Data Communication
(DC)

Lecture 6

Overview of the contents

- Network layer services
- Packet switching
- Network layer performance
- IPV4 addresses

Network Layer

Network Layer Service

The Network layer is responsible for Source-to-Destination (host-to-host) delivery of datagrams, probably across multiple networks. It provides services to the transport layer and receives services from the data-link layer

- The data-link layer handles frame deliveries within the same network.
- The network layer ensures that packets reach from their origin place to the final destination.

The network layer adds a header that contains:

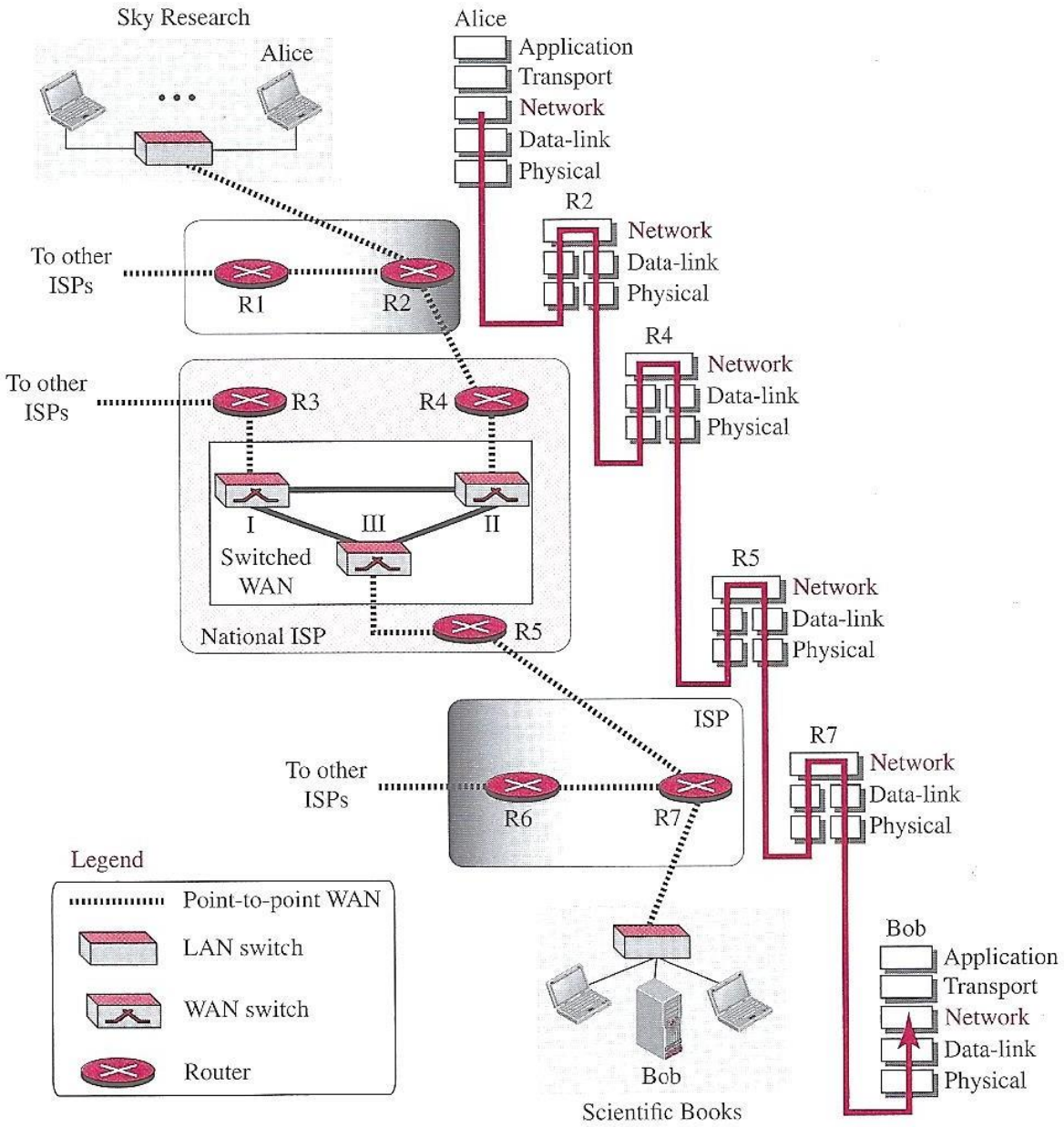
- **logical** address of sender
- **logical** address of receiver
- And some other information

One of the features of the network layer is to offer a routing mechanism.

Network Layer

Routers en route from Alice to Bob are not allowed to extract the encapsulated data packets unless they have to be fragmented.

If the packet is fragmented at the source or at routers along the path, the network Layer at the destination is responsible for waiting until all fragments arrive, reassembling them, and delivering them to the upper-layer protocol.

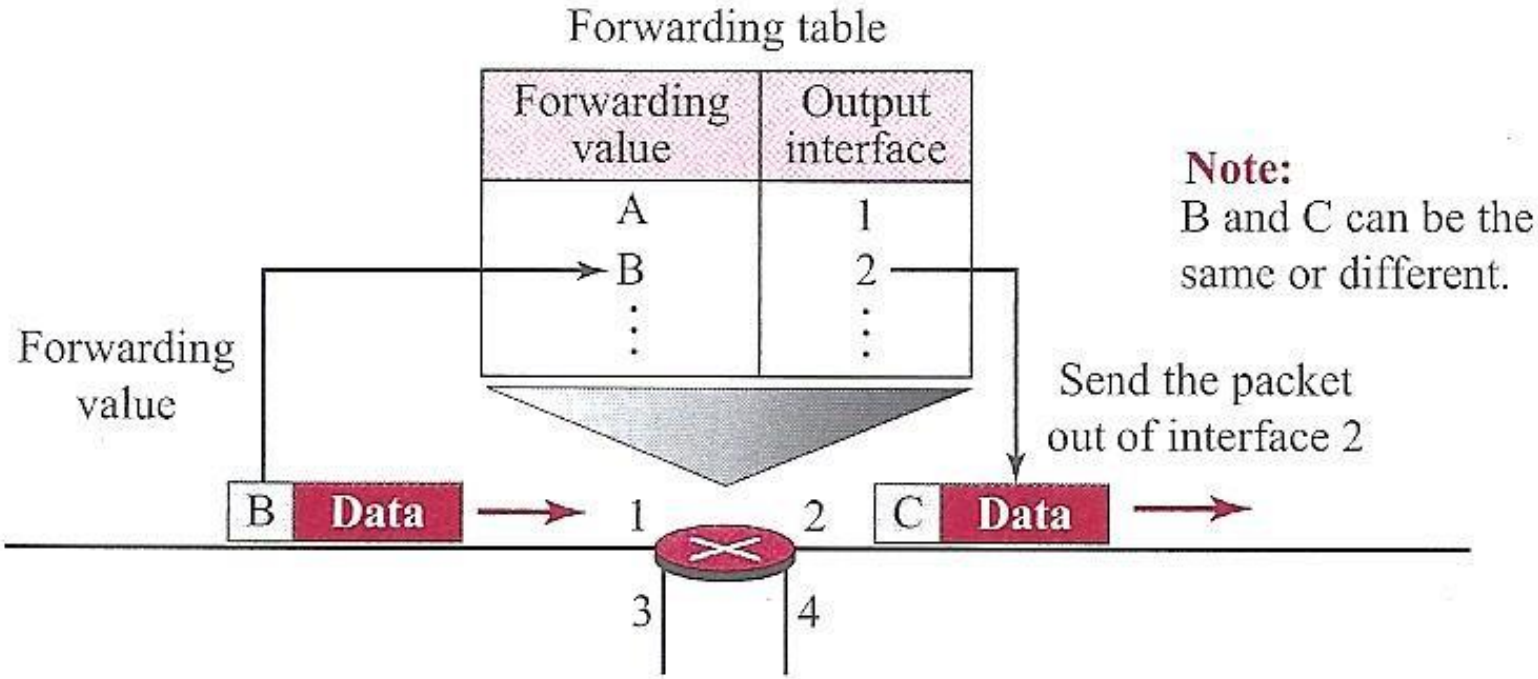


Network Layer

Routing and forwarding

It is the task of the Network layer to find the best route through the physical networks.

- **Routing** is applying strategies and running some routing protocols to create the **decision-making tables (forwarding or routing tables)** for each router.
- **Forwarding** is the action each router takes when a packet arrives at one of the router's ports and must be forwarded via another port according to the tables.



Network layer

Other services: Error control

We have previously talked about error detection and error correction.
Although error control can also be implemented here at the network layer,
We choose not to do it in the Internet. **One of the reasons is that packages can be fragmented, which makes error checking at this layer very inefficient.**

However, a checksum field has been added to the datagram so that you can check if the header is corrupted or not.
Note that this only applies to the header, not the entire datagram!

It should also be mentioned here that although the network layer does not directly provide error control functionality, an auxiliary protocol is used

”**I**nternet **C**ontrol **M**essage **P**rotocol (**ICMP**)”,

which is used when datagrams are discarded or have unknown information in the header.

Network layer

Other services: Flow control

Flow control is used to control the amount of data that a sender can send without overloading the receiver.
To make flow control work, the receiver must send some feedback back to the sender.

The network layer does <u>not</u> directly offer any flow control.
Datagrams are sent when they are ready, without focusing on whether the receiver can receive the datagrams.

Here are a few reasons why flow control is not offered:

- Since there is already no error checking, the network layer's job here is so simple that it is unlikely that the receiver will be overloaded.
- The upper layers that use the network layer can implement buffers to receive data from the network layer, so they do not have to consume data as fast as it is received at the network layer.
- flow control is provided for most of the upper-layer protocols that use the services of the network layer, so another level of flow control makes the network layer more complicated and the whole system less efficient.

Network Layer

Other services: Congestion control

Congestion control is a topic that the network layer needs to address.

Network congestion occurs when there is too much data traffic (too many datagrams) in an area of the Internet.

If a computer has more capacity than the network and the routers it is connected to, then congestion can occur.

In such situations, some routers may drop some of the datagrams, but due to the error control handled in the upper layers, the lost datagrams are re-transmitted. **This can lead to further congestion and perhaps eventually collapse.**

Congestion control in the network layer will be discussed (even if it is not implemented in the Internet)

Network Layer

Other services: Quality of Service (QoS)

As real-time audio and video streaming services became popular on the Internet, **QoS** is a concept that has become increasingly important.

The Internet has thrived by providing better quality of service to support these applications.

However, to keep the network layer untouched, these provisions are mostly implemented in the upper layer.

Network layer

Other services: Security

Security is an important issue in connection with data traffic on the network layer.

Security was not originally a topic of concern, as initially only a few universities and a few users used the Internet for research purposes. Others did not have access to the Internet at that time.

The network layer was created at that time without any kind of security facilities.

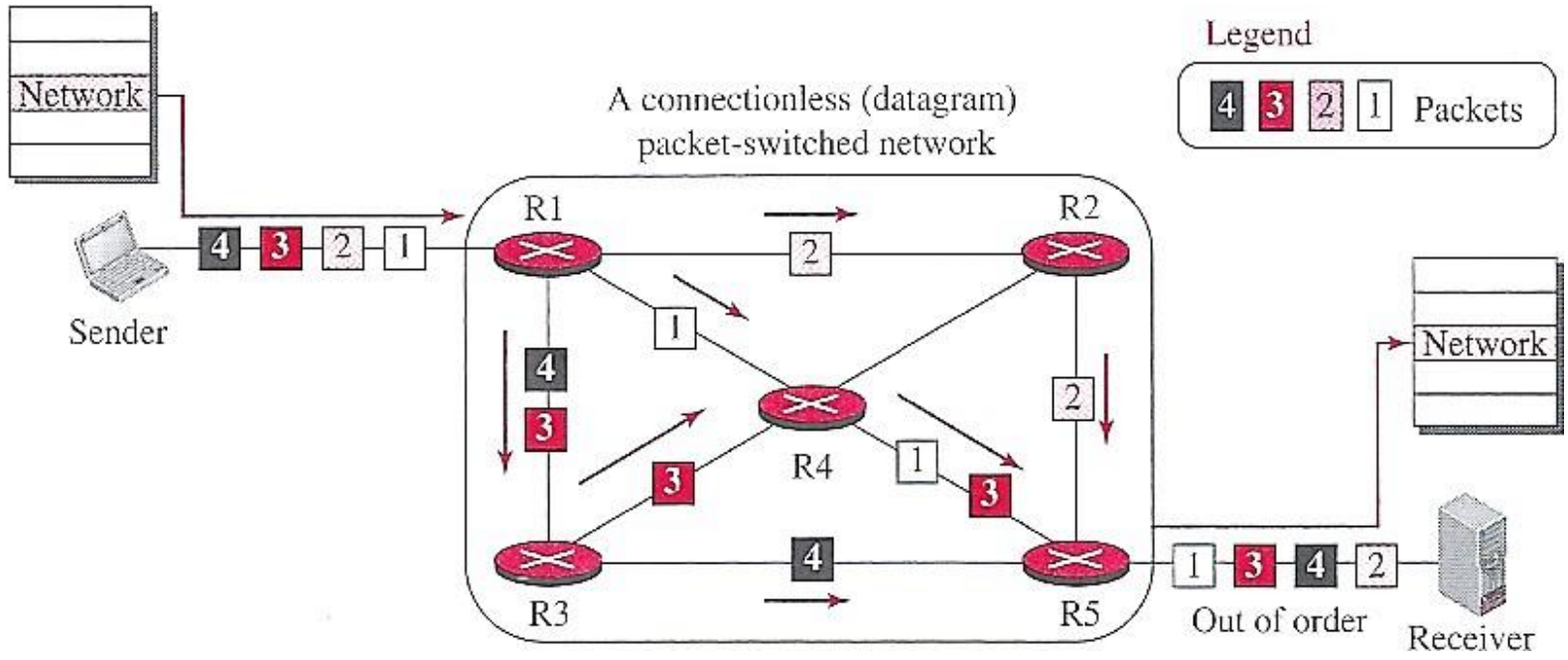
Today, security is important!

In order to add security to a connectionless network layer, we need to have another virtual layer that changes the connectionless service to a connection-oriented service. This virtual layer is called **IPSec**. (not part of the course)

Network Layer

Datagram approach: connectionless service

To make things simpler in the early days of the Internet, the network layer was designed to provide a connectionless service.

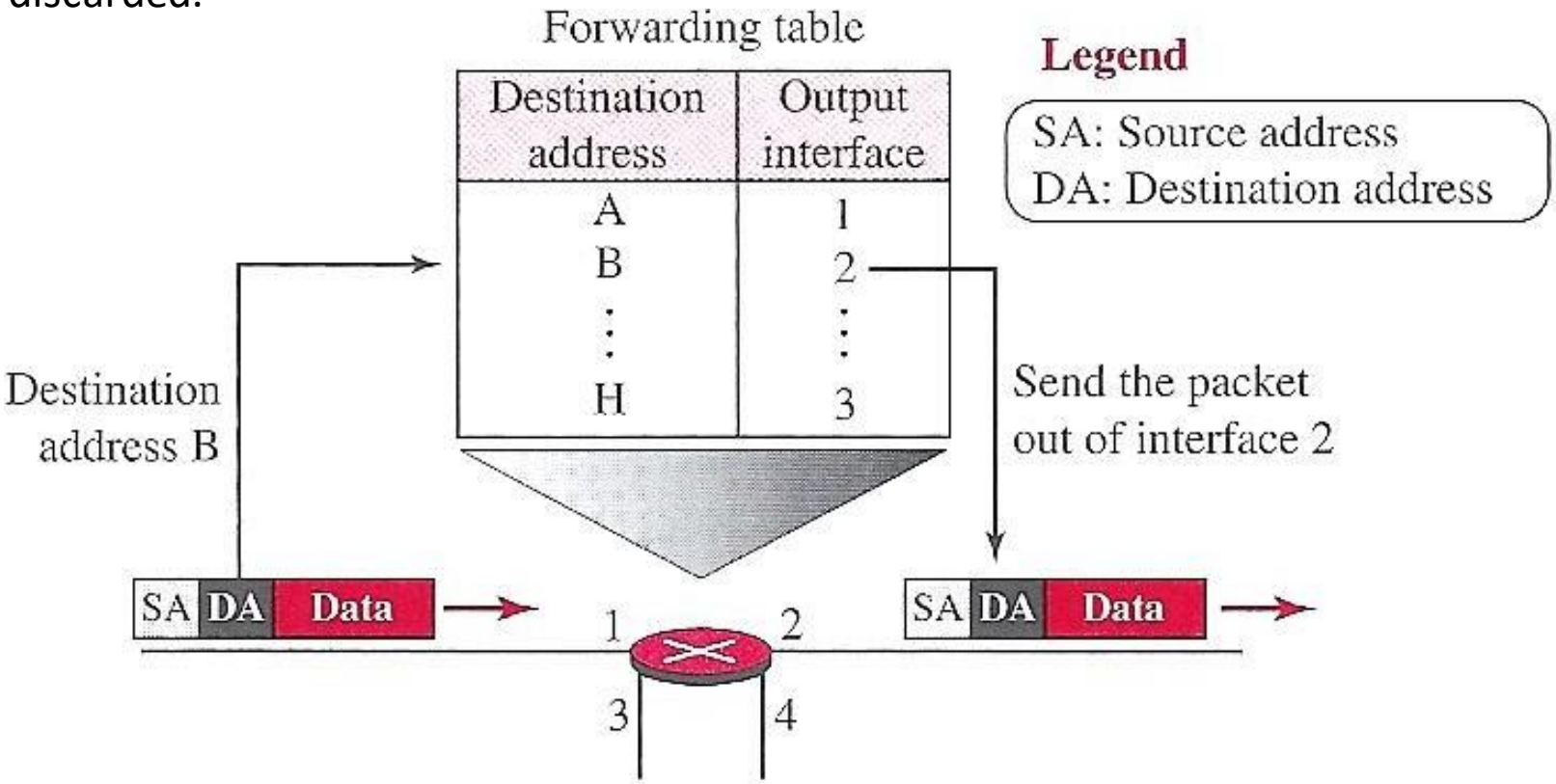


- Here, the network layer processes each packet independently, without any relation to each other.
- The idea was that the network layer was only responsible for delivery of packets from sender to receiver, not for the route the packets took.

Network Layer

Datagram approach: Forwarding mechanism

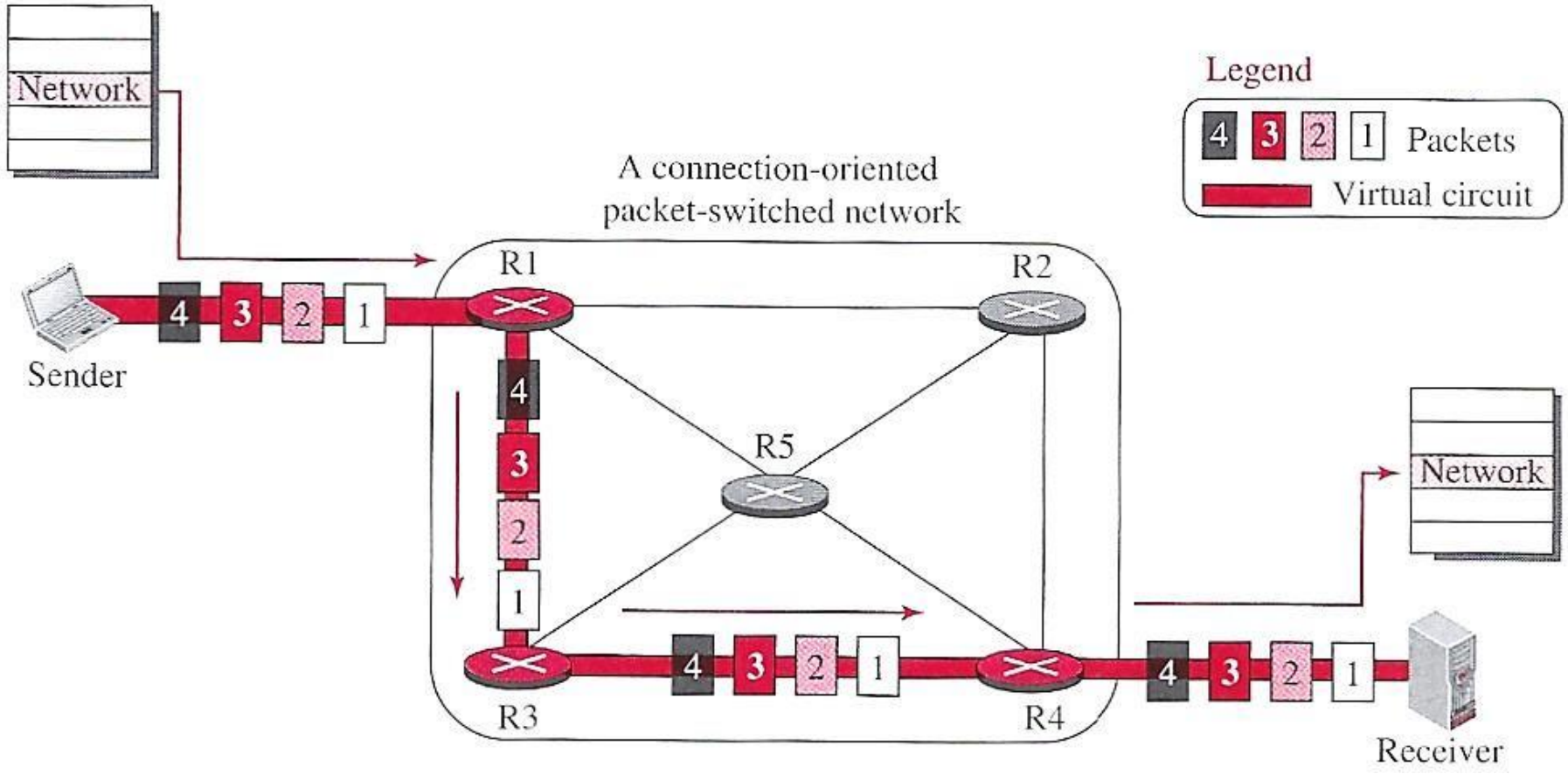
- Only the destination address is used for routing.
- The source address is used to send error message back to the sender if the packet is discarded.



Network Layer

Virtual-circuit approach: Connection-oriented service

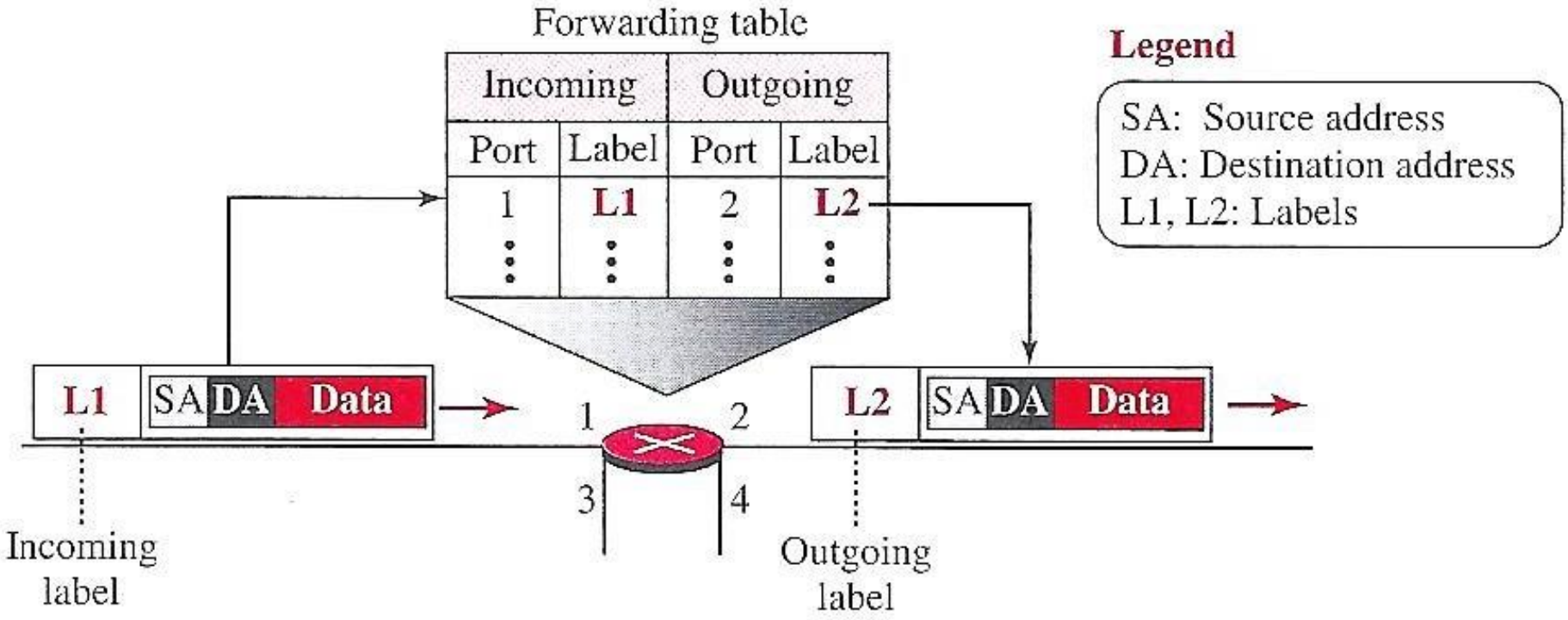
If this method is used, then there is a relationship between all packets belonging to the same message. Before all packets in a message can be sent, a virtual connection should be set up to define the path for the datagrams. After connection setup, the datagrams can all follow the same path. **A flow label, a virtual circuit identifier**, is used to define the path.



Network Layer

Virtual-circuit approach: Connection-oriented service

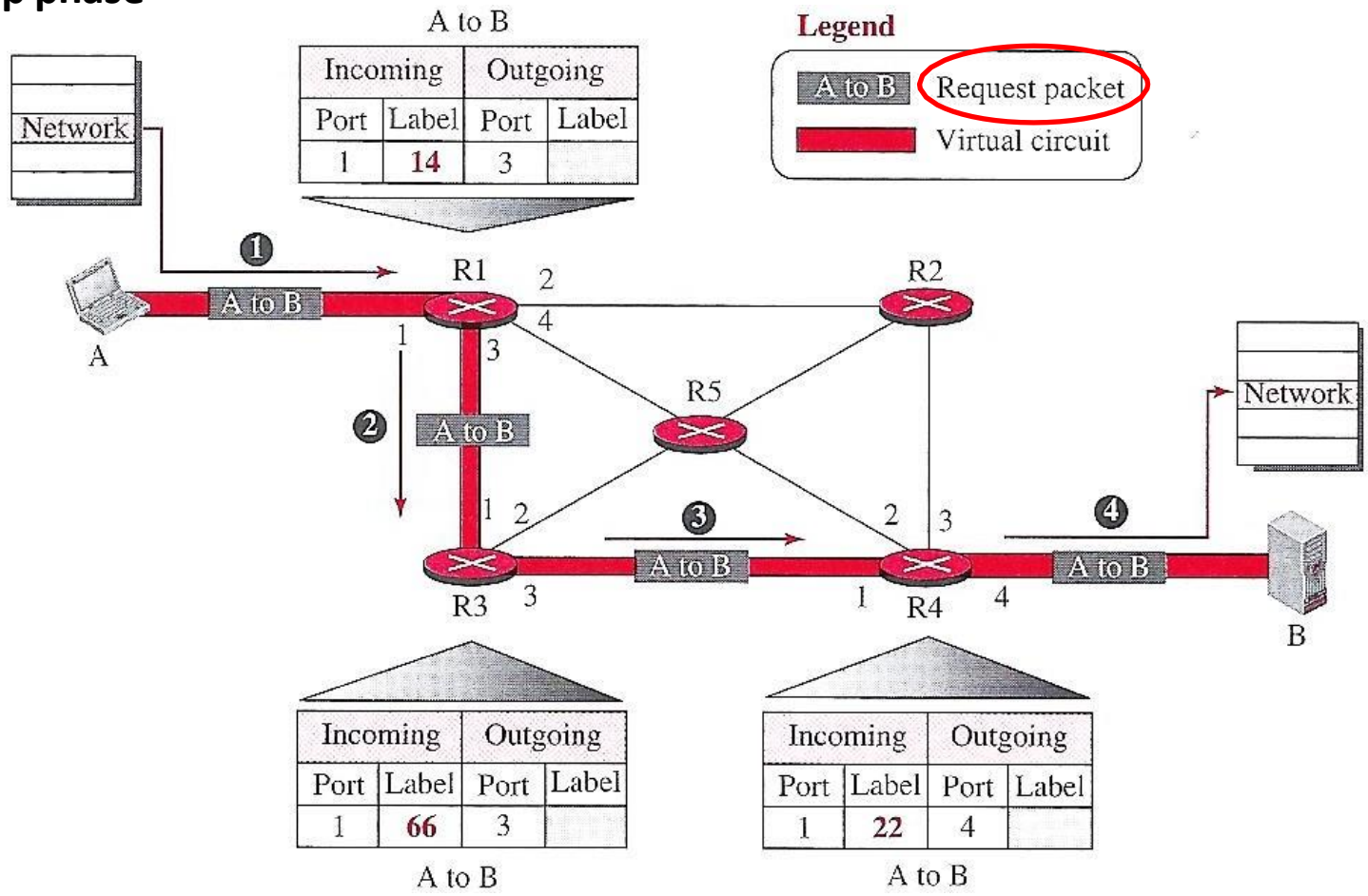
The datagram is packaged with the flow label. And the router uses this info for forwarding



Network Layer

Virtual-circuit approach: Connection-oriented service

The setup phase



Field, Outgoing Label, is reserved in all routers' forwarding tables with Request packet

Network Layer

Virtual-circuit approach: Connection-oriented service

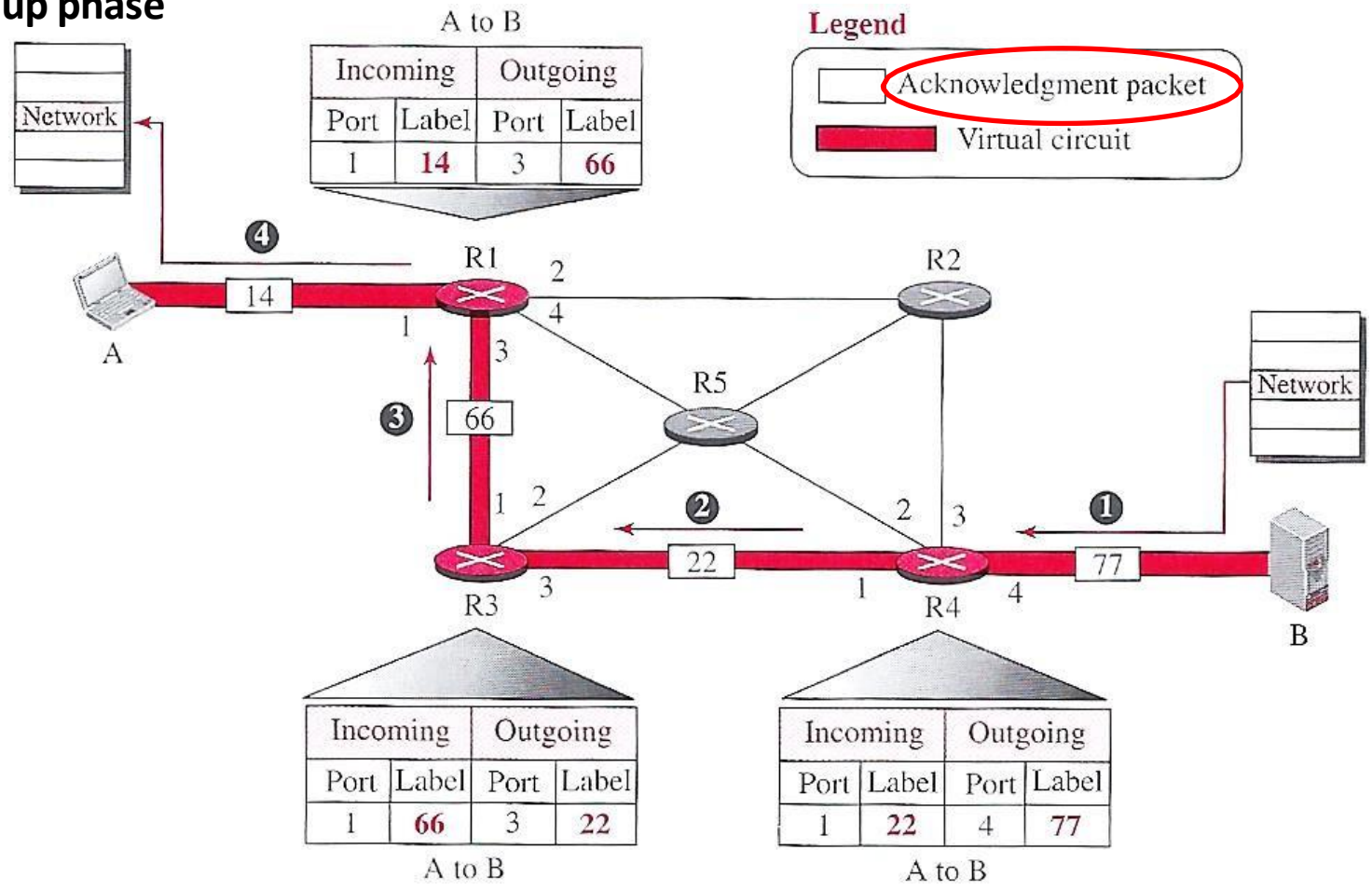
The setup phase

1. **A** sends a request packet to **R1**
2. **R1** receives the package. It can see that the connection must go from **A** to **B** through its own **port 3. R1** creates a line in its forwarding table, but can only fill in 3 of the 4 fields:
   - Incoming port = *1*
   - Incoming label = *14* (**R1** chooses it by itself)
   - Outgoing port = *3* (a ”knowledge” that **R1** has based on destination **B**)
   - Outgoing label = *?* (just as **R1** itself has chosen its label 14, so **R3** will do so too, but **R1** does not know it yet)
3. A request packet is sent from **R1** to **R3** which also reserves a line in its table, fills it in and sends a request packet on to **R4** etc. …
4. **B** receives the Request packet from **R4**, if it is ready to receive data from **A**, then it sends an Ack packet (here with a label = 77) back to **R4**

Network Layer

Virtual-circuit approach: Connection-oriented service

The setup phase

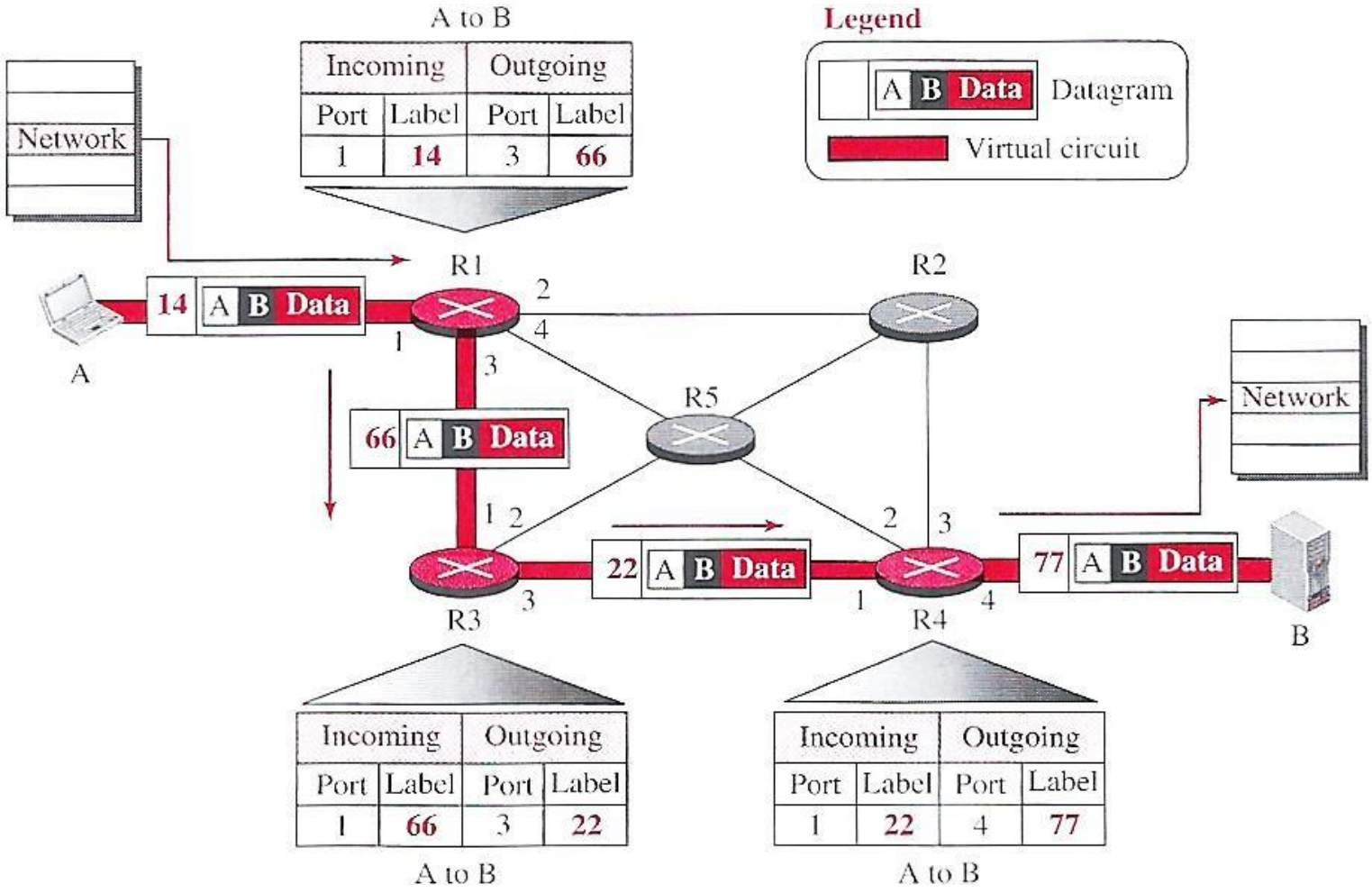


R4 can now fill in its missing field in its forward table and send an Ack packet to **R3** which can then fill in its missing field etc…

Network Layer

Virtual-circuit approach: Connection-oriented service

Once the virtual circuit is established, the data traffic in the virtual circuit will look like this:



Network Layer

Performance

Upper layer protocols that use service from the network layer expect to get an ideal service. But the network layer is not perfect.

We can measure the performance of the network layer in terms of:

- **Delay**
- **Throughput**
- **Packet loss**

You can improve the performance by **congestion control**

Network Layer

Performance: Delay

Anyone who has tried to use a network knows that there is a certain delay in the response you get from the network. This delay consists of four components:

Transmissions delay:

$$\textbf{Delay}_{tr} = \textbf{(packet length)/(transmission speed)}$$

e.g., 10.000 bits on a 100Mbps connection gives 100 µs

Propagation delay:

$$\textbf{Delay}_{pg} = \textbf{(distance)/(propagation speed in the medium)}$$

e.g., 2 km point-to-point WAN with a speed of 2×10^8 m/s in the cable gives 10 µs

Processing delay:

$\textbf{Delay}_{pr}$ **= time required to process packets in routers or destinations host**

Queuing delay:

$\textbf{Delay}_{qu}$ **= the time a packet is waiting in input and output queues in a router**

Total delay:

$$\textbf{Total Delay} = (n+1)(\textbf{Delay}_{tr} + \textbf{Delay}_{pg} + \textbf{Delay}_{pr}) + (n)(\textbf{Delay}_{qu})$$

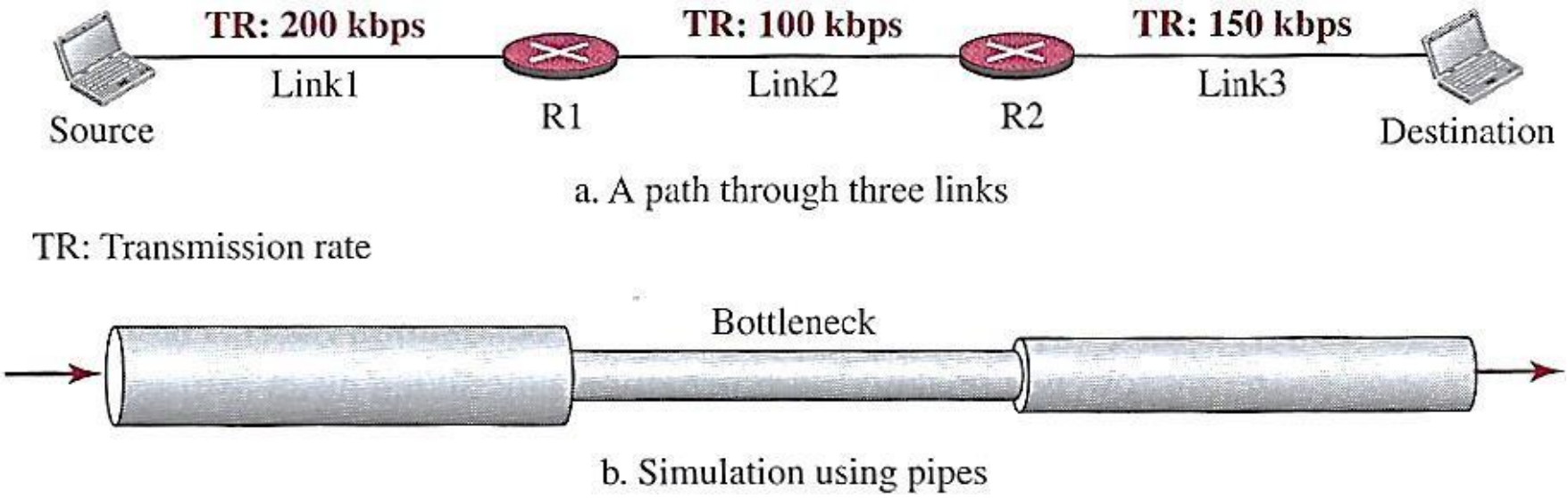
$(n+1)(\text{Delay}_{tr} + \text{Delay}_{pg} + \text{Delay}_{pr})$ are the links
$(n)(\text{Delay}_{qu})$ are the routers

Network Layer

Performance: Throughput

Throughput is defined as the number of bits that pass a given point in the network per second.

A packet that follows a route from sender to receiver will typically pass several links (networks) with different transmission speeds.



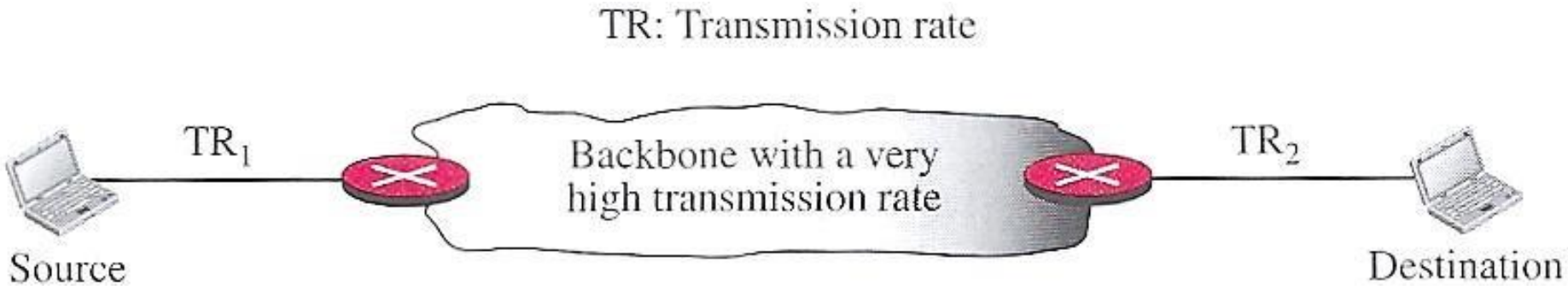
Here it is "the weakest link in the chain“ with "the smallest diameter“ that determines the throughput.

Thoughput = minimum{TR_1, TR_2, ... TR_n}

Network Layer

Performance: Throughput

However, the actual situation on the Internet is that the data normally passes through two access networks and the Internet backbone.

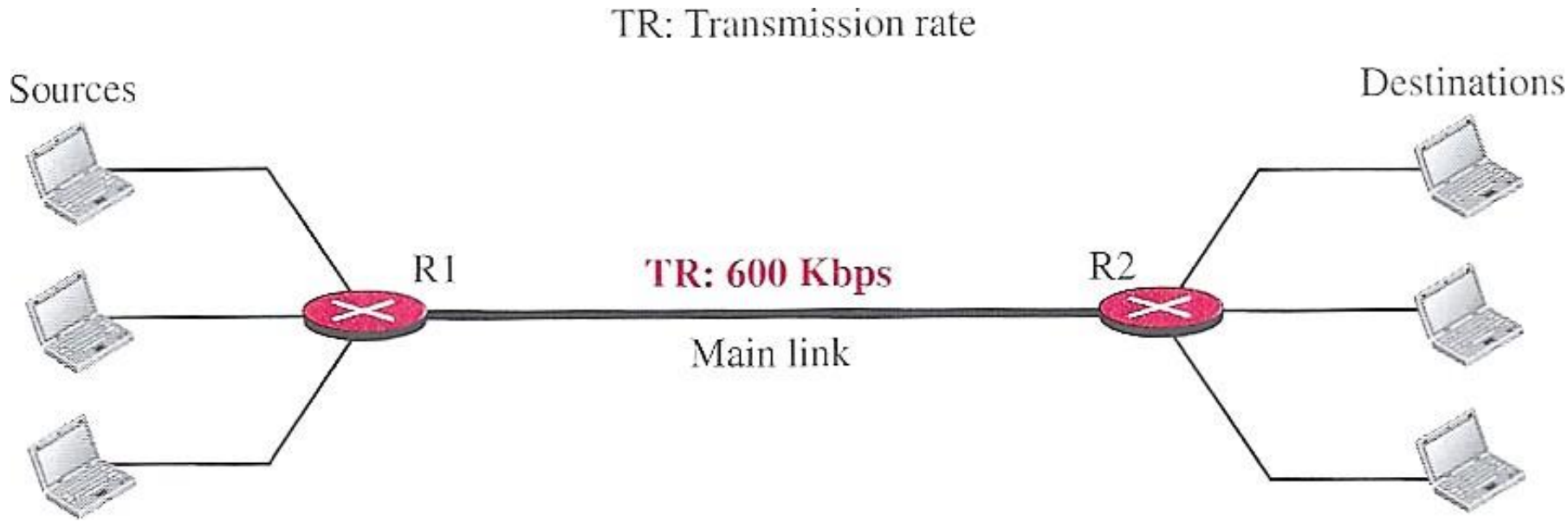


The Internet backbone has a very high transmission rate, in the range of gigabits per second. This means that the throughput is normally defined as the minimum transmission rate of the two access links that connect the source and destination to the backbone.

Network Layer

Performance: Throughput

You may also find that several hosts have to share links on some routes.



In this case, TR becomes only 200 Kbps, as the capacity must be shared between 3 pairs of hosts.

Network Layer

Performance: Packet loss

Package loss has a serious effect on the performance that can be achieved.

Although a router has buffers to store packets while others are being processed, these buffers have only limited sizes.

If they are filled up completely, then it can lead to loss of packets because the coming packets will be dropped!

This means that lost packets must be re-transmitted, which can lead to even worse data traffic and thus loss of even more packets.

Network Layer

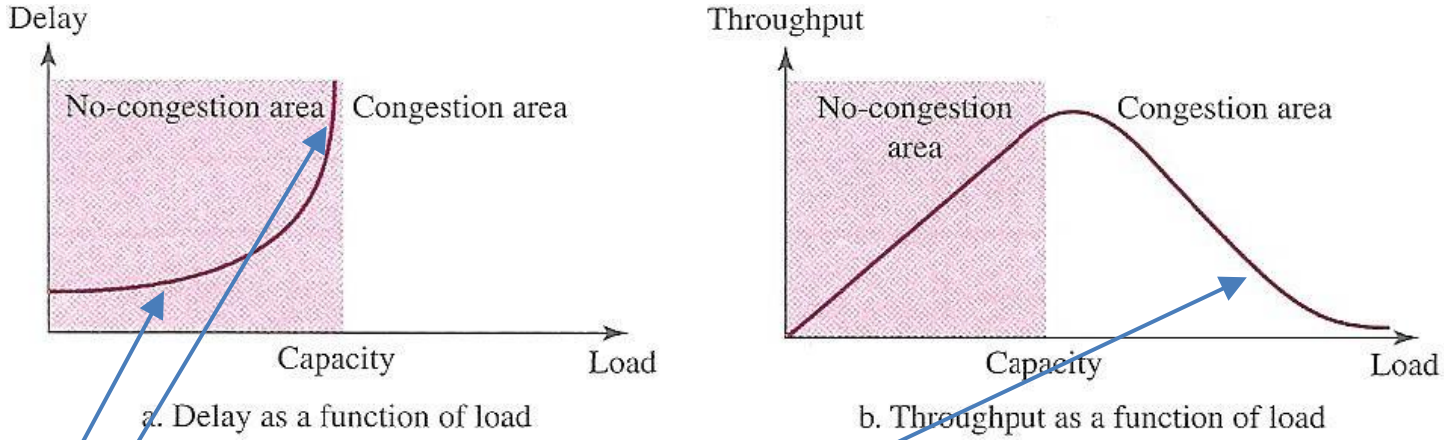
Performance: Congestion control

Congestion control is a mechanism that should help increase the performance. Although the Internet model does not directly address this issue on the network layer, but more on the transport layer (which we will see later).

A study of congestion at the network layer may help us to better understand the cause of congestion at the transport layer and find possible remedies to be used at the network layer.

Network Layer

Performance: Congestion control



The congestion in a network relates to two things::

- Delay
- Throughput

The shaded area shows the capacity of a medium

- As long as the load is less than the capacity, the delay is acceptable.
- But as the load approaches capacity, the delay increases extremely!
- When the capacity is exceeded, the throughput will begin to decrease due to packet loss.
- Discarding packets does not reduce the number of packets in the network because the sources retransmit the packets. (which will only aggravate the situation)

Network Layer

Performance: Congestion control

Two categories of control techniques:

- **Open-loop congestion control** (prevention = **before** the problem occurs)
- **Closed-loop congestion control** (removal = **after** the problem has occurred)

Network Layer

Performance: Open-loop congestion control

Retransmission policy: Retransmission cannot always be avoided. Here it is the sender who starts a timer.
If the timer expires before the sender has confirmed that the packet has arrived, it is retransmitted. (we have also seen this at the Data Link layer level)

It is important that the retransmission timer is optimized so that an answer has a reasonable time for reaching. But, also not for too short time as this may increase the congestion and worsen efficiency.

We remember that too much retransmission increases the load on the network.

Network Layer

Performance: Open-loop congestion control

Window policy: The types of windows (buffer space) used by the sender also affect the load in the system.

Selective Repeat Window is better than **Go-Back-N Window** in relation to the load

- Selective Repeat Window retransmits only missing packets. It is the receiver's responsibility to put them in order when they arrive.

- Go-Back-N Window retransmits all packets from a missing packet.
  The receiver does not need to be able to put them in order, as it will always receive the packet in order (if they do not come in order, the recipient asks to get them back from where the loss occurred)
  requires larger buffer at the sender.

Network Layer

Performance: Open-loop congestion control

Acknowledgement policy: If the sender is receipt-controlled by the receiver, then a proper frequency of the ACK can help regulate the load.

The receiver can also just send a receipt for **N** packets. Because we remember that ACK packages are also considered as loads in the system.

This policy seems familiar! This corresponds to flow and error control at the Data Link layer level.

Network Layer

Performance: Open-loop congestion control

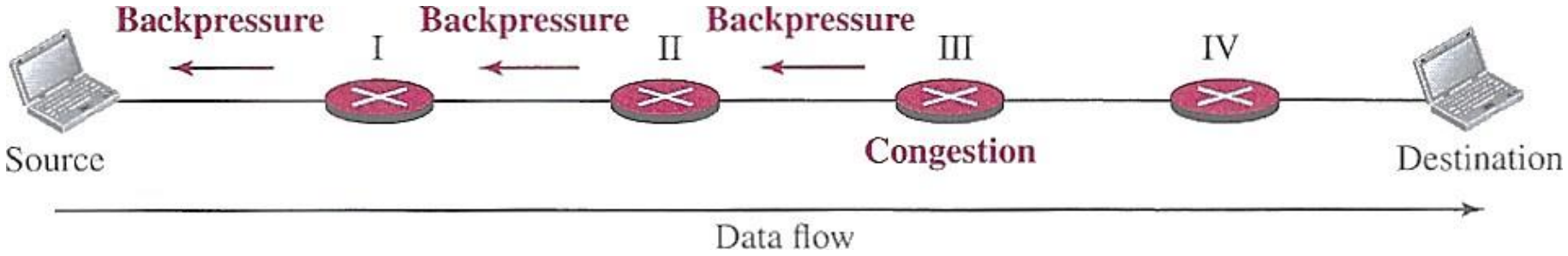
Discarding policy: Here it is the router that discards packets.
A good policy here can prevent congestion without harming the integrity of the transmission. An example here could be audio transmission, if you could discard less sensitive packets then congestion could be avoided without significantly affecting the sound quality.

Admission policy: this policy is linked to a QoS (quality-of-service) mechanism, which we will not go into in more detail in this course.
But it is equivalent to applying for a permit for a free link through the Internet, i.e., allocating resources (port reservation) in all routers/switches on the path "a kind of green channel", so that in principle you get an available point-to-point connection between sender and receiver.

Network Layer

Performance: Closed-loop congestion control

Backpressure is a technique in which a congested router stops receiving packets from its upstream node or nodes.



Note that this technique can only be used for connection-oriented data traffic, where the routers know where the traffic comes from.

(This is a bit like cars stopping in a queue)

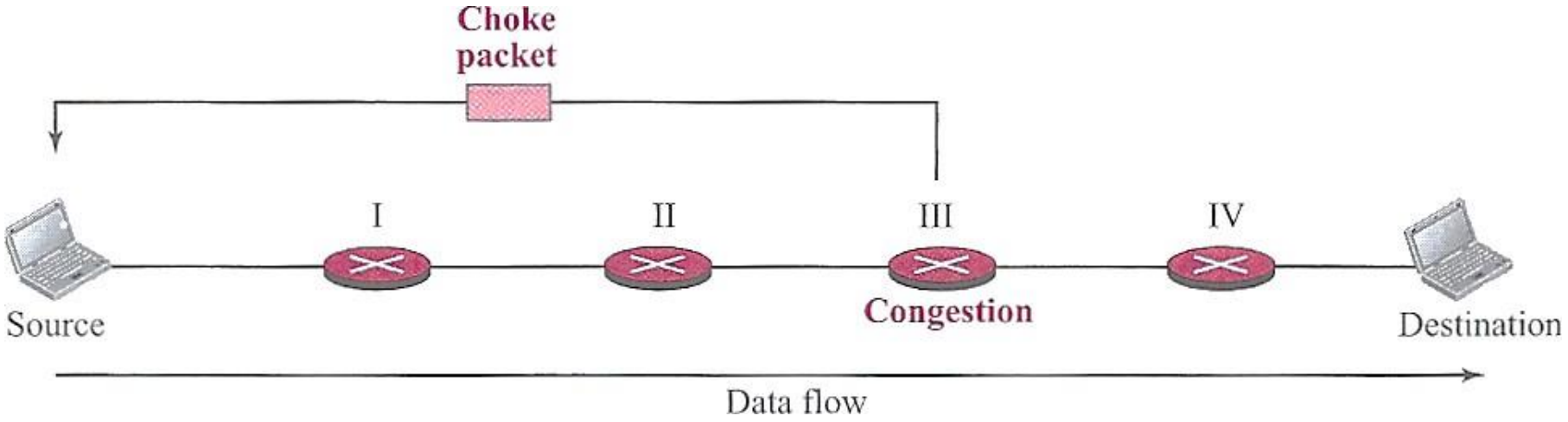
Network Layer

Performance: Closed-loop congestion control

Choke Packet: a Choke Packet is a special data packet sent directly from a congested router to the sender (source) to inform of the congestion.

As we will see later in the course, the **ICMP** (**I**nternet **C**ontrol **M**essage **P**rotocol) protocol is used.

- Here, a router finds that it is being overwhelmed with IP datagrams.
- It discards some of them.
- And notifies the sender directly here via **ICMP**.
- The intermediate routers (I and II in the figure) do not get notified.



Network Layer

Performance: Closed-loop congestion control

Implicit signaling: In implicit signaling, there is no communication between the congested node or nodes and the source.

If a sender (source) detects that it takes too long before a ACK receipt is received, then it can interpret it as an overload problem and slow down sending packets.

Explicit signaling: A router that experiences an overload can explicitly send a signal to the sender or receiver.

This method differs from the **Choke packet method**, where a separate packet was sent. (e.g., ICMP packet)

Here, the signal is included in the packets that carry data (and thus not in another protocol).
This type of congestion control is often seen in **ATM** (**A**synchronous **T**ransmit **M**ode) networks, i.e., long-distance networks.

Network Layer

Logical addressing

In order to deliver packets from one place in the world to another, we need a global addressing system.

When we refer to the TCP/IP protocol suite, this is called **IP addressing**.

IP address is the address of the connection, not the host.

If the host is moved to another network, then the IP address is probably different.

Network Layer

IPv4 addressing

An IP-address (ver. 4), **IPv4**, used today is a **32-bit address**.
This address defines a unique connection of a device (for example, router or computer) to the Internet.

Two different devices connected to the Internet cannot have the same addresses. If a device has ***m*** connections to the Internet, then it also needs to have ***m*** addresses.
A router is an example of such a device.

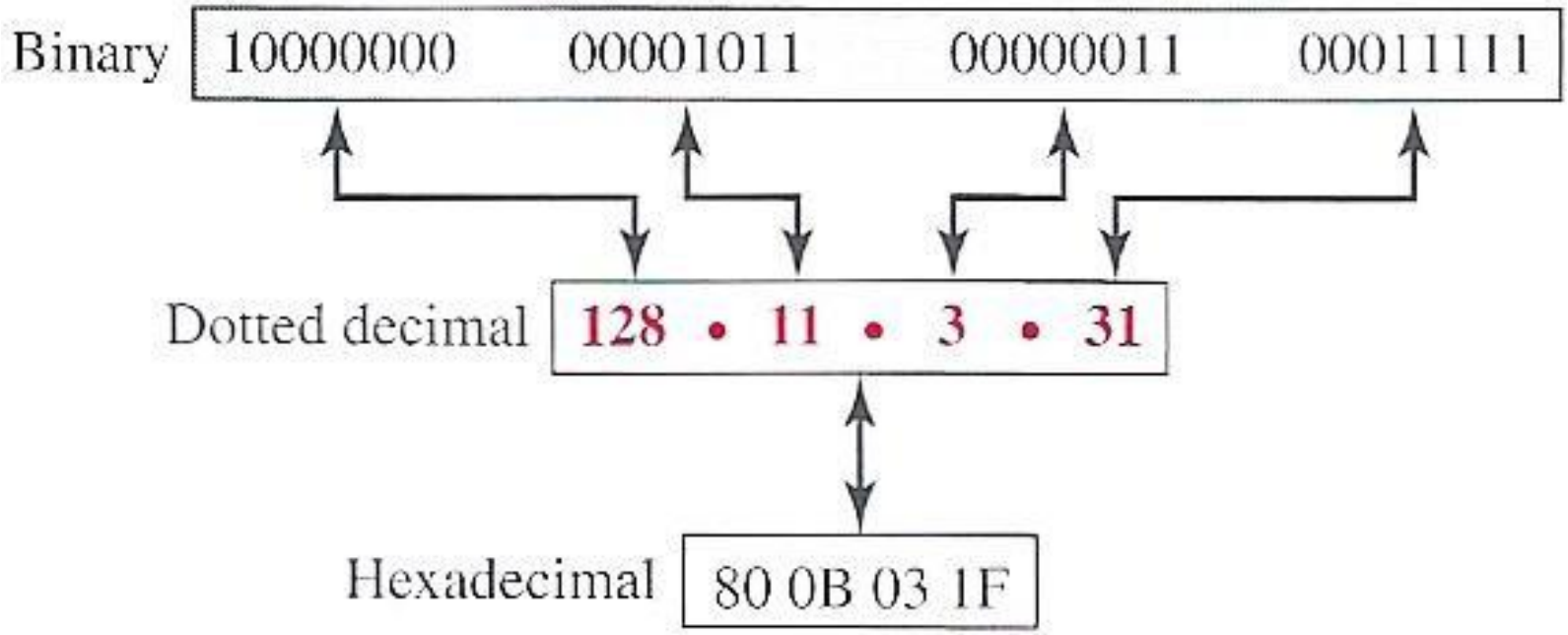
Everyone who wants to connect to the Internet must accept the IP addressing system.

Network Layer

IPv4 addressing

Since a IPv4 address has 32-bit, it gives:

$IPv4\colon\ 2^{32} = 4.294.967.296\ different\ addresses\ (or\ almost\ 4.3\ billion)$

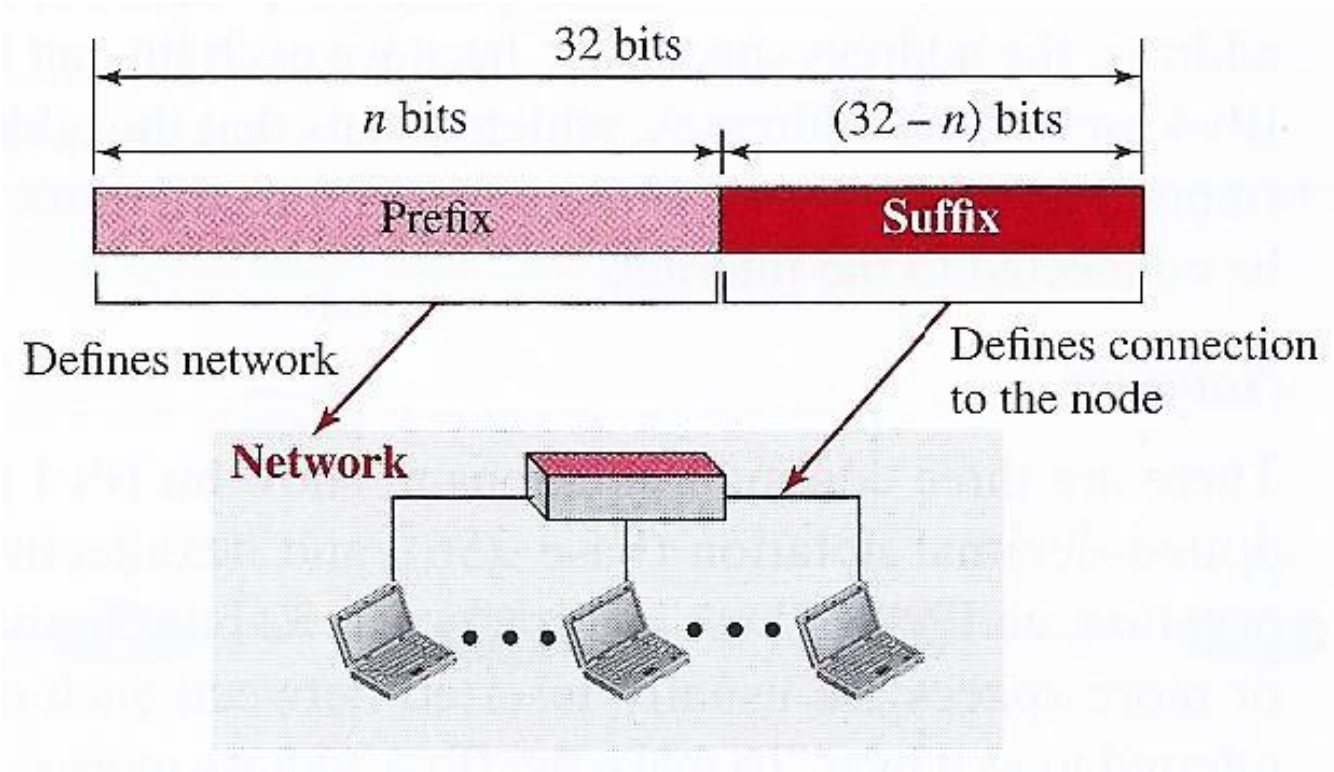


The decimal notation is the most common.

Network Layer

IPv4 addressing

A 32-bit IPv4 address is hierarchical and divided into two parts: a **prefix** and a **suffix**.

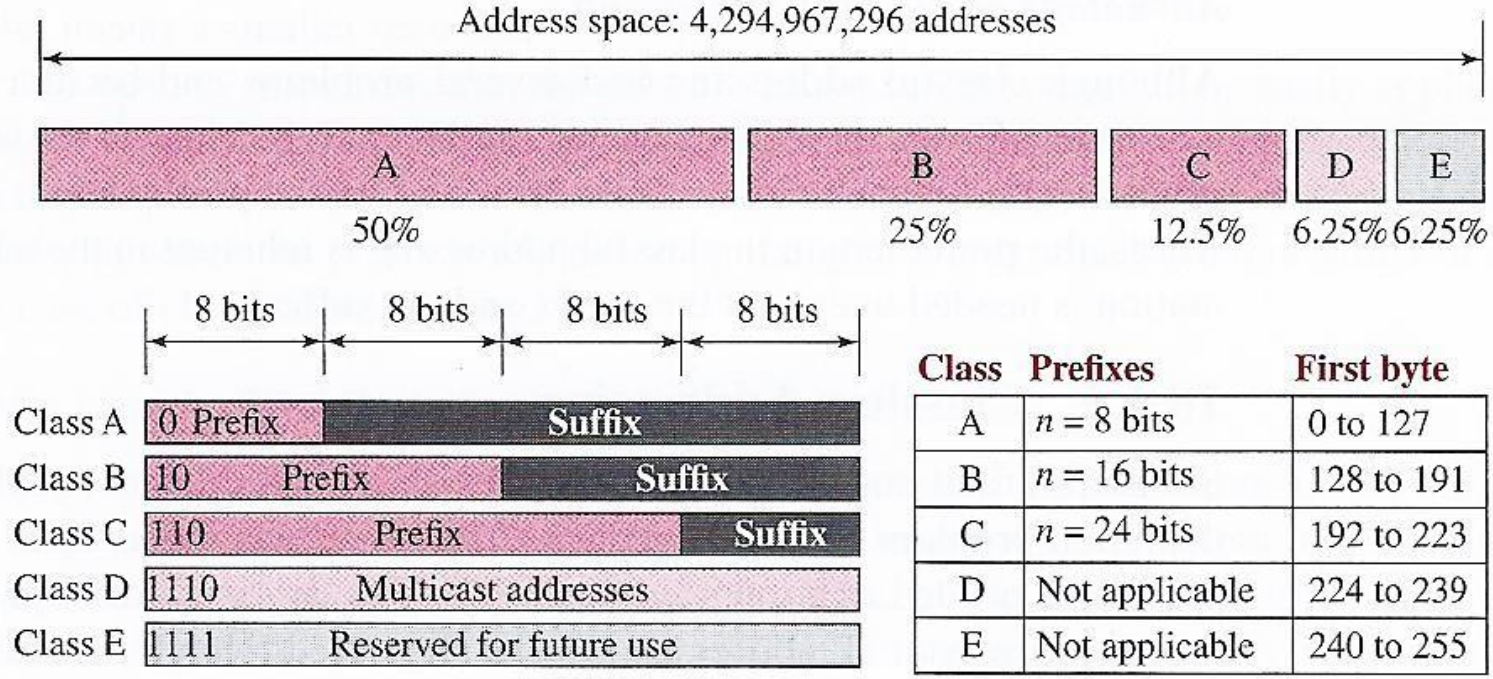


- The prefix defines the network (**Net ID**)
- The suffix indicates the node connection (**Host ID**)

Network Layer

IPv4 addressing: Classful Addressing

Initially, a classification system was used when addresses had to be specified.
IP addresses were divided into 5 classes: A, B, C, D and E.
Each of these classes occupied a portion of the address space.
Here, fixed prefixes were used, but in order to accommodate both small and large networks, 3 prefixes with fixed lengths are used: n = 8, n = 16 and n = 24.



Class	Prefixes	First byte
A	$n = 8$ bits	0 to 127
B	$n = 16$ bits	128 to 191
C	$n = 24$ bits	192 to 223
D	Not applicable	224 to 239
E	Not applicable	240 to 255

Network Layer

IPv4 addressing: Classful Addressing

One of the problems here was that the classification system divided each class into a fixed number of blocks with a fixed size. Since the addresses were not distributed properly, the Internet was faced with the problem of the addresses being rapidly used up, resulting in no more addresses available for organizations and individuals (**address depletion**).

Class	*Number of Blocks*	*Block Size*	*Application*
A	128	16,777,216	Unicast
B	16,384	65,536	Unicast
C	2,097,152	256	Unicast
D	1	268,435,456	Multicast
E	1	268,435,456	Reserved

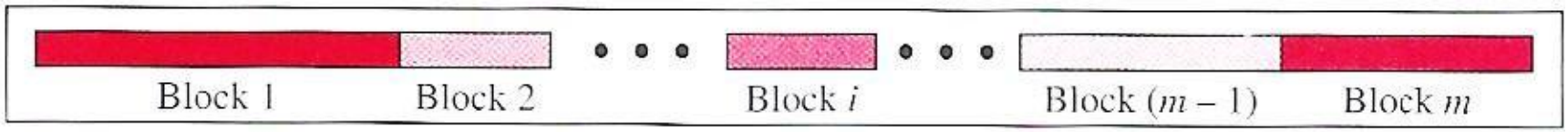
- Class A was far too large for almost all organizations. Too many unused addresses.
- Class B was also too large for most organizations. Too many unused addresses.
- Class C was too small for most organizations. Too few addresses.

This classification leads to a consequence that too many available addresses were not used and wasted, which is the reason that the classful addressing has become obsolete.

Network Layer

IPv4 addressing: Classless addressing - Address block

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.



Address space

Address blocks

When a small or large business needs to get access to the Internet, it is assigned a block of addresses.

The size of this block is determined by the type and size of the business.

- A private household may receive only a few addresses.
- A large company might get a few thousand addresses.
- An **ISP** (**I**nternet **S**ervice **P**rovider) may get 1000 or 100,000 addresses (based on the number of customers)

Network Layer

IPv4 addressing: Classless addressing - Address block

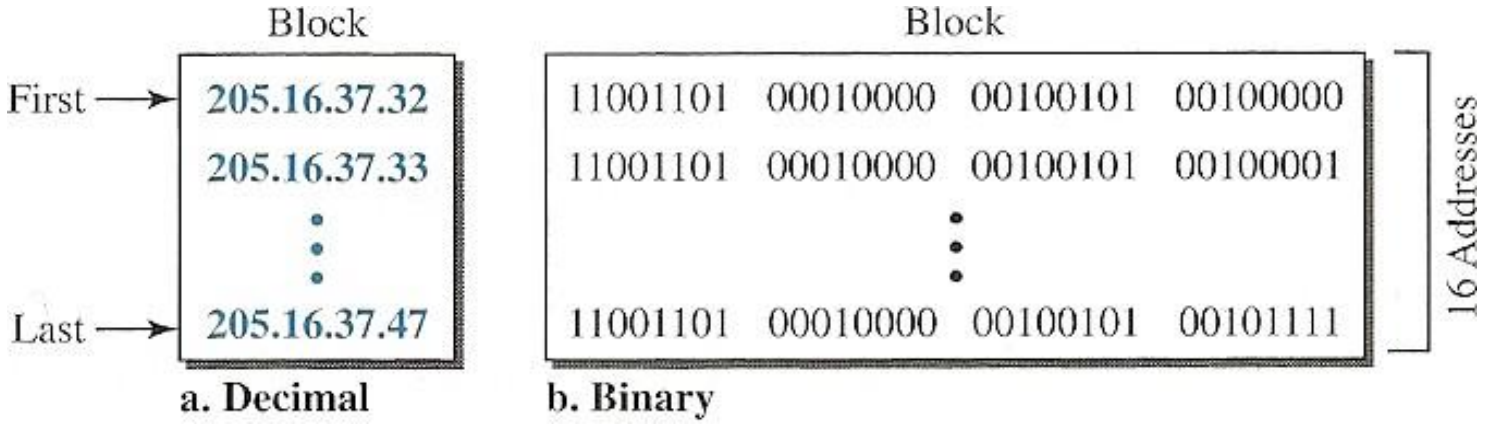
To simplify the handling of addresses, the Internet authorities have imposed 3 restrictions for classless address blocks:

1. An address block must have a set of continuous addresses.
2. The number of addresses in a block must be 2^n (1, 2, 4,8,…)
3. The first address (in decimal) in the block must be divisible by the number of addresses in the block.

Internet **C**orporation for **A**ssigned **N**ames and **N**umbers (**ICANN**) is the authority for block allocation.

Network Layer

IPv4 Addressing: Classless Addressing - Address Block (Example)



As we can see, the guidelines for this address block have been followed.

- The addresses are continuous.
- The number of addresses follows 2^n, There are 16 addresses (which is 2^4).
- The first address is divisible by 16 (see below).

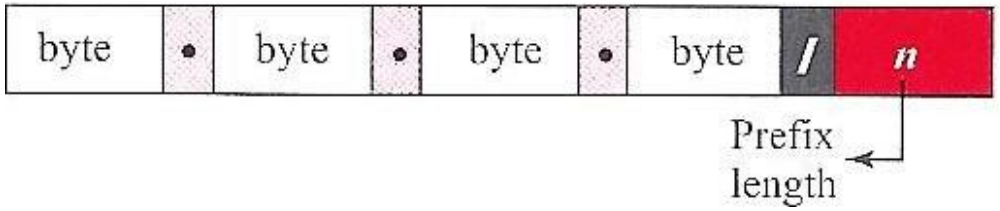
$$\mathbf{205} \cdot 256^3 + \mathbf{16} \cdot 256^2 + \mathbf{37} \cdot 256^1 + \mathbf{32} \cdot 256^0 = 3.440.387.360$$

$$\frac{3.440.387.360}{16} = 215.024.210$$

Network Layer

IPv4 Addressing: Classless Addressing - Prefix Length: Slash Notation

Since the prefix length is not inherent in the classless addressing, we need to separately give the length of the prefix. In this case, 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**.



Examples:
12.24.76.8/**8**
23.14.67.92/**12**
220.8.24.255/**25**

In IPv4 addressing, a block of addresses can be defined as:

x.y.z.t/n

where **x.y.z.t** is an address in this block and **/n** is the prefix length.

An **address mask** is defined as a 32-bit number in which the **n** leftmost bits are set to 1s and the rest of the bits **(32 – n)** are set to 0s.

Network Layer

IPv4 addressing: CIDR – example

205.16.37.39/28

What is the first address?

It can be calculated by the bit-wise **AND** operation between the address and the corresponding address mask.

Address:	11001101	00010000	00100101	00100111	205.16.37.39
Mask:	**11111111**	**11111111**	**11111111**	**1111**0000	/28
First address:	11001101	00010000	00100101	00100000	**205.16.37.32**

Network Layer

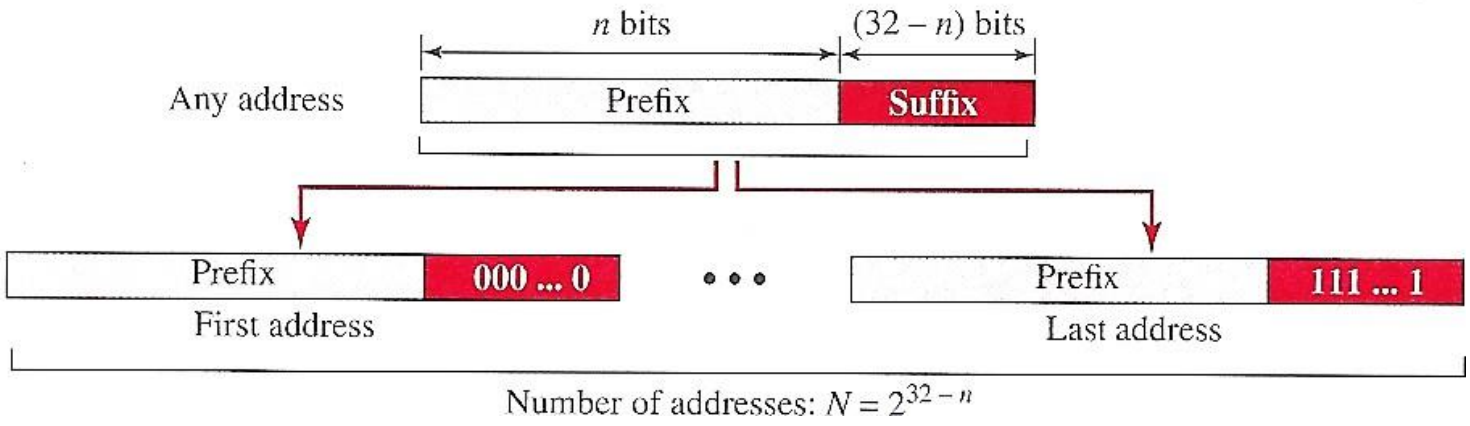
IPv4 addressing: CIDR – example

205.16.37.39/28

What is the last address?

It can be calculated by the bit-wise **OR** operation between the address and the **complement** of the address mask.

Address:	11001101	00010000	00100101	00100111	205.16.37.39
Comp(Mask):	**00000000**	**00000000**	**00000000**	**0000****1111**	comp(/28)
Last address:	11001101	00010000	00100101	00101111	**205.16.37.47**



Network Layer

IPv4 addressing: CIDR – example

205.16.37.39/28

How many addresses are there?

The number of addresses in the block is found as: where ***n*** is the mask (here 28)

$$2^{32-n} = 2^{32-28} = 16$$

or: the complement of the mask (in decimal) +1

Comp(Mask): **00000000 00000000 00000000 0000**1111 comp(/28) = 15

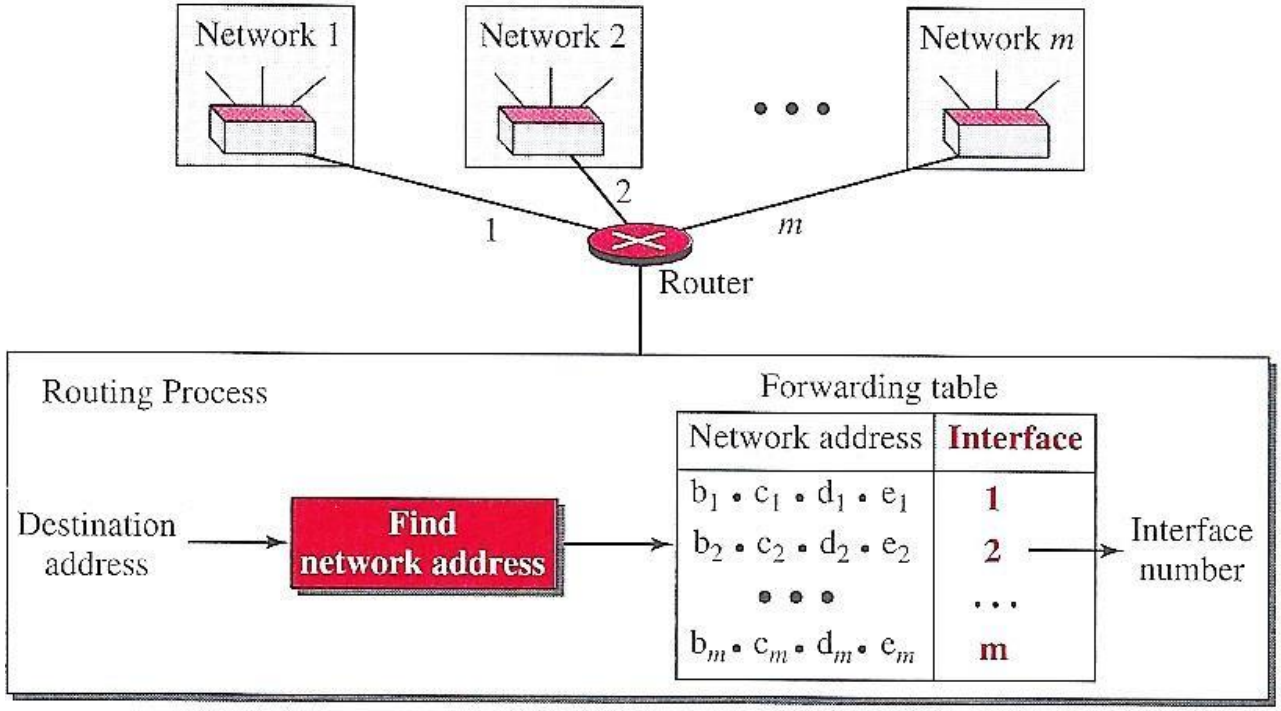
15 + 1 =**16**

Network Layer

IPv4 addressing: Network address

The first address in the network (block) is called **network address**, which is particularly important because it is used in routing a packet to its destination network.

Let us assume that the Internet consists of ***m*** networks and a router with ***m*** interfaces (ports). When a packet arrives at the router from any source host, the router needs to know which network the packet should be sent to: from which interface the packet should be sent out.



Once the network address is found, the forwarding table is consulted to find the interface number, from which the packet should be sent out.

Network Layer

IPv4 Addressing: Network Address - Subnetting

More levels of hierarchy can be created using subnetting. An organization (or an ISP) that has been assigned a large block may want to divide the block into different **subnetworks** or **subnets**.

- The outside world still view the organization as a single network, but internally there are several different subnets.
- All data is sent to a router connected to the rest of the Internet.
- The router also forward data to the relevant subnets.

The network block has one mask, and each subnet inside also has their own mask.

Network Layer

IPv4 addressing: Network address - Subnetting design

It is important that subnets are designed with care so that it is possible for a router to route packets to the relevant subnets.

Designing subnets

If an organization has been assigned N addresses (the prefix length is n) the block is divided into subnets, the assigned number of addresses in each subnetwork is N_{sub}. Then the following procedure should be followed:

- The number of addresses in each subnet must follow n-th power of 2 (1, 2, 4, …)
- The prefix length in each subnet is found as $n_{sub} = 32 - log_2 N_{sub}$
- The starting address of a subnet must be divisible by the number of addresses in the subnet

Note that this follows the Classless Block Division (ICANN) guidelines

Network Layer

IPv4 addressing: Network address - Subnetting design example

An organization has a given block **17.12.14.0/26**, which contains 64 addresses. The organization has three offices and needs to divide its block into three sub-networks of **32**, **16** and **16** addresses.

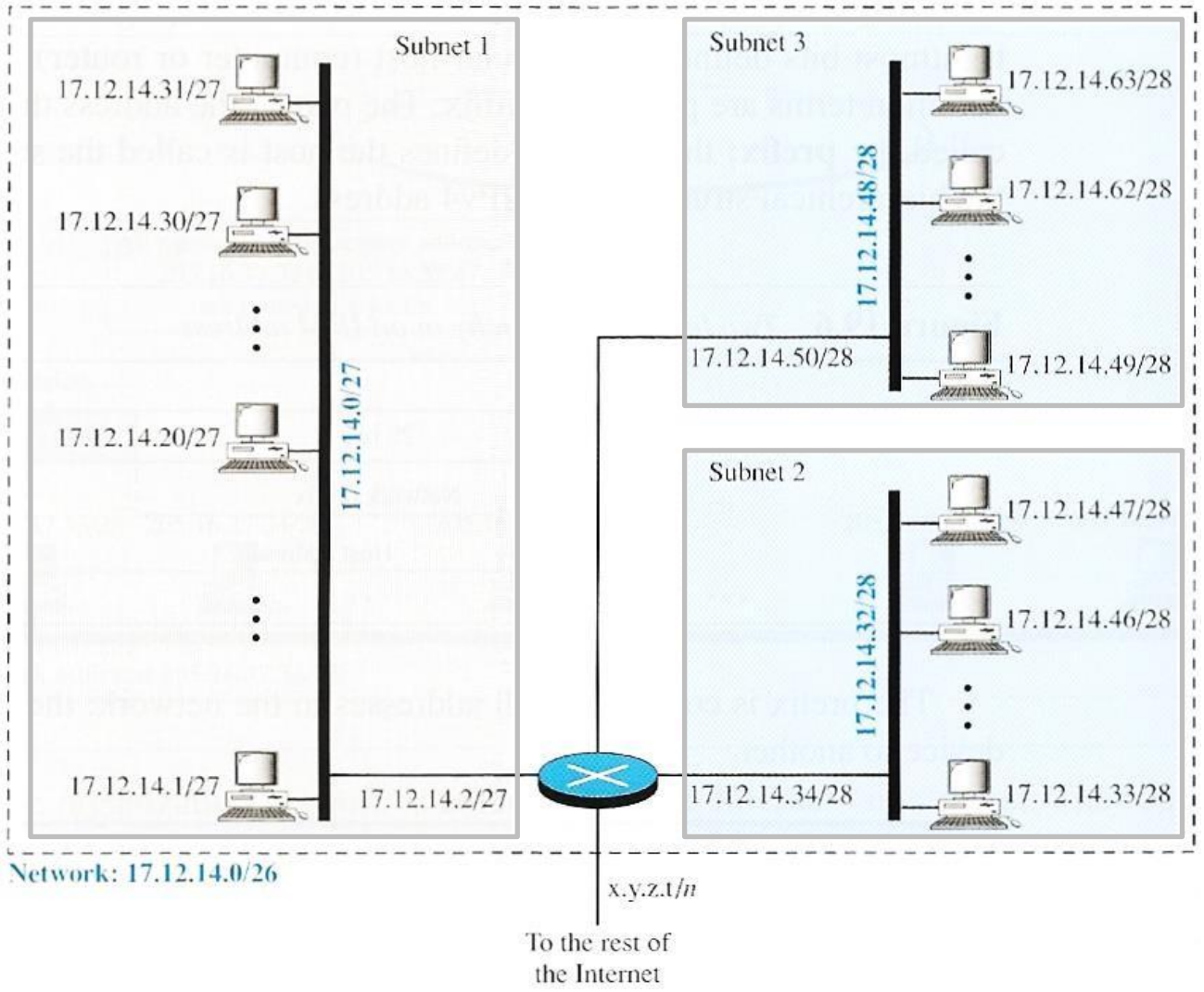
1. Assume that the prefix length of subnet 1 is ***n1***, then 2^{32-n1} must give 32, which means that ***n1 = 27***.
2. Assume that the prefix length of subnet 2 is ***n2***, then 2^{32-n2} must give 16, which means that ***n2 = 28***.
3. Assume that the prefix length of subnet 3 is ***n3***, then 2^{32-n3} must give 16, which means that ***n3 = 28***.

One could, according to the guidelines, have also done in another way:

1. Subnet 1: **n1 = 32 - $\log_2 32$**, which gives n1 = 32 – 5 **=27**
2. Subnet 1: **n2 = 32 - $\log_2 16$**, which gives n2 = 32 – 4 **=28**
3. Subnet 1: **n3 = 32 - $\log_2 16$**, which gives n3 = 32 – 4 **=28**

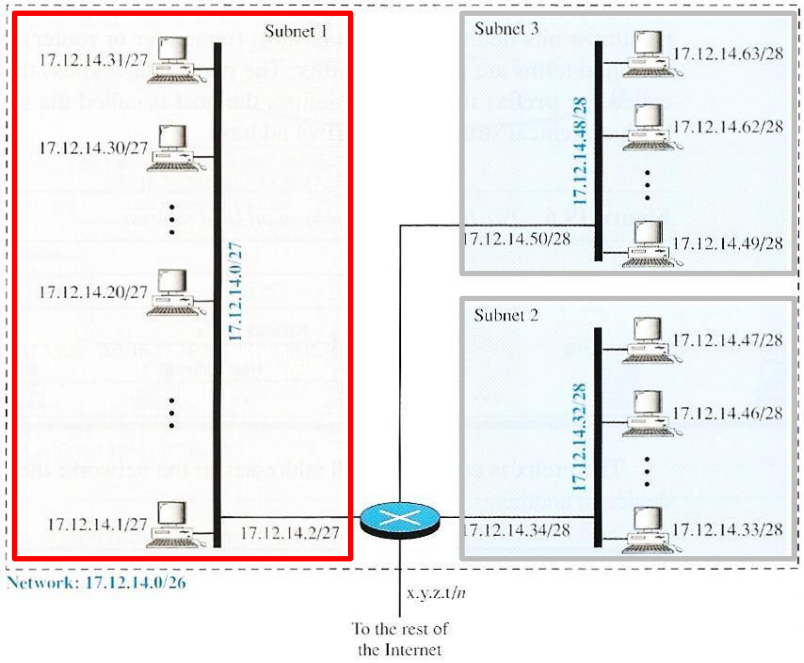
Network Layer

IPv4 addressing: Network address - Subnetting design example



Network Layer

IPv4 addressing: Network address - Subnetting design example

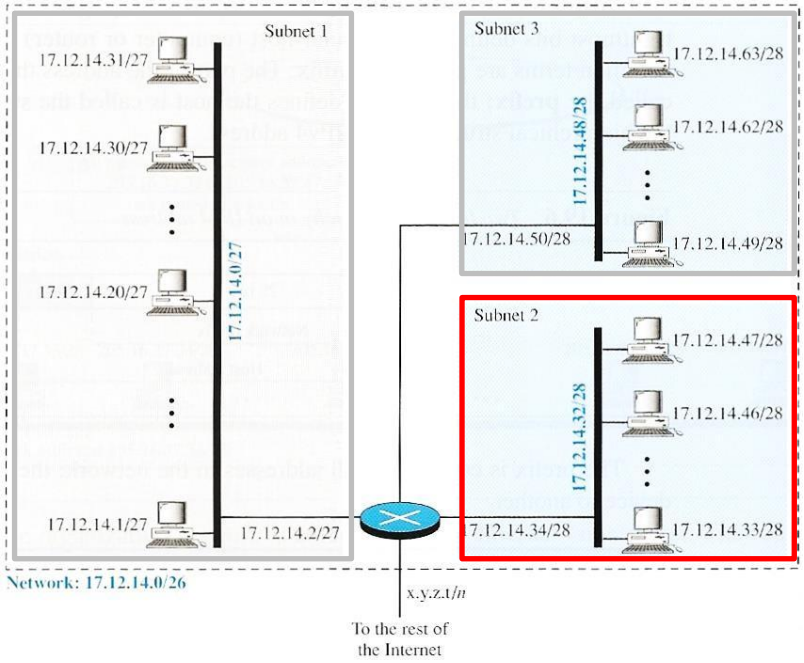


In subnet 1, e.g., **17.12.14.29/27** give us the subnet address if we use bit-wise AND operation with the mask **/27**.

Host:	00010001	00001100	00001110	00011101	17.12.14.29
Mask:	**11111111**	**11111111**	**11111111**	**11100000**	/27
Subnet 1:	00010001	00001100	00001110	00000000	**17.12.14.0**

Network Layer

IPv4 addressing: Network address - Subnetting design example

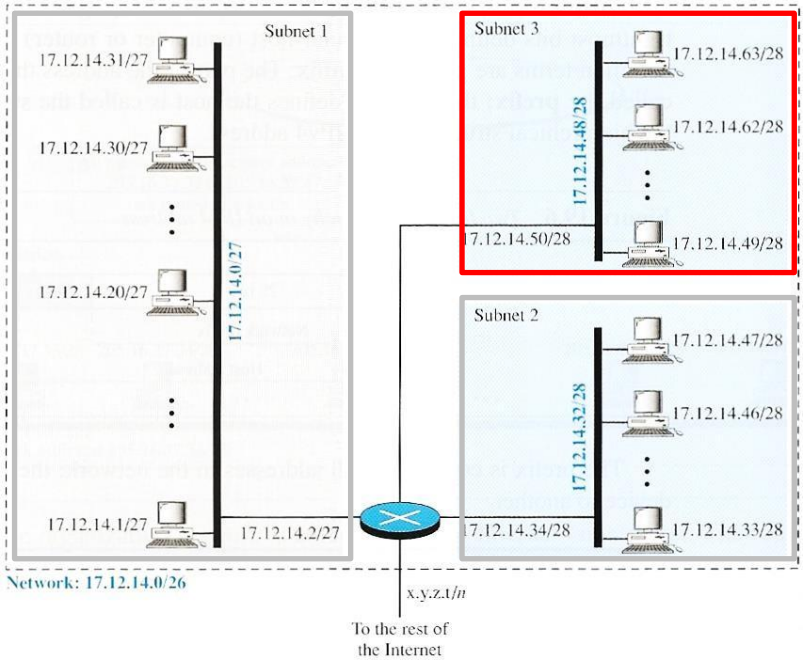


In subnet 2, e.g. **17.12.14.45/28** give us the subnet address if we use bit-wise AND operation with the mask **/28**.

Host:	00010001	00001100	00001110	00101101	17.12.14.45
Mask:	**11111111**	**11111111**	**11111111**	**1111**0000	/28
Subnet 2:	00010001	00001100	00001110	00100000	**17.12.14.32**

Network Layer

IPv4 addressing: Network address - Subnetting design example

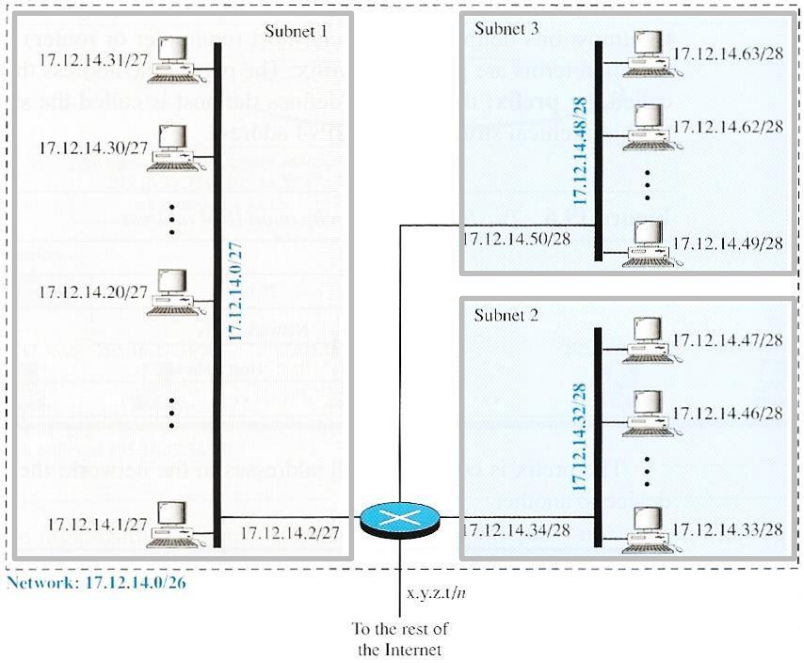


In subnet 3, e.g., **17.12.14.50/28** give us the subnet address if we use bit-wise AND operation with the mask **/28**.

Host:	00010001	00001100	00001110	00110010	17.12.14.50
Mask:	**11111111**	**11111111**	**11111111**	**1111**0000	/28
Subnet 3:	00010001	00001100	00001110	00110000	**17.12.14.48**

Network Layer

IPv4 addressing: Network address - Subnetting design example



Note that if we apply the mask **/26** to any address in the organization's network, we get the network address of the entire network: **17.12.14.0/26**.

Host:	00010001	00001100	00001110	00110010	17.12.14.50
Mask:	**11111111**	**11111111**	**11111111**	**11000000**	/26
Network:	00010001	00001100	00001110	00000000	**17.12.14.0**

Network Layer

IPv4 addressing: Network address - Subnetting design example

More information can be extracted with more bit-wise logical operations as introduced before

For subnet 1, the following applies:

Subnet 1

Address:	00010001	00001100	00001110	00000000	17.12.14.0
Mask n1:	**11111111**	**11111111**	**11111111**	**111**00000	/27
First address:	00010001	00001100	00001110	00000000	**17.12.14.0**

Note that the first address is also the subnet address (and that it is the same as the network address)

Address:	00010001	00001100	00001110	00000000	17.12.14.0
Comp(Mask):	**00000000**	**00000000**	**00000000**	**000**11111	comp(/27)
Last address:	00010001	00001100	00001110	00011111	**17.12.14.31**

1 is added to subnet 1's last address. This gives the subnet address of subnet 2

Network Layer

IPv4 addressing: Network address - Subnetting design example

Subnet 1

Last address:	00010001	00001100	00001110	00011111	**17.12.14.31**

For subnet 2, the following applies:

Subnet 2

First address:	00010001	00001100	00001110	00100000	**17.12.14.32**

The last address in subnet 2 can be found as follows:

Address:	00010001	00001100	00001110	00100000	**17.12.14.32**
Comp(Mask):	**00000000**	**00000000**	**00000000**	**0000**1111	comp(/28)
Last address:	00010001	00001100	00001110	00101111	**17.12.14.47**

1 is added to subnet 2's last address. This gives the subnet address of subnet 3

Network Layer

IPv4 addressing: Network address - Subnetting design example

Subnet 2

Last address:	00010001	00001100	00001110	00101111	**17.12.14.47**

For subnet 3, the following applies:

Subnet 3

First address:	00010001	00001100	00001110	00110000	**17.12.14.48**

The last address in subnet 3 can be found as follows:

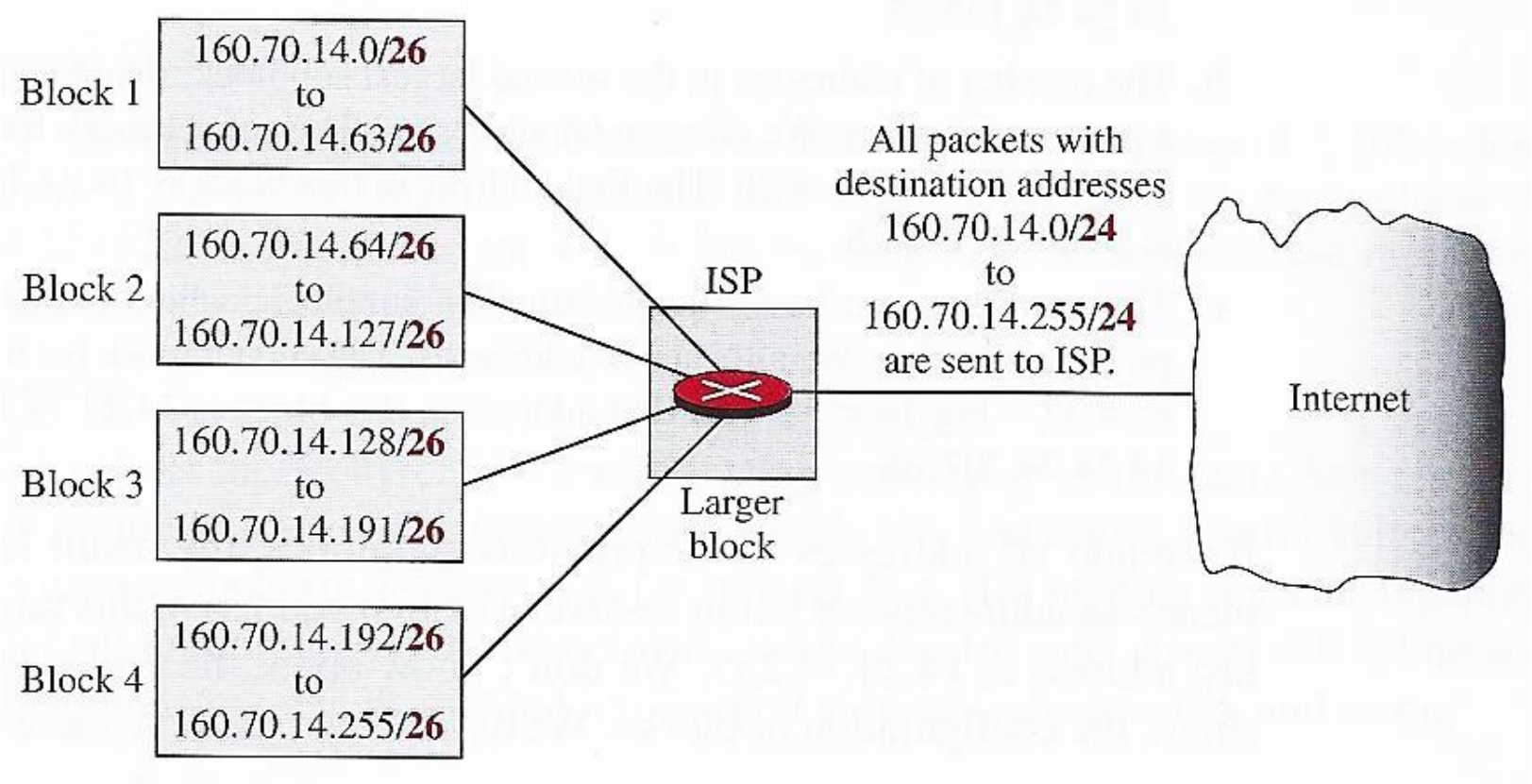
Address:	00010001	00001100	00001110	00110000	**17.12.14.48**
Comp(Mask):	**00000000**	**00000000**	**00000000**	**0000**1111	comp(/28)
Last address:	00010001	00001100	00001110	00111111	**17.12.14.63**

Network Layer

IPv4 Addressing: Network Address - Address Aggregation

One of the advantages of the CIDR strategy is **address aggregation**. When blocks of addresses are combined to create a larger block, routing can be done based on the prefix of the larger block.

All packets for the large block is sent to a router connected to the smaller blocks. This router (here ISP) then forward the packets to the right subnets.



Network Layer

IPv4 addressing: Special addresses

Before we end the topic of addresses in IPv4, we just need to mention some special addresses:

- **This-host address**
- **Limited broadcast address**
- **Loop-back address**
- **Private address (NAT)**
- **Multicast address**

This-host address: 0.0.0.0/32 this address is used when a host needs to send an IP datagram but does not know its own IP address to use as the source address.

Network Layer

IPv4 addressing: Special addresses

Before we end the topic of addresses in IPv4, we just need to mention some special addresses:

- **This-host address**
- **Limited-broadcast address**
- **Loop-back address**
- **Private address (NAT)**
- **Multicast address**

Limited-broadcast adressen: 255.255.255.255/32 this address is used when a router or host needs to send a datagram to all devices in a network.

Note that such a datagram can NOT leave the network!
Routers in the network block further transmission out of the network (hence the word ***Limited***)

Network Layer

IPv4 addressing: Special addresses

Before we end the topic of addresses in IPv4, we just need to mention some special addresses:

- **This-host address**
- **Limited-broadcast address**
- **Loop-back address**
- **Private address (NAT)**
- **Multicast address**

Loop-back address: 127.0.0.0/8 A packet with one of these (16,777,216) addresses never leaves its host!

The addresses are used for test purposes e.g., client/server programs in which one of the addresses in the block is used as the server address. In this way, we can test the programs using the same host to see if they work before running them on different computers.

Network Layer

IPv4 addressing: Special addresses

Before we end the topic of addresses in IPv4, we just need to mention some special addresses:

- **This-host address**
- **Limited-broadcast address**
- **Loop-back address**
- **Private address (NAT)**
- **Multicast address**

Private address (NAT): there are 4 blocks within this category:

- **10.0.0.0/8** (16.777.216 pcs.)
- **172.16.0.0/12** (1.048.576 pcs.)
- **192.168.0.0/16** (65.536 pcs.)
- **169.254.0.0/16** (65.536 pcs.)

These addresses are typically used **internally** on a network, with only a few IP addresses connected to the Internet. Thus, one can have many nodes connected to the network without using many (maybe only one) IP addresses externally to the Internet. The technique is called **N**etwork **A**ddress **T**ranslation (NAT) and is typically used in households.

Network Layer

IPv4 addressing: Special addresses

Before we end the topic of addresses in IPv4, we just need to mention some special addresses:

- **This-host address**
- **Limited-broadcast address**
- **Loop-back address**
- **Private address (NAT)**
- **Multicast address**

Multicast address: 224.0.0.0/4 (268.435.456 pcs.) these addresses are reserved for the multicast (one to many).

Network Layer

Dynamic **H**ost **C**onfiguration **P**rotocol (**DHCP**)

We have seen how ISPs can receive a large block of addresses from **ICANN** (**I**nternet **C**orporation for **A**ssigned **N**ames and **N**umbers).

We have seen how smaller organizations can receive smaller blocks of addresses from ISPs.

After a block of addresses are assigned to an organization, the network administration can manually assign addresses to the individual hosts or routers.

This can be done manually or automatically. If the automatic method is used, then it is done with **Dynamic Host Configuration Protocol (DHCP)**.

Network Layer

Dynamic **H**ost **C**onfiguration **P**rotocol (**DHCP**)

- **DHCP** is an application-layer program that works in the ***client-server*** paradigm and that helps TCP/IP at the network layer.

- **DHCP** has found widespread use in the Internet and is so popular that it is often called the ***plug-and-play*** protocol.

- **DHCP** is often used to assign permanent or temporary IP addresses to network users. (e.g., schools, etc.)

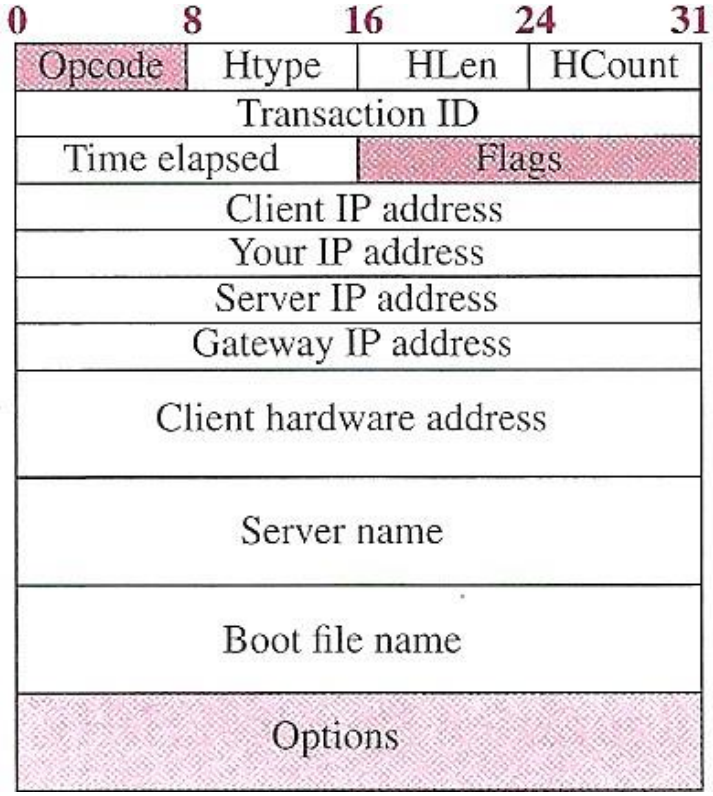
A computer connected to a network often needs the following information:

- **Its IP address**
- **Network Prefix** (or address mask)
- The address of the network's **default router** (so that it can communicate with other networks)
- The address of a **name server** (so names can be used instead of IP addresses)

DHCP can be used to collect the information.

Network Layer

DHCP message format



Fields:

Opcode: Operation code, request (1) or reply (2)
Htype: Hardware type (Ethernet, ...)
HLen: Length of hardware address
HCount: Maximum number of hops the packet can travel
Transaction ID: An integer set by the client and repeated by the server
Time elapsed: The number of seconds since the client started to boot
Flags: First bit defines unicast (0) or multicast (1); other 15 bits not used
Client IP address: Set to 0 if the client does not know it
Your IP address: The client IP address sent by the server
Server IP address: A broadcast IP address if client does not know it
Gateway IP address: The address of default router
Server name: A 64-byte domain name of the server
Boot file name: A 128-byte file name holding extra information
Options: A 64-byte field with dual purpose described in text

The option field has two purposes.

- Additional information can be found here
- Some specific vendor information

Network Layer

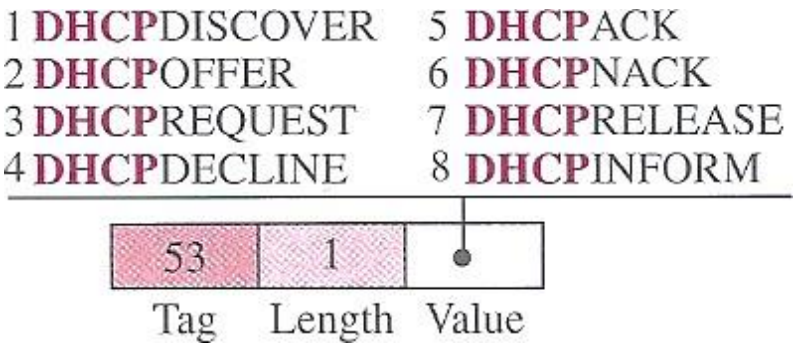
DHCP message format

The server uses a number, called a ***magic cookie***, the number is in the format of IP address and has the value **99.130.83.99**.

When a client reads a DHCP message, it looks for that number.
If it exists, then it knows that the next 60 bytes are options. These options have the format:

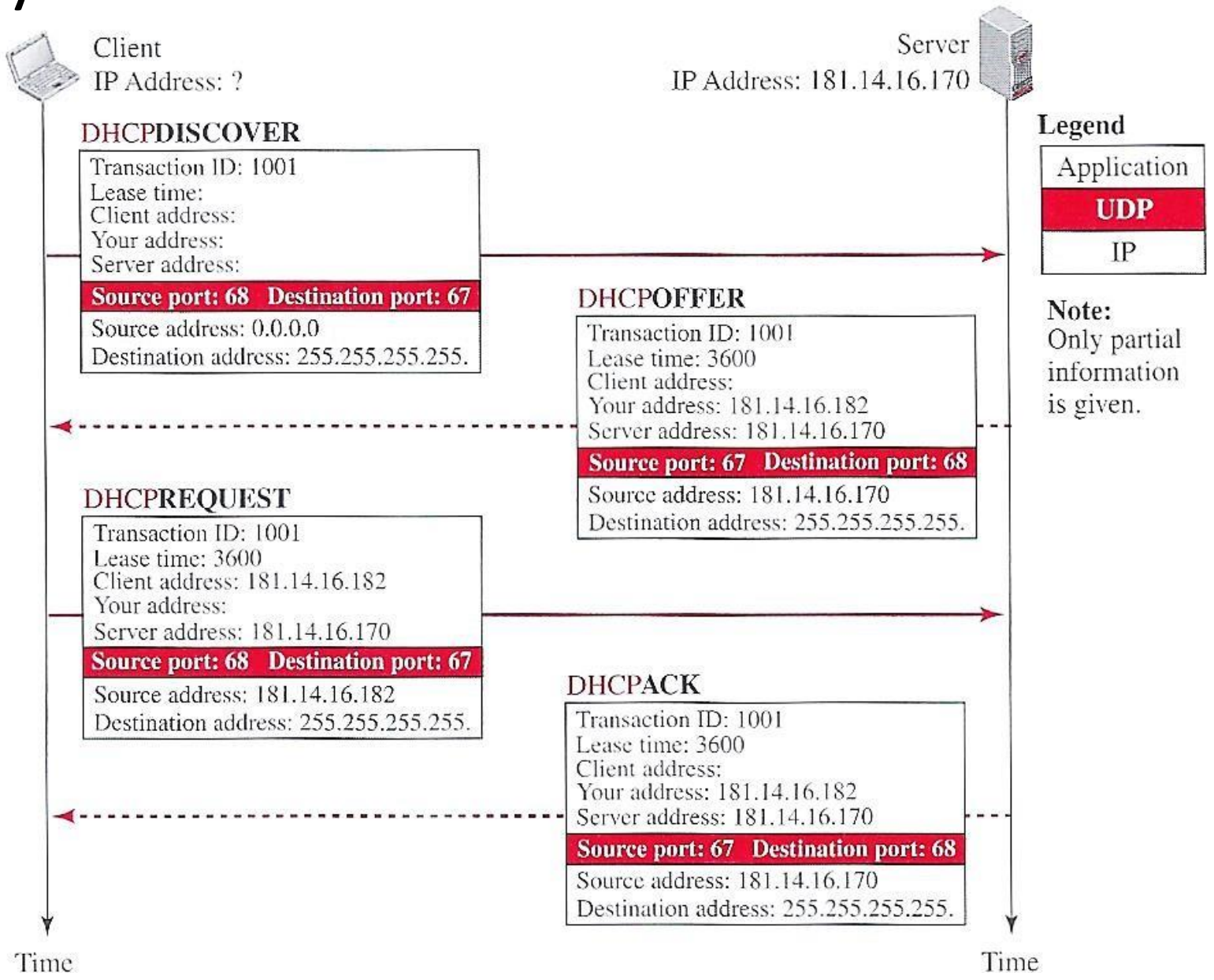
- **1-byte tag-field**
- **1-byte length-field**
- a **variable length value-field**

If the tag-field has the value **53**, then the value field will define one of 8 different packet types:



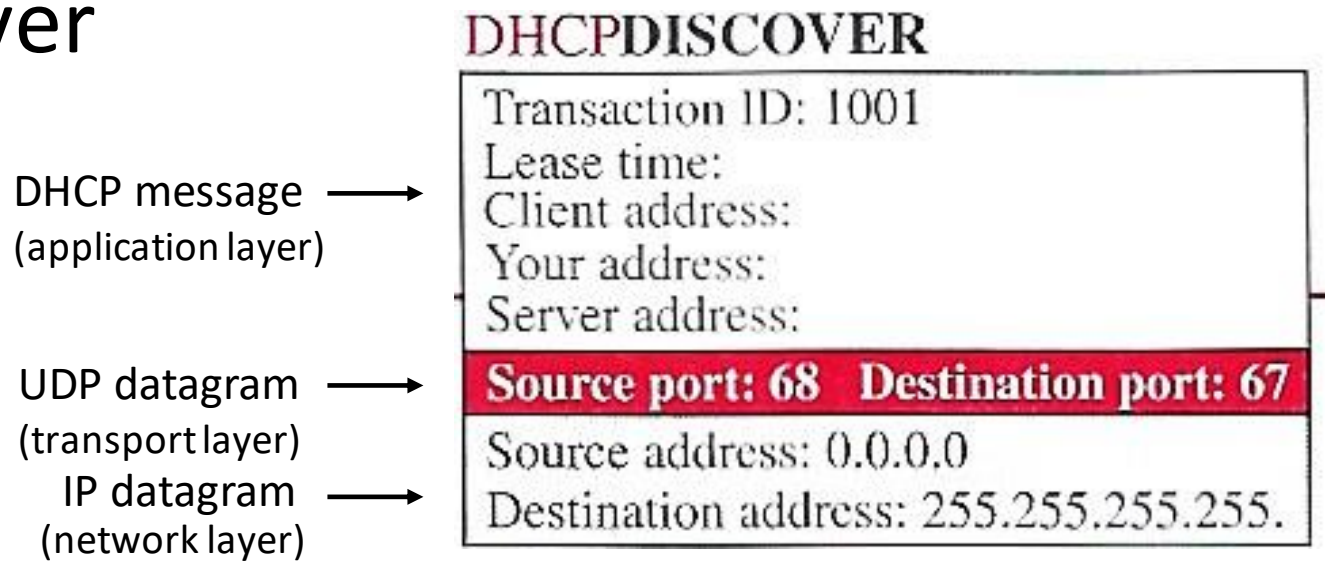
Network Layer

DHCP example



Network Layer

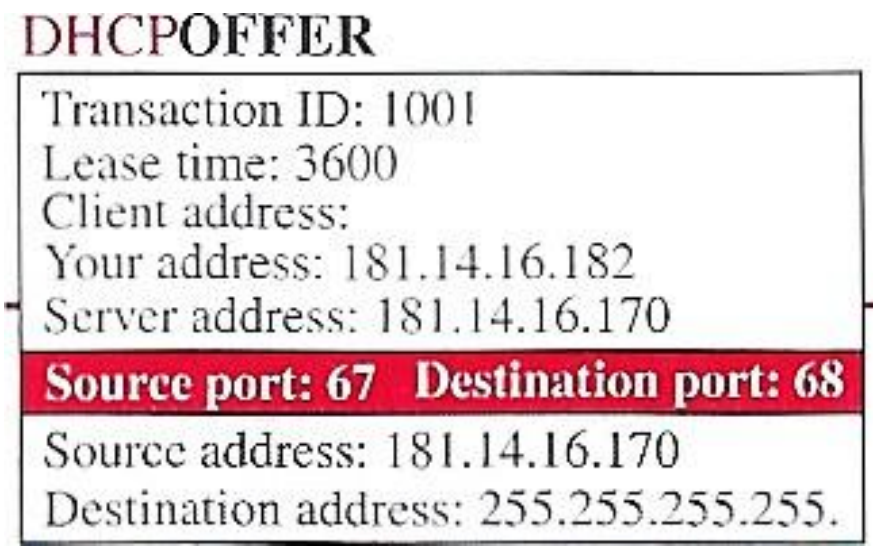
DHCP example



1. First, the client wants to connect to the network by sending a DHCPDISCOVER message. Note that the message only contains **transaction ID** (random). The message is encapsulated in a **UDP** (**U**ser **D**atagram **P**rotocol) datagram with port numbers **68** and **67** for source and destination (these are well-known port numbers). The user datagram is further encapsulated in an IP datagram with the source address set to 0.0.0.0 ("**this host**") and the destination address set to 255.255.255.255 (**broadcast address**). The reason is that the joining host knows neither its own address nor the server address.

Network Layer

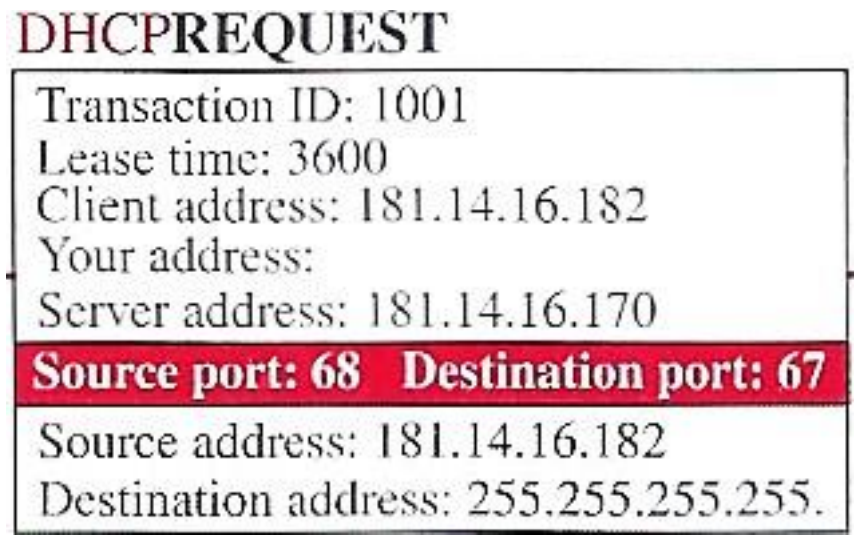
DHCP example



2. A DHCP server (there can be more than one) responds with a DHCPOFFER message. Here the client is offered an IP address and a lease time. The server address field includes the IP address of the server. The message is broadcast as the server allows other DHCP servers to receive the offer and give a better offer if they can.

Network Layer

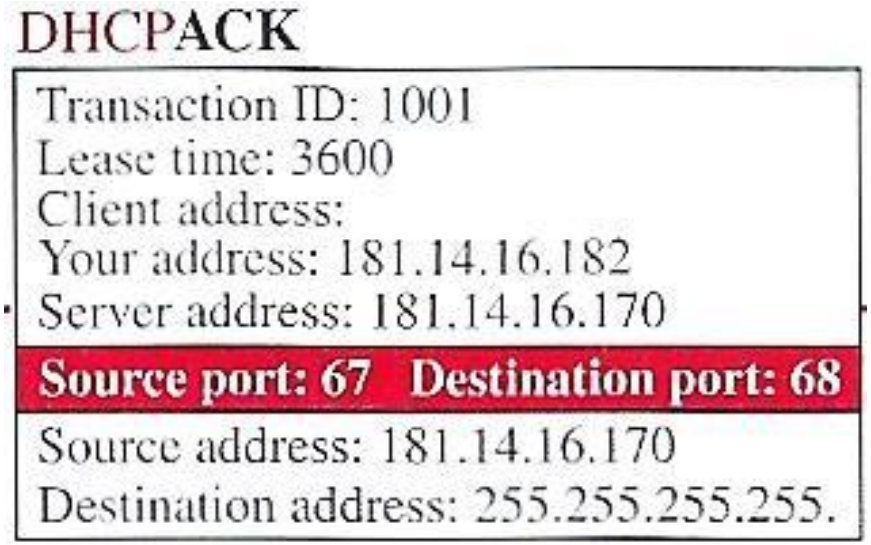
DHCP example



3. The client who may have received more than one offer selects the best and sends a DHCPREQUEST message. Note that the client has entered the offered IP address as its own sender address. The destination address is still set to the broadcast address to let the other servers know that their offers were not accepted.

Network Layer

DHCP example



4. The DHCP server acknowledges with a DHCPACK message.
   This completes the connection procedure.
   If for some reason the server still cannot keep its offer, then a DHCPNACK would have been sent and the client would have to repeat the procedure. The message is broadcast to let other servers know that the request is accepted or rejected.

The DHCP server does not send all information to the client. However, the DHCPACK message specifies a path to a file that contains all the information that could be requested (for example, the address of DNS server, etc.). The client can use **FTP** (**F**ile **T**ransfer **P**rotocol) to retrieve the rest of the needed information.

Network Layer

DHCP state diagram

Error control

DHCP uses the services of UDP, which is not a reliable protocol!

To add error control, DHCP does the following:

- DHCP requires UDP to use checksum, which is optional in UDP.
  (We will look at UDP later in the course)
- The DHCP client applies a timer and re-transmission policy
  if it does not receive the DHCP reply to a request.