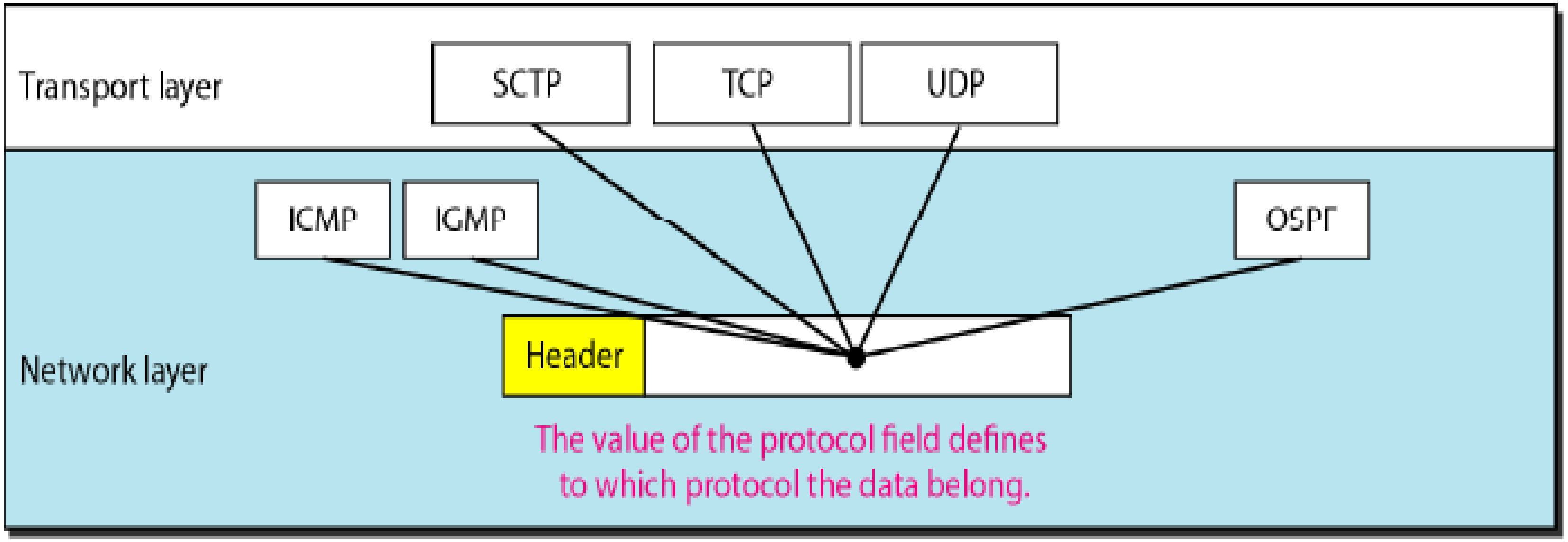
i. Protocol.

- This **8-bit** field defines the **higher-level protocol** that uses the services of the **IPv4** layer.
- An **IPv4 datagram** can encapsulate data from several higher-level protocols such as **TCP, UDP, ICMP,** and **IGMP**.
- Since the **IPv4** protocol carries data from different other **protocols**, the value of this **field** helps the **receiving network layer** know to which **protocol** the data belongs.

j. Checksum.

- The **checksum** is used to **secure** the **IPv4 header**.

Datagram



Transport layer
SCTP
TCP
UDP
ICMP
IGMP
OSPF
Header
Network layer
The value of the protocol field defines
to which protocol the data belong.

Datagram

k. Source address.

- This **32-bit field** defines the **IPv4 address** of the **source**. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.

l. Destination address.

- This **32-bit field** defines the **IPv4 address** of the **destination**. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.

m. Options.

- The header of the **IPv4 datagram** is made of two parts: a **fixed part** and a **variable part**. The **fixed part** is **20 bytes** long. The **variable part** comprises the **options** that can be a maximum of **40 bytes**. They can be used for **network testing** and **debugging**.

Examples

Example 1

An **IPv4 packet** has arrived with the first **8 bits** as shown:

01000010

The receiver **discards** the packet. Why?

Solution

- There is an error in this packet.
- The 4 leftmost bits (0100) show the version, which is correct.
- The next 4 bits (0010) show an invalid header length (2 x 4 =8).
- The minimum number of bytes in the header must be 20.
- The packet has been **corrupted** in transmission.

Examples

Example 2

In an **IPv4 packet**, the value of **HLEN** is **1000** in binary. How many bytes of **options** are being carried by this packet?

Solution

- The **HLEN** value is 8, which means the total number of bytes in the header is 8 x 4, or **32 bytes.**
- The first **20 bytes** are the **base header**, the next **12 bytes** are the **options.**

Examples

Example 3

In an **IPv4 packet**, the value of **HLEN** is **5**, and the value of the **total length** field is 0**x0028**. How many bytes of data are being carried by this packet?

Solution

- The **HLEN** value is **5**, which means the **total number** of **bytes** in the **header** is **5 × 4,** or **20 bytes** (no options).
- The total length is **40** (decimal equivalent of 0x0028) **bytes**, which means the packet is carrying **20 bytes** of **data** (40 − 20).

Examples

Example 4

An **IPv4 packet** has arrived with the first few hexadecimal digits as shown. **0x45000028000100000102 . . .** How many hops can this packet travel before being dropped? The data belong to what upper-layer protocol?

Solution

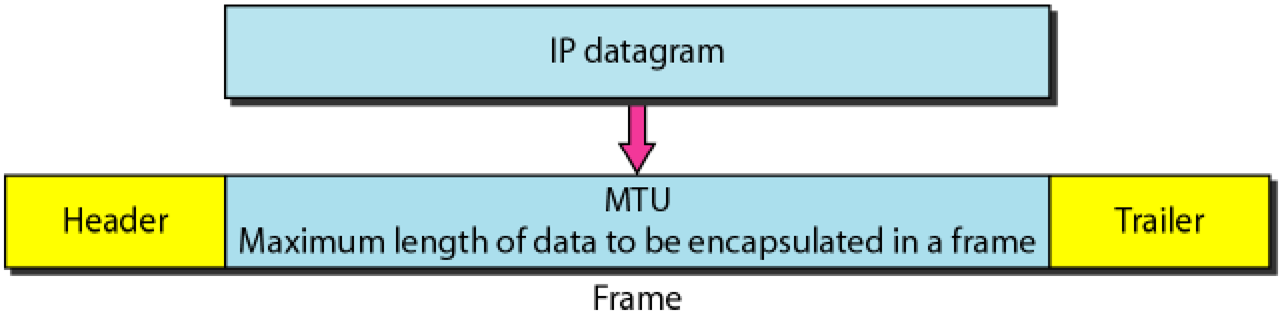
- In the IPv4 header, the **9th byte** is the **TTL.**
- The given hex digits :45 00 00 28 00 01 00 00 **[01]** 02 …
- The **TTL byte** is **0x01**, so the packet can traverse **one router (one hop)** before the **TTL** is decremented to **0** and the packet is **dropped.**
- The **protocol field** is the **next byte** (**02**), which means that the **upper-layer protocol** is **IGMP.**

Fragmentation

- A **datagram** can travel through **different networks.**
- Each **router decapsulates** the **IPv4 datagram** from the **frame** it receives, processes it, and then **encapsulates** it in another frame.
- The **format** and **size** of the **received frame** depend on the **protocol** used by the **physical network** through which the frame has **just traveled.**
- The **format** and **size** of the **sent frame** depend on the **protocol** used by the **physical network** through which the frame is **going to travel.**
- For **example,** if a **router** connects a **LAN** to a **WAN,** it receives a **frame** in the **LAN format** and sends a **frame** in the **WAN format.**
- Each **data link layer** protocol has its own **frame format** in most protocols.
- One of the **fields** defined in the **format** is the **maximum size** of the **data field.**

Fragmentation

- In other words, when a **datagram** is **encapsulated** in a **frame**, the **total size** of the **datagram** must be **less than** this **maximum size**, which is **MTU(Maximum Transferable Unit)** defined by the restrictions imposed by the hardware and software used in the network called



- The value of the **MTU depends** on the **physical network protocol**.
- Table on the next shows the values for some **protocols.**

MTU(Maximum Transferable Unit)

Protocol	*MTU*
Hyperchannel	65,535
Token Ring (16 Mbps)	17,914
Token Ring (4 Mbps)	4,464
FDDI	4,352
Ethernet	1,500
X.25	576
PPP	296

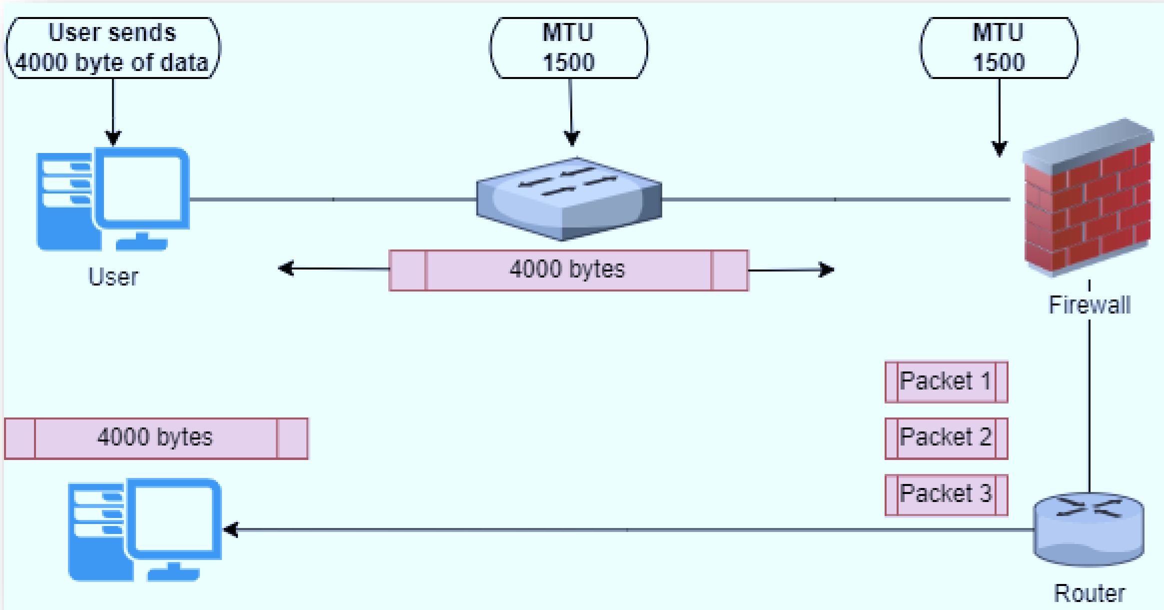
Fragmentation

- To make the **IPv4 protocol** independent of the **physical network**, the designers decided to make the **maximum length** of the **IPv4 datagram** equal to **65,535 bytes**.
- This makes **transmission more efficient** if we use a **protocol** with an **MTU** of this size.
- However, for other **physical networks**, we must **divide** the **datagram** to make it possible to pass through these networks. This is called **fragmentation.**
- When a datagram is **fragmented**, each **fragment** has its own **header** with most of the **fields repeated**, but with **some changed.**
- A **fragmented datagram** may itself be **fragmented** if it encounters a network with an even smaller **MTU.**

Fragmentation

- In other words, a **datagram** can be **fragmented** **several times** before it reaches the final destination.
- In **IPv4,** a datagram can be **fragmented** by the **source host** or any **router** in the **path** although there is a tendency to limit fragmentation only at the source.
- The **reassembly** of the **datagram**, however, is done **only** by the **destination host** because each **fragment** becomes an **independent datagram.**
- Whereas the **fragmented datagram** can travel through **different routes**, and we can never control or guarantee which route a fragmented datagram may take.
- All the **fragments** belonging to the **same datagram** should **finally arrive** at the **destination host**.
- So it is logical to do the **reassembly** at the **final destination**.

Fragmentation



Fragmentation

- When a **datagram** is **fragmented**, required parts of the **header** must be copied by all fragments.
- The **option field** may or may not be copied.
- The **host** or **router** that **fragments** a **datagram** must change the values of **three fields** in **IPv4 Header**:
  - ***flags,***
  - ***fragmentation offset*** *and*
  - ***total length****.*
- The rest of the fields must be **copied.**

Fields Related to Fragmentation

a. Identification

This **16-bit field** identifies a **datagram** **originating** from the **source host.**

- The **combination** of the **identification** and **source IPv4 address** must **uniquely define** a **datagram** as it **leaves** the **source host.**
- When the **IPv4 protocol** sends a datagram, it copies the current value of the **counter** to the **identification field** and **increments** the **counter** by **1.**
- When a **datagram** is **fragmented**, the value in the **identification field** is **copied** to **all fragments**. In other words, all fragments have the same identification number, the same as the original datagram.
- The **identification number** helps the **destination** in **reassembling** the datagram. It knows that all fragments having the same identification value must be assembled into one datagram.

Fields Related to Fragmentation

b. Flags.

This is a **3-bit** field.

- The **first bit** is **reserved.**
- The **second bit** is called the ***do not fragment*** **bit**.
- If its **value is 1**, the machine must **not fragment** the datagram. If it cannot pass the datagram through any available physical network, it **discards** the **datagram** and sends an **ICMP error message** to the **source host .**
- If its **value is 0**, the datagram ***can be fragmented*** if necessary.
- The **third bit** is called the ***more fragment*** **bit**.
- If its **value is 1**, it means **the datagram** is **not** the **last fragment**; there are **more fragments** after this one.
- **If its value is 0**, it means this is **the last** or **only fragments**.

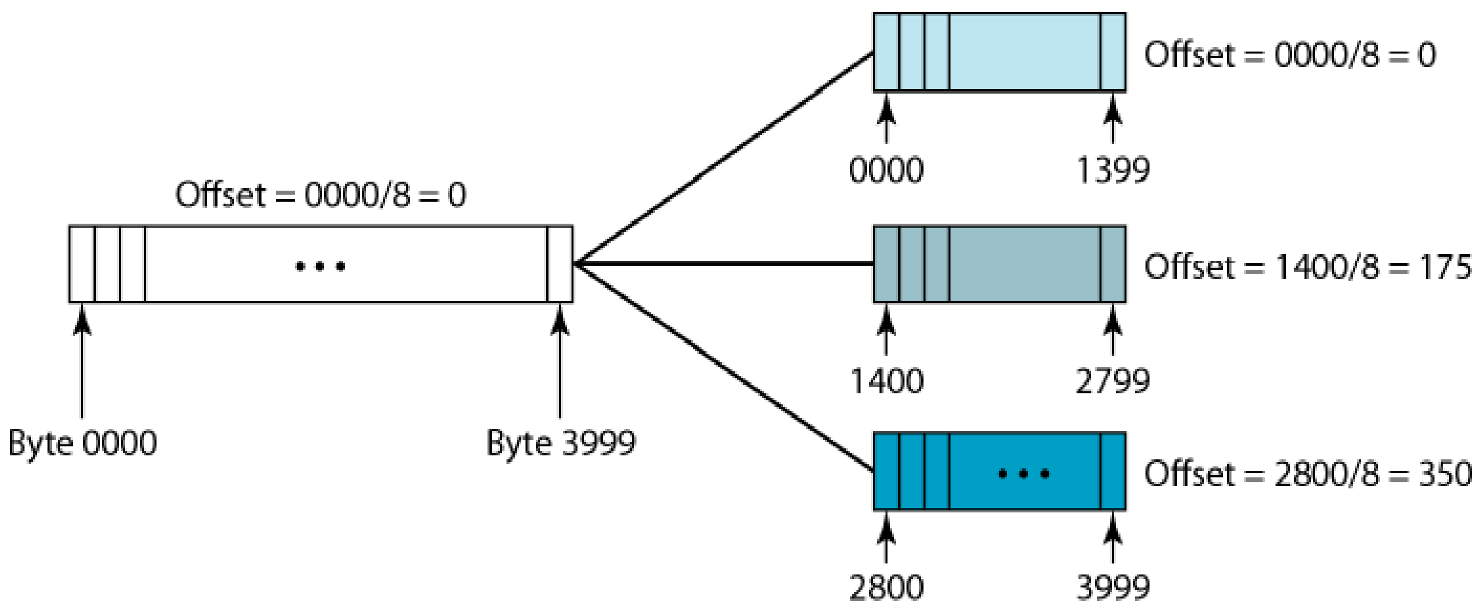
Fields Related to Fragmentation

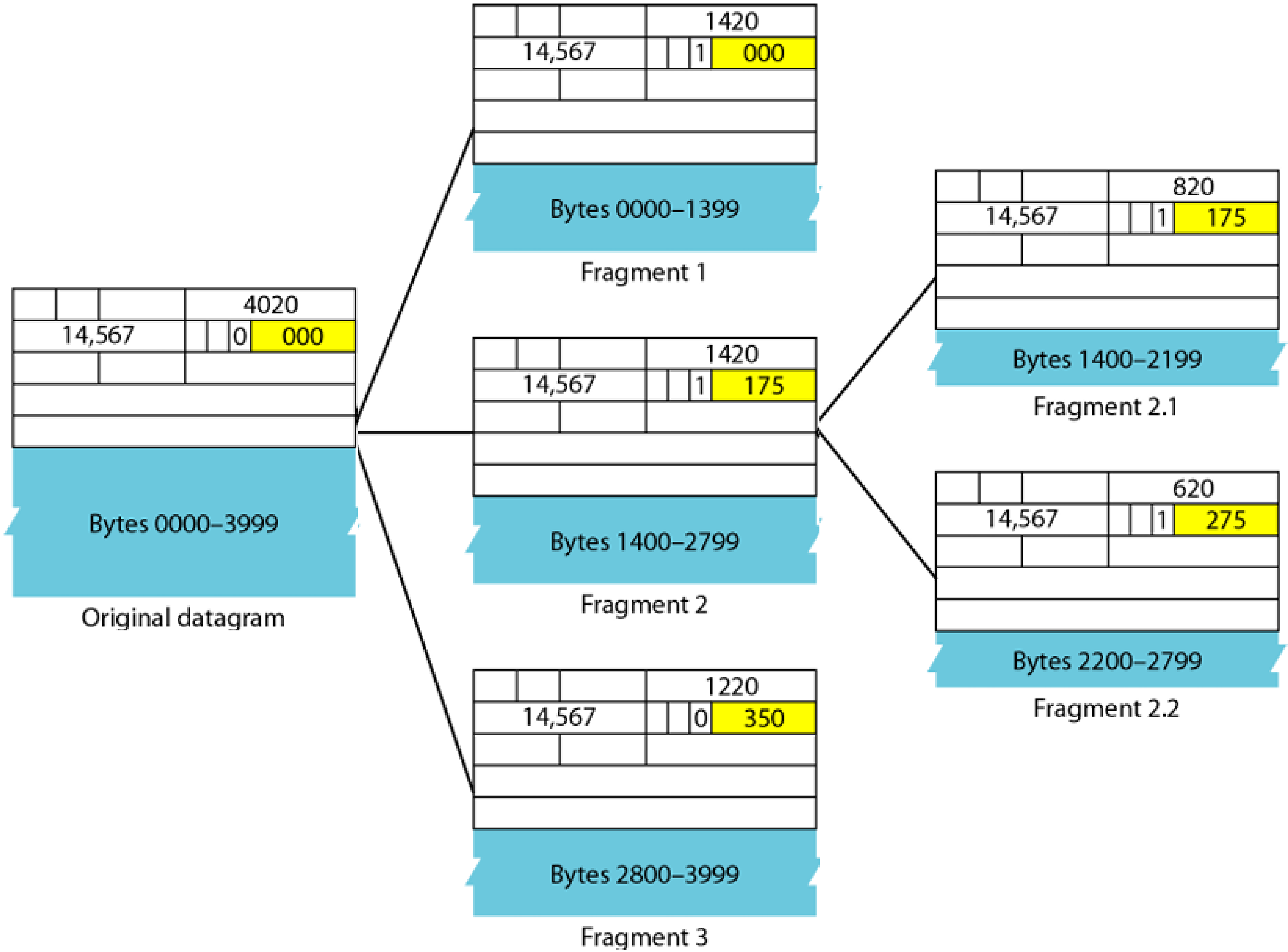
c. Fragmentation offset.

This **13-bit** field shows the **relative position** of this **fragment** with respect to the **whole datagram**.

- It is the **offset** of the data in the original datagram measured in **units** of **8 bytes**.
- Figure on next slide shows a datagram with a data size of **4000 bytes** fragmented into **three fragments**.
- The **bytes** in the **original datagram** are numbered **0 to 3999**.
- The **first fragment** carries bytes **0 to 1399**, the **offset** for this datagram is **0/8 =0.**
- The **second fragment** carries bytes **1400 to 2799**; the **offset** value for this fragment is **1400/8 = 175.**
- Finally, the **third fragment** carries bytes **2800 to 3999**. The **offset value** for this fragment is **2800/8 =350.**

Fragmentation Example




4020
14,567
0
000
Bytes 0000–3999
Original datagram
1420
14,567
1
000
Bytes 0000–1399
Fragment 1
1420
14,567
1
175
Bytes 1400–2799
Fragment 2
1220
14,567
0
350
Bytes 2800–3999
Fragment 3
820
14,567
1
175
Bytes 1400–2199
Fragment 2.1
620
14,567
1
275
Bytes 2200–2799
Fragment 2.2

Fragmentation Example

Example 1

An **IP datagram** of size **2000 bytes** (without header) with **identification field** = 12345 (say), Let the **MTU = 1500 bytes** (without header) perform **fragmentation** by assuming **IP header size = 20 bytes**. How **reassembling** will be done at receiver side?

Solution

- As **MTU=1500 bytes**, so **IP header + data** in each **packet** must be **≤ 1500 bytes**
- So **maximum data** that can be **packed** into **one fragment** = **1480 bytes**
- Thus we will break this **packet** into **2 fragments** of size **1480** and **520 bytes.**
- Put an **IP header** on **each fragment of 20 bytes.**

Fragmentation Example

Example 1. Solution (Contd.)

- **Header fields Fragment 1**
  - Identification = 12345, Offset = 0/8=0, MF = 1, Total length = 1500
- **Header fields Fragment 2**
  - Identification = 12345, Offset = 1480/8=185, MF = 0, Total length = 540
  - Rest of the fields same (except checksum).
- Send the **two fragments** as independent IP packets

Fragmentation Example

Example 1. Solution (Contd.)

- **Reassembly at receiver** can happen only after all fragments are received.
- If any one **fragment** is **lost/corrupted**, the **datagram** cannot be **reassembled.**
- All **fragments** received of that **datagram** **dropped** after a **timeout.**
- **How to know all fragments received?**
- **Offset** at each **fragment** says how many **bytes present** in other **datagrams** before this.
- **Gaps** can be **easily detected.**

Fragmentation Example

Example 2

Suppose the fragments created in **Example 1** need to be passed through a **router A** with the next link **MTU** value is **900**. How these fragments can be handled now?

Solution

- **Fragment 2** of size **520 bytes** will pass fine.
- **Fragment 1** of size **1480 bytes** will be fragmented again into **2 fragments**
- **Fragment 1a** and **1b** with size **880 bytes** (= 900 – size of IP header) and **600** (1480 – 880) bytes
- **Fragment 1a** Identification = 12345, Offset = 0, MF = 1, Total length = 900
- **Fragment 1b** Identification = 12345, Offset = 110, MF = 1, Total length = 620
- ***What if the MTU was 400, so both fragments will get fragmented again?***

Fragmentation Example

Example 3

A **packet** has arrived with an ***M* bit value** of **0**. Is this the **first fragment,** the **last fragment,** or a **middle fragment?** Do we know if the packet was fragmented?

Solution

- If the ***M* bit** is **0**, it means that there are no more fragments; the fragment is the last one.
- However, we cannot say if the original packet was fragmented or not.
- A **non fragmented** packet is considered the **last fragment.**

Fragmentation Example

Example 4

A **packet** has arrived with an ***M* bit** value of **1.** Is this the **first fragment**, the **last fragment**, or a **middle fragment**? Do we know if the packet was fragmented?

Solution

- If the ***M* bit** is **1**, it means that there is **at least one more** fragment.
- This **fragment** can be the **first one** or a **middle one**, but not the **last one.**
- We don't know if it is the **first one** or a **middle one**; we need **more information** (the value of the fragmentation offset).

Fragmentation Example

Example 5

A packet has arrived in which the **offset value** is **100**. What is the number of the **first byte**? Do we know the number of the **last byte**?

Solution

- To find the number of the first byte, we multiply the offset value by 8.
- This means that the first byte number is 800.
- We cannot determine the number of the last byte unless we know the length of the data.

Fragmentation Example

Example 6

A packet has arrived in which the **offset value** is **100**, the value of **HLEN** is **5**, and the value of the **total length** field is **100.** What are the **numbers** of the **first byte** and the **last byte?**

Solution

- The **first byte number** is **100 x 8 = 800.**
- The **total length** is **100 bytes**, and the **header length** is **20 bytes** (5 x 4), which means that there are **80 bytes** in this **datagram.**
- If the **first byte number** is **800**, the **last byte number** must be **879.**