# Flooding

- A **simple** local technique is **flooding, in which every incoming packet is sent out on every** outgoing line except the one it arrived on.
- Flooding is a simple algorithm to send a packet along all paths.
- Flooding obviously generates many numbers of **duplicate packets**, in fact, an **infinite** number unless some measures are taken to damp the process.

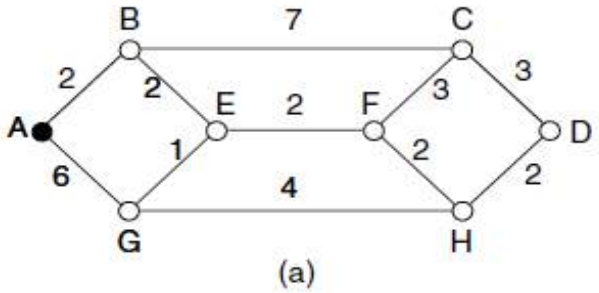
# Flooding routing control

- One such measure is to have a **hop counter** contained in the header of each packet that is decremented at each hop, with the packet being discarded when the counter reaches zero.
- Ideally, the hop counter should be **initialized to the length** of the path from source to destination.

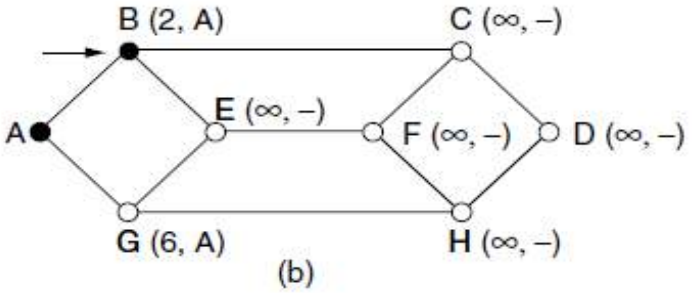
# Shortest path vector routing

- Shortest path routing first developed by E. W. **Dijkstra** algorithm.
- Find the shortest path from a specified source to all other destinations in the network.

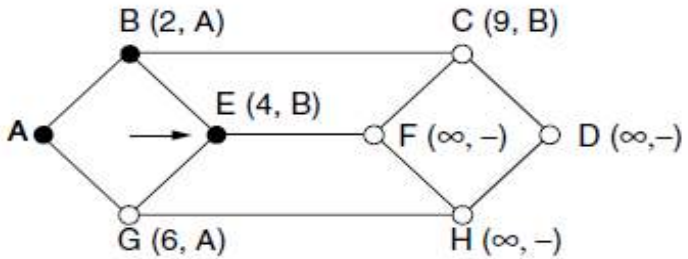
# The first six steps used in computing the shortest path from *A to D.*



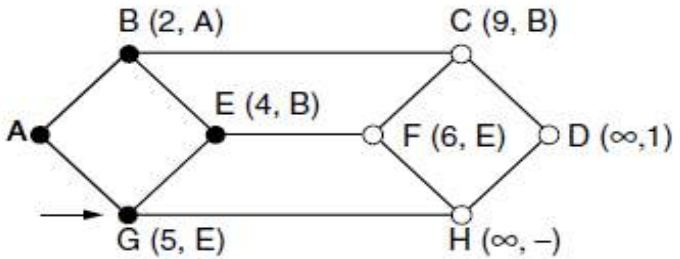
(a)



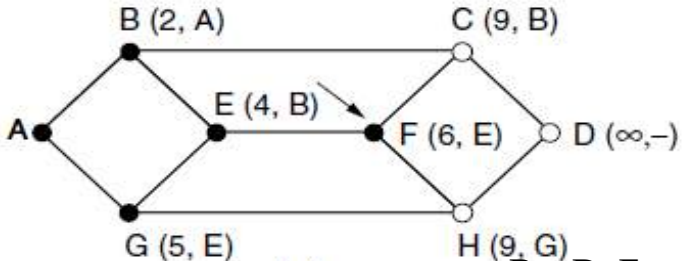
(b)



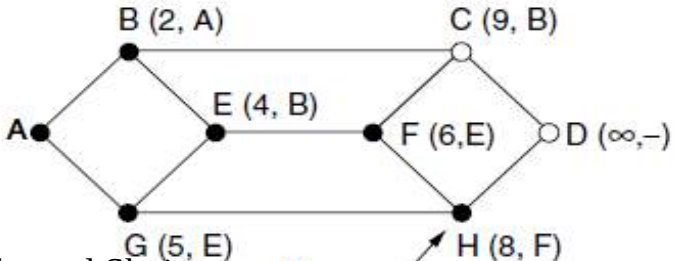
(c)



(d)



(e)



(f)

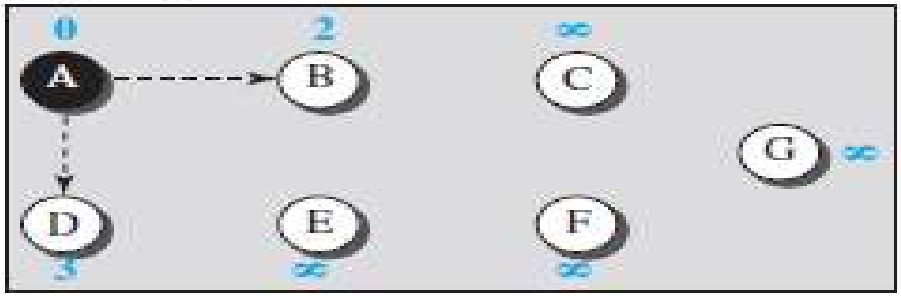
# Dynamic Routing Algorithm

- Distance Vector Routing.
- Link state routing.

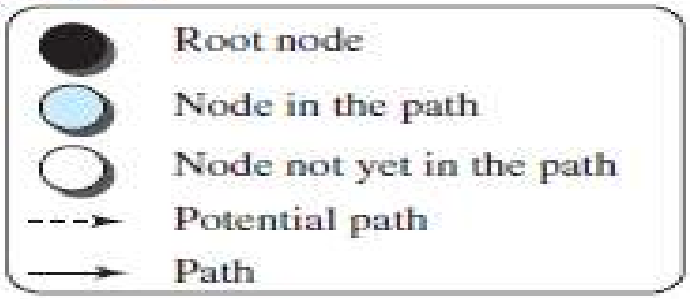
# Distance Vector Routing

- Distance Vector routing is **intra-domain** protocols, **inside** Autonomous system, but not between Autonomous system.
- distance-vector routing are based on the **least-cost** goal.
- Distance Vector developed by **Bellman-Ford** algorithm.
- Bellman **equation** is used to find the **least** cost (**shortest distance**) between a source to destination.

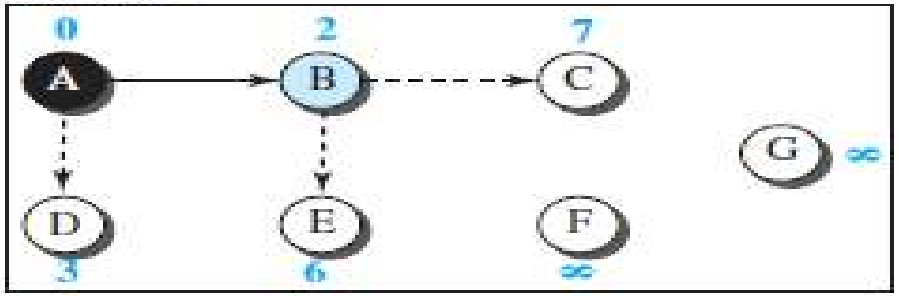
Initialization



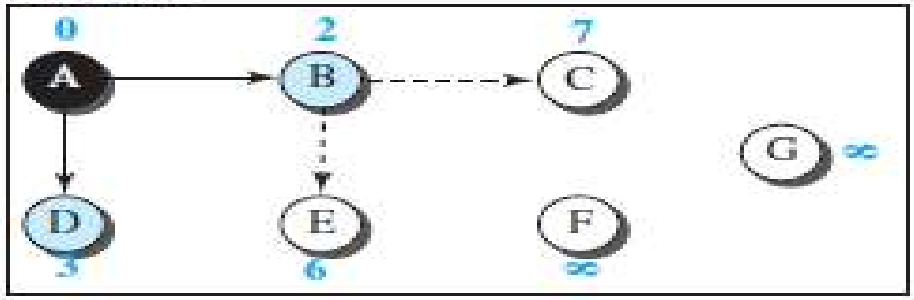

Legend



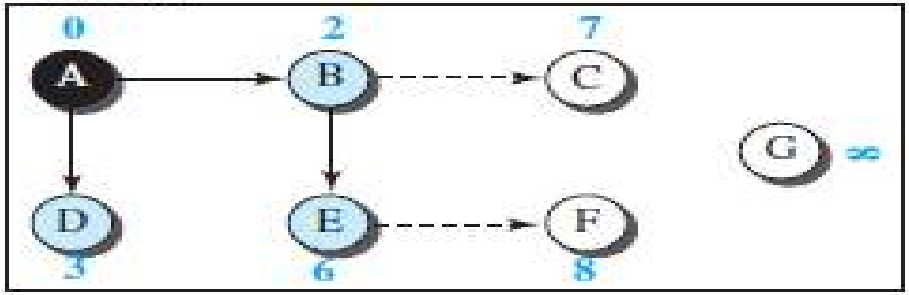

Iteration 1



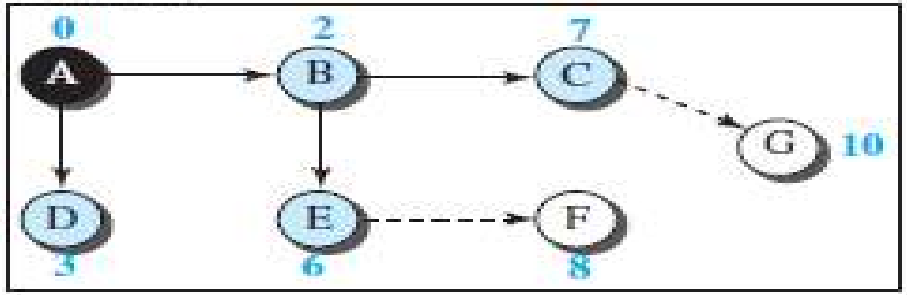

Iteration 2



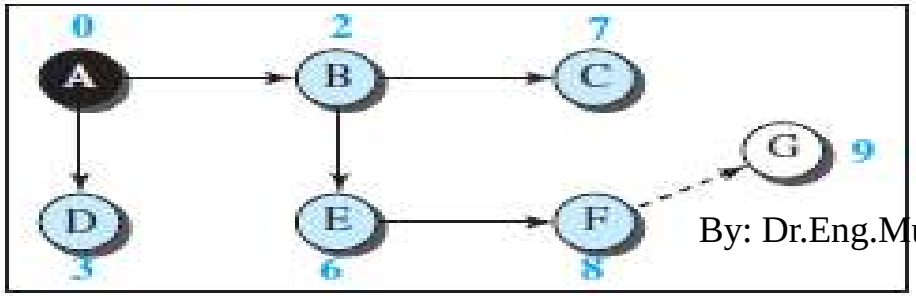

Iteration 3



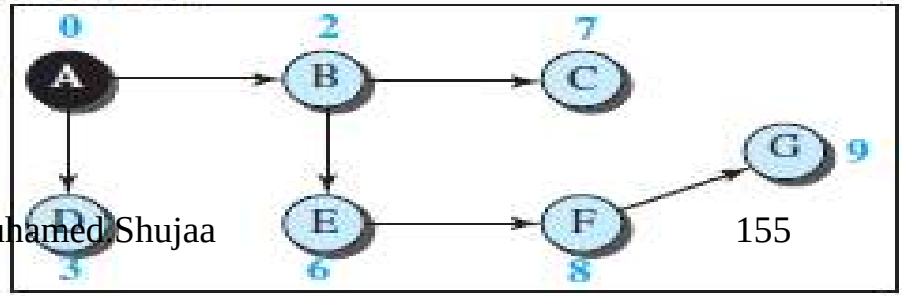

Iteration 4




Iteration 5



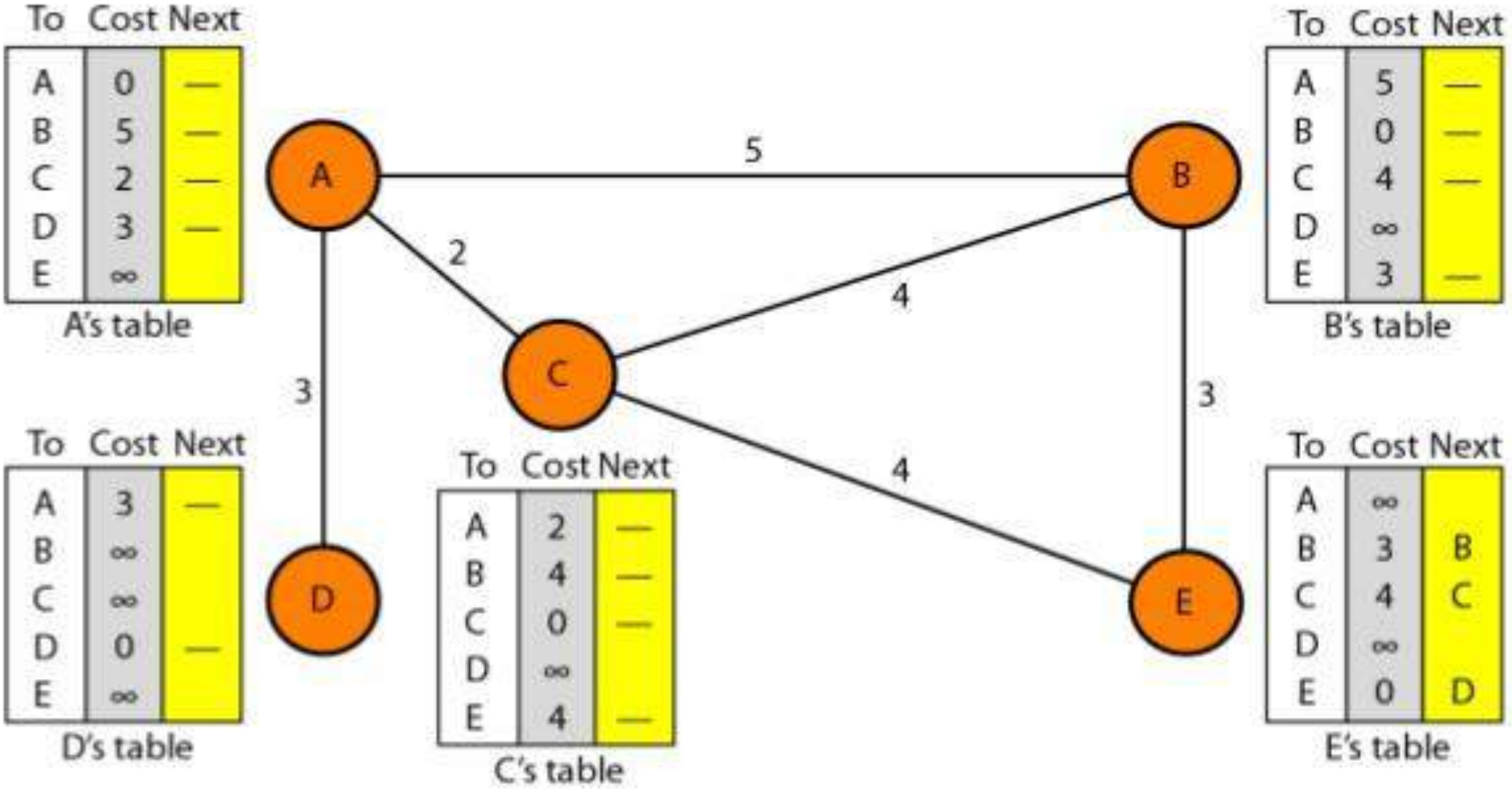

Iteration 6



# Distance….

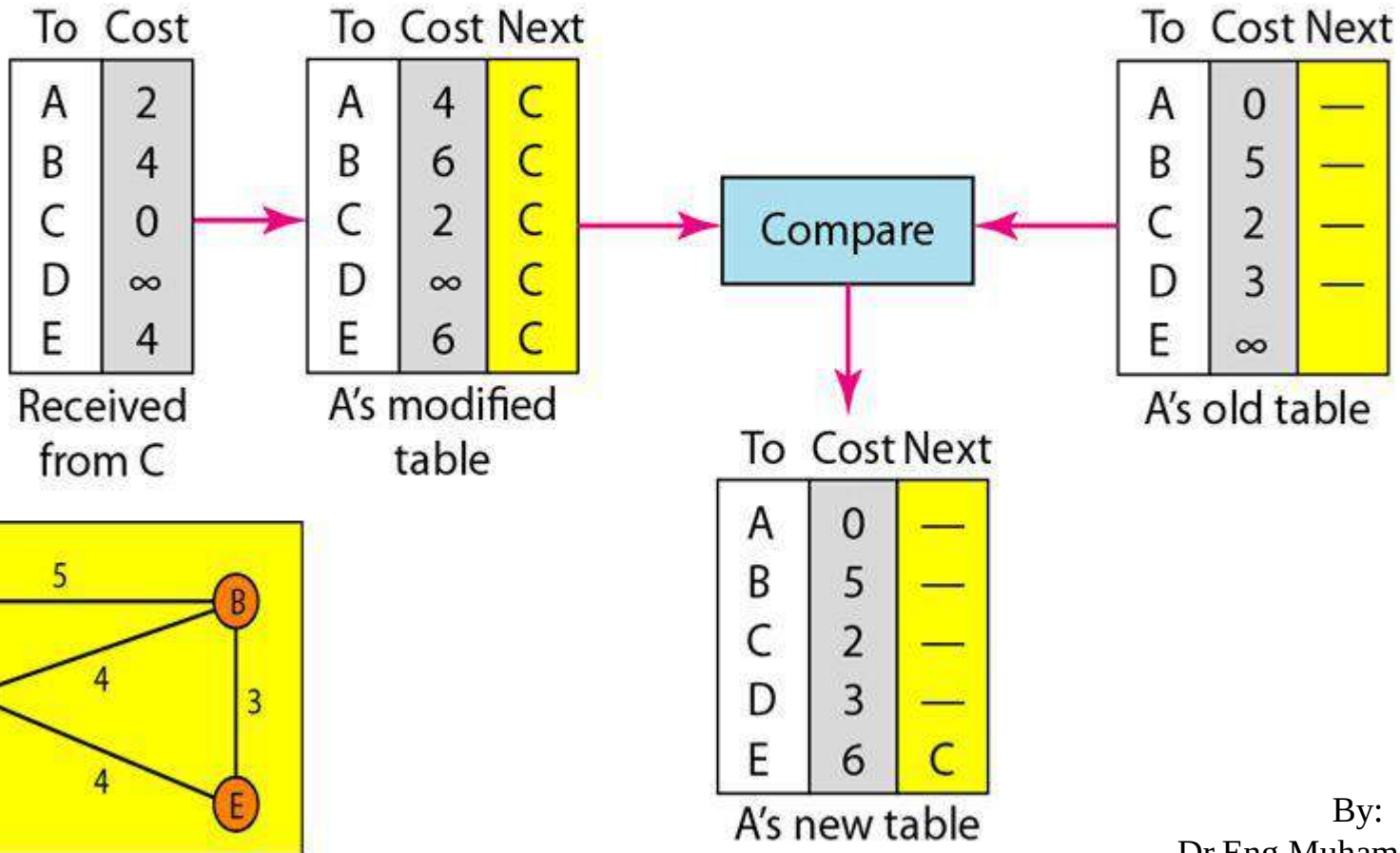
- A **distance vector routing algorithm** operates by having each router **maintain a table** (i.e., a **vector**) giving the best known distance to each destination and which link to use to get there.
- These tables are **updated by exchanging** information with the neighbors router. Every router knows the **best link** to reach each destination.

# *Initialization of tables in distance vector routing (DVR)*



# Distance vector routing

## Sharing: Updating in distance vector routing







By:
Dr.Eng.Muhamed.Shujaa
158

# Distance vector specifications

- RIP based on distance vector routing, each router **shares**, **at regular intervals**, its knowledge about entire AS with its neighbor.
- Distance vector routing work properly theoretically but it has serious **problem practically**.
- It is so **slow** and does not take **Bandwidth** into consideration when choose the root.

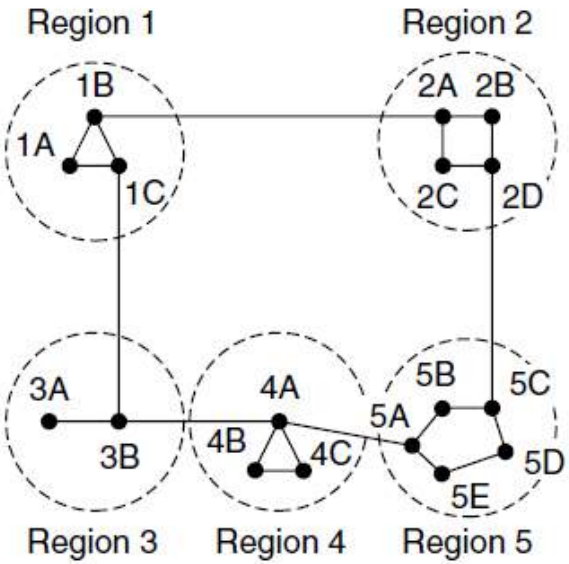
# Hierarchical Routing

- As networks grow in size, the router routing tables grow proportionally.
- Not only is router memory consumed by ever-increasing tables, but more CPU time is needed to scan them and more bandwidth is needed to send status reports about them.
- So router can not have table about the entire network.

# Hierarchical Routing

- When hierarchical routing is used, the routers are divided into what we will call **regions.**
- Each router knows all the details about how to route packets to destinations within its **own region** but knows nothing about the internal structure of other regions.(ie EGP protocol)

# Hierarchical routing.




(a)

Full table for 1A

| Dest. | Line | Hops |
|---|---|---|
| 1A | – | – |
| 1B | 1B | 1 |
| 1C | 1C | 1 |
| 2A | 1B | 2 |
| 2B | 1B | 3 |
| 2C | 1B | 3 |
| 2D | 1B | 4 |
| 3A | 1C | 3 |
| 3B | 1C | 2 |
| 4A | 1C | 3 |
| 4B | 1C | 4 |
| 4C | 1C | 4 |
| 5A | 1C | 4 |
| 5B | 1C | 5 |
| 5C | 1B | 5 |
| 5D | 1C | 6 |
| 5E | 1C | 5 |

(b)

Hierarchical table for 1A

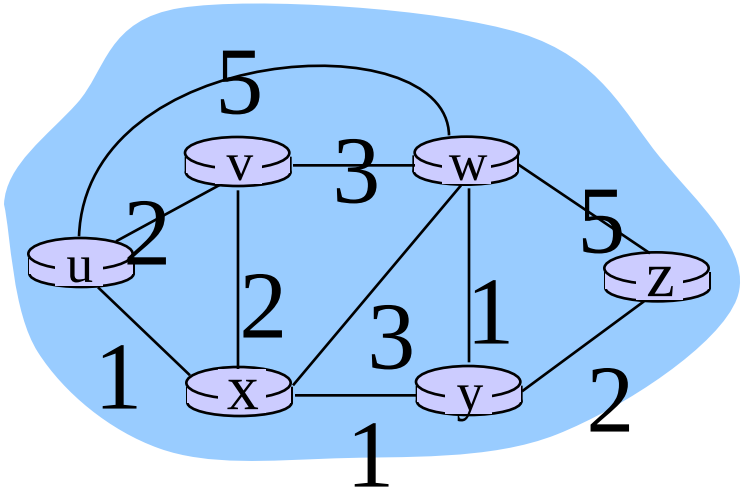
| Dest. | Line | Hops |
|---|---|---|
| 1A | – | – |
| 1B | 1B | 1 |
| 1C | 1C | 1 |
| 2 | 1B | 2 |
| 3 | 1C | 2 |
| 4 | 1C | 3 |
| 5 | 1C | 4 |

(c)



By:
Dr.Eng.Muhamed.Shujaa

# Routing Graph abstraction



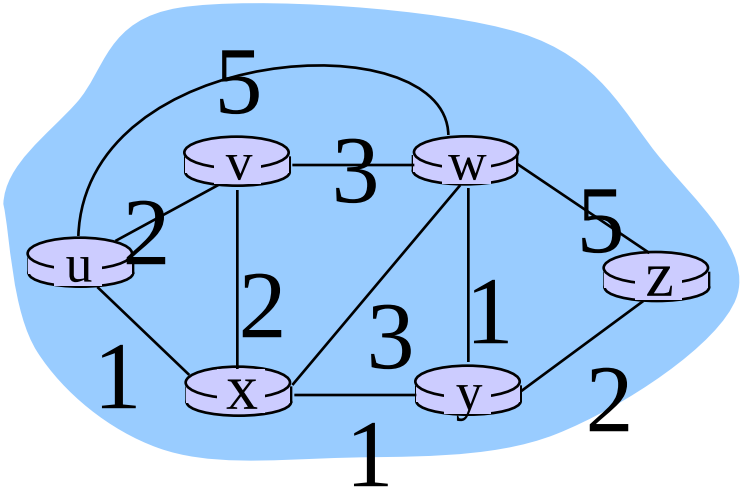

graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

*aside:* graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

# Graph abstraction: costs




c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

*key question:* what is the least-cost path between u and z ?
*routing algorithm:* algorithm that finds that least cost path

# Distance vector algorithm

*Bellman-Ford equation (dynamic programming)*

let

$d_x(y)$ := cost of least-cost path from x to y

Then

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

$d_v(y)$: cost from neighbor v to destination y

$c(x,v)$: cost to neighbor v

$\min_v$: *min* taken over all neighbors v of x

# Distance vector algorithm

- $D_x(y)$ = estimate of least cost from x to y

  x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$

- node x:
  - knows cost to each neighbor v: $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

# Distance vector algorithm

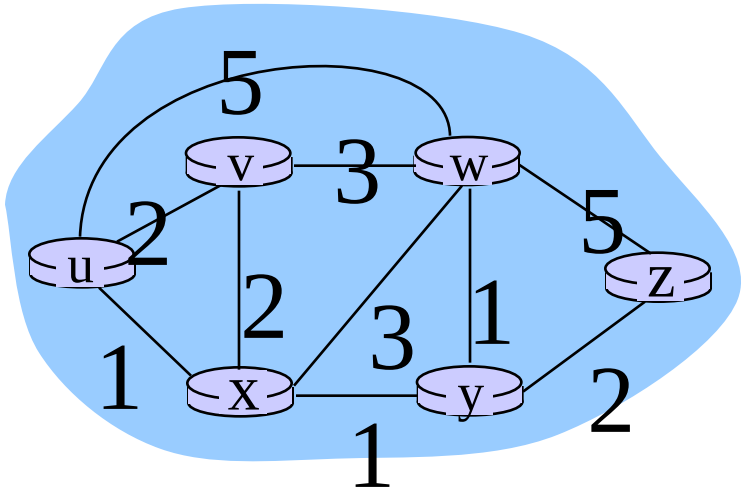
*key idea:*

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

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

- under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

# Bellman-Ford example




clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

node achieving minimum is next hop in shortest path, used in forwarding table

# A Link-State Routing Algorithm

***Dijkstra's algorithm***

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives *forwarding table* for that node
- iterative: after k iterations, know least cost path to k dest.' s

*notation:*

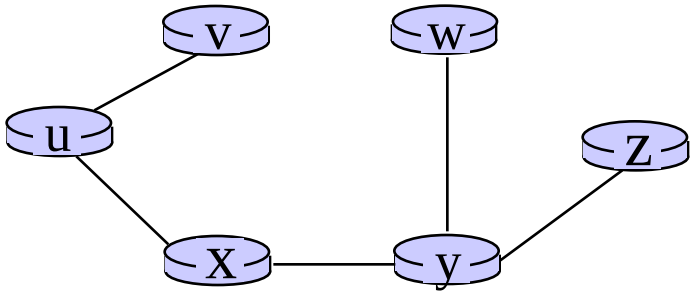
- c(x,y): link cost from node x to y; $= \infty$ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

# Dijsktra's Algorithm

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

# Dijkstra's algorithm: example

resulting shortest-path tree from u:

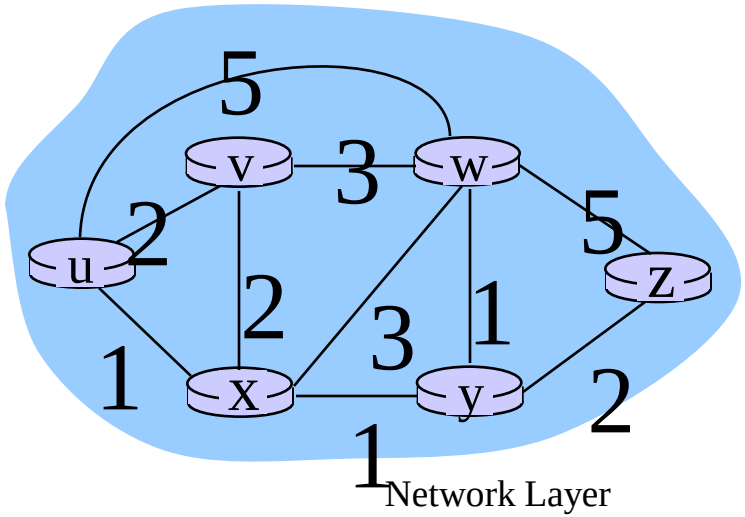



resulting forwarding table in u:

| destination | link |
|---|---|
| v | (u,v) |
| x | (u,x) |
| y | (u,x) |
| w | (u,x) |
| z | (u,x) |

# Dijkstra's algorithm: example

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



# Dijkstra's algorithm, discussion

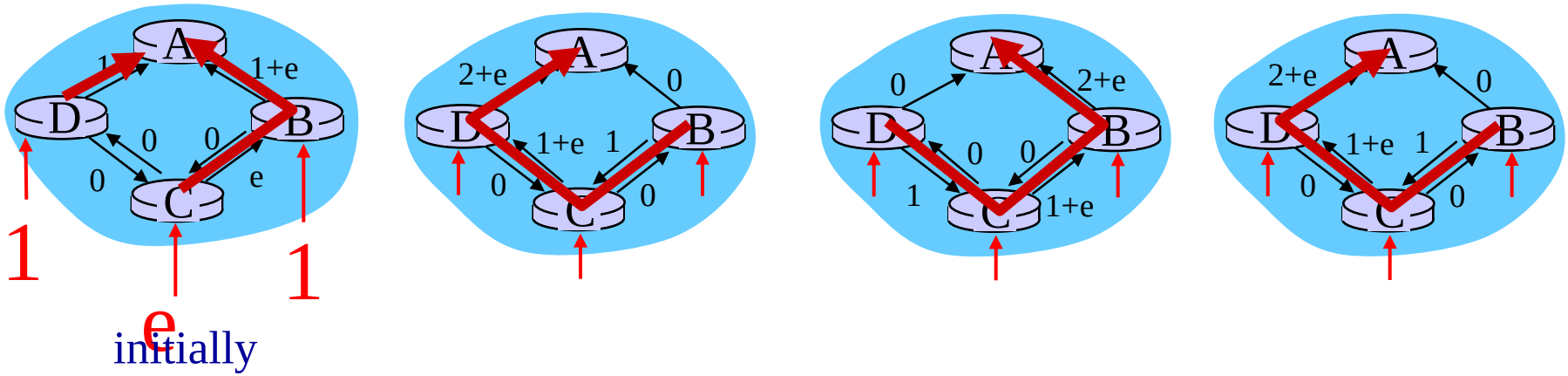
*algorithm complexity:* n nodes

- each iteration: need to check all nodes, w, not in N
- n(n+1)/2 comparisons

*oscillations possible:*

- e.g., support link cost equals amount of carried traffic:

given these costs, find new routing resulting in new costs



# Comparison of LS and DV algorithms

*message complexity*

- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***DV:*** exchange between neighbors only
  - convergence time varies

*speed of convergence*

- ***LS:*** $O(n^2)$ algorithm requires O(nE) msgs
  - may have oscillations
- ***DV:*** convergence time varies
  - may be routing loops
  - count-to-infinity problem

*robustness:* what happens if router malfunctions?

*LS:*

- node can advertise incorrect *link* cost
- each node computes only its *own* table

*DV:*

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

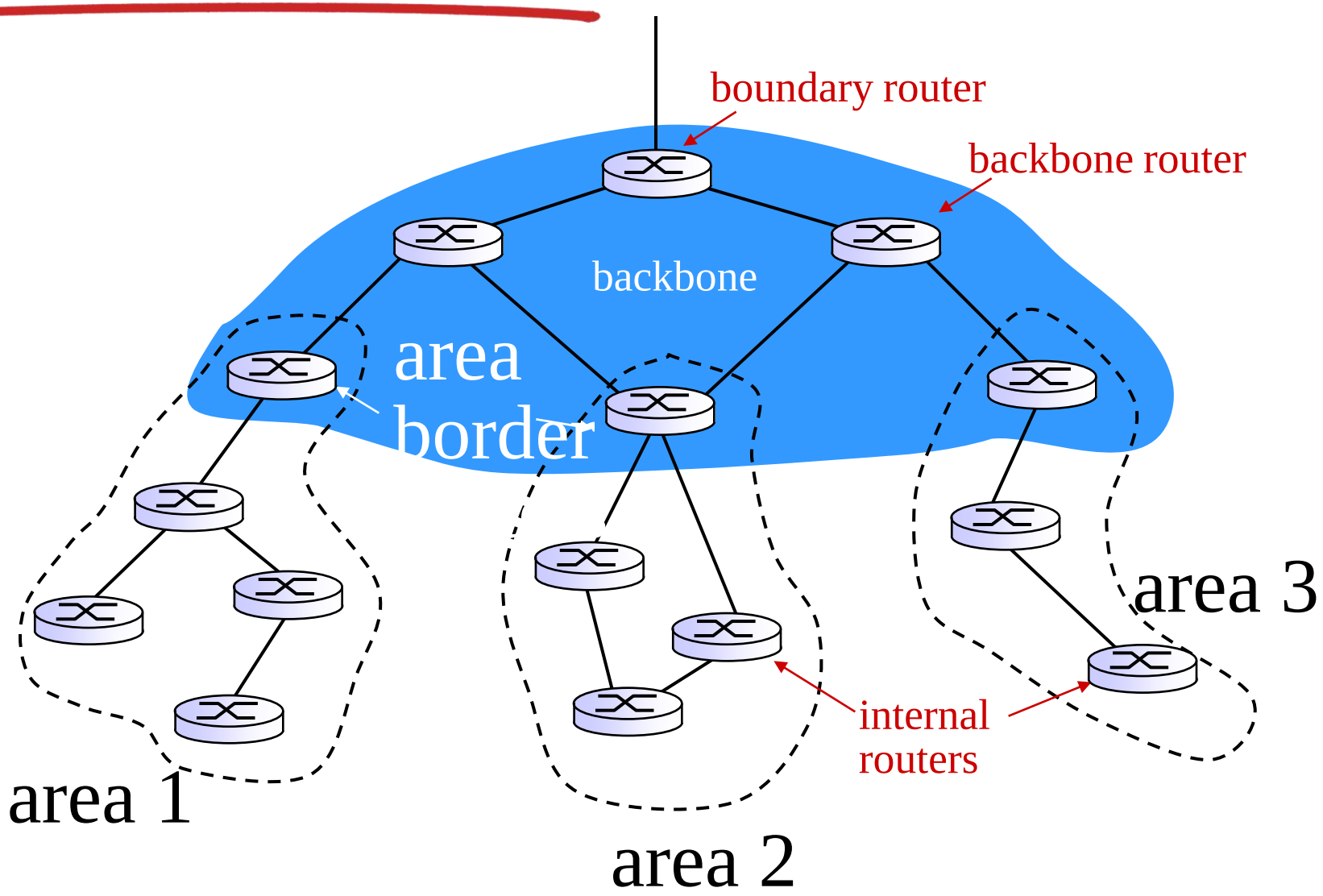
# OSPF (Open Shortest Path First)

- "open": publicly available
- uses link state algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbor
- advertisements flooded to *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP

# OSPF "advanced" features (not in RIP)

- *security:* all OSPF messages authenticated (to prevent malicious intrusion)
- multiple same-cost paths allowed (only one path in RIP)
- for each link, multiple cost metrics for different TOS (e.g., satellite link cost set "low" for best effort ToS; high for real time ToS)
- integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- hierarchical OSPF in large domains.

# Hierarchical OSPF



# Hierarchical OSPF

- *two-level hierarchy:* local area, backbone.
  - link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- *area border routers:* "summarize" distances to nets in own area, advertise to other Area Border routers.
- *backbone routers:* run OSPF routing limited to backbone.
- *boundary routers:* connect to other AS's.

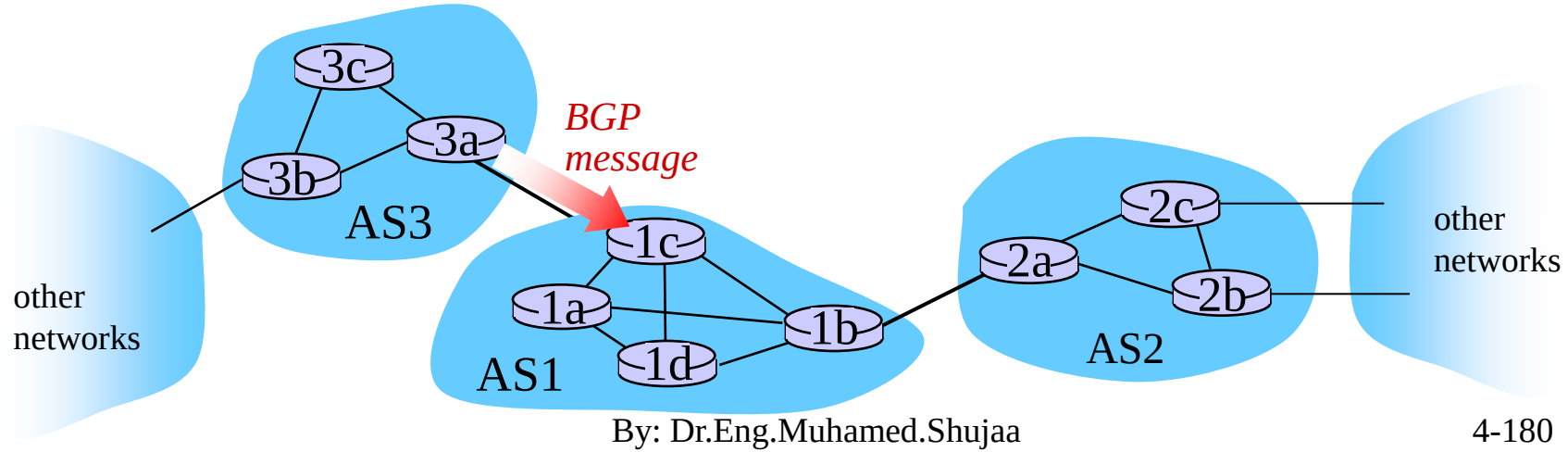
# Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- BGP provides each AS a means to:
  - eBGP: obtain subnet reach ability information from neighboring ASs.
  - iBGP: propagate reach ability information to all AS-internal routers.
  - determine "good" routes to other networks based on reach ability information and policy.
- allows subnet to advertise its existence to rest of Internet: *"I am here"*

# BGP basics

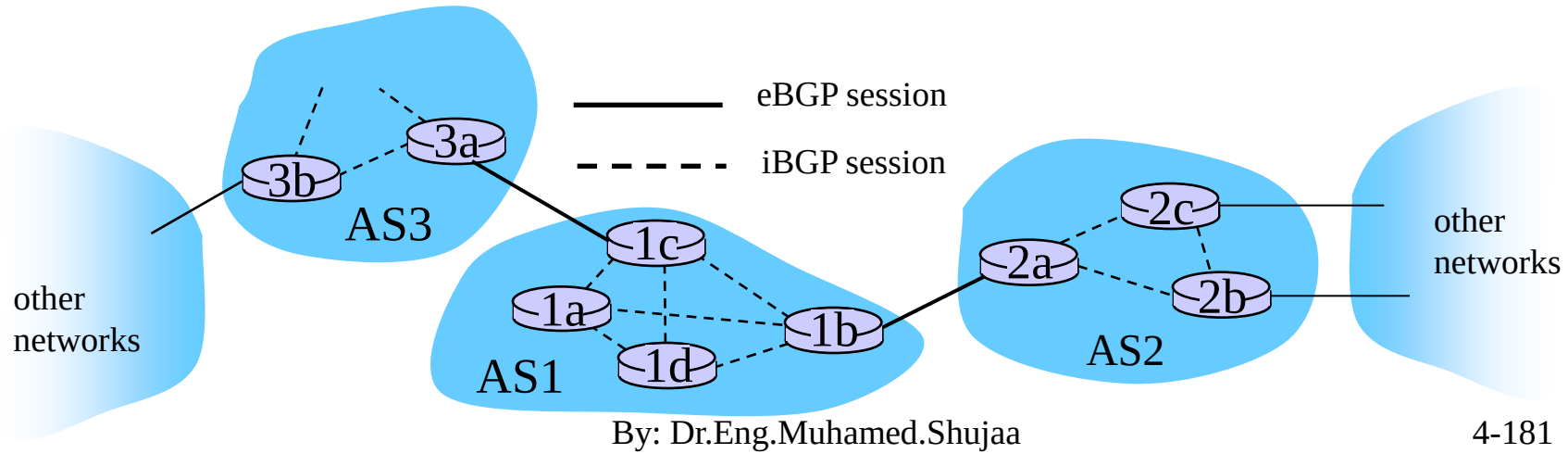
- BGP session: two BGP routers ("peers") exchange BGP messages:
  - advertising *paths* to different destination network prefixes ("path vector" protocol)
  - exchanged over semi-permanent TCP connections

- when AS3 advertises a prefix to AS1:
  - AS3 *promises* it will forward datagrams towards that prefix
  - AS3 can aggregate prefixes in its advertisement



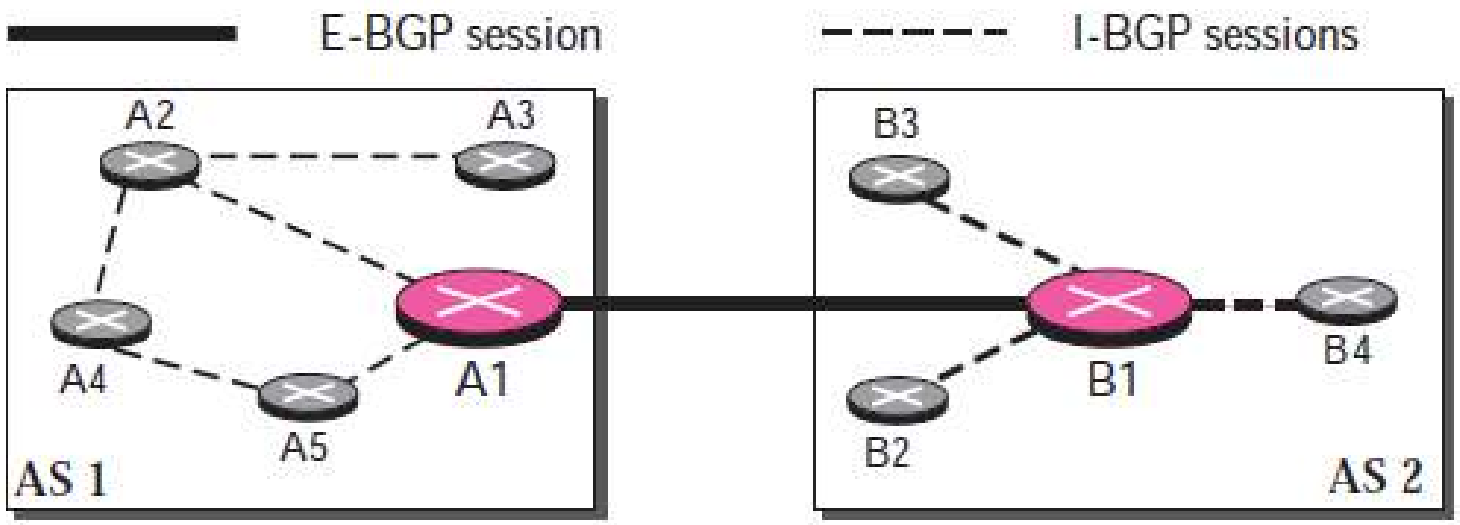
# BGP basics: distributing path information

- using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
  - 1c can then use iBGP do distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- when router learns of new prefix, it creates entry for prefix in its forwarding table.



# *Internal and external BGP sessions*

- The IBGP used to connect different routers have same AS(same company)
- The EBGP used to connect different routers have different AS(different company)



# BGP route selection

- router may learn about more than 1 route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

# BGP messages

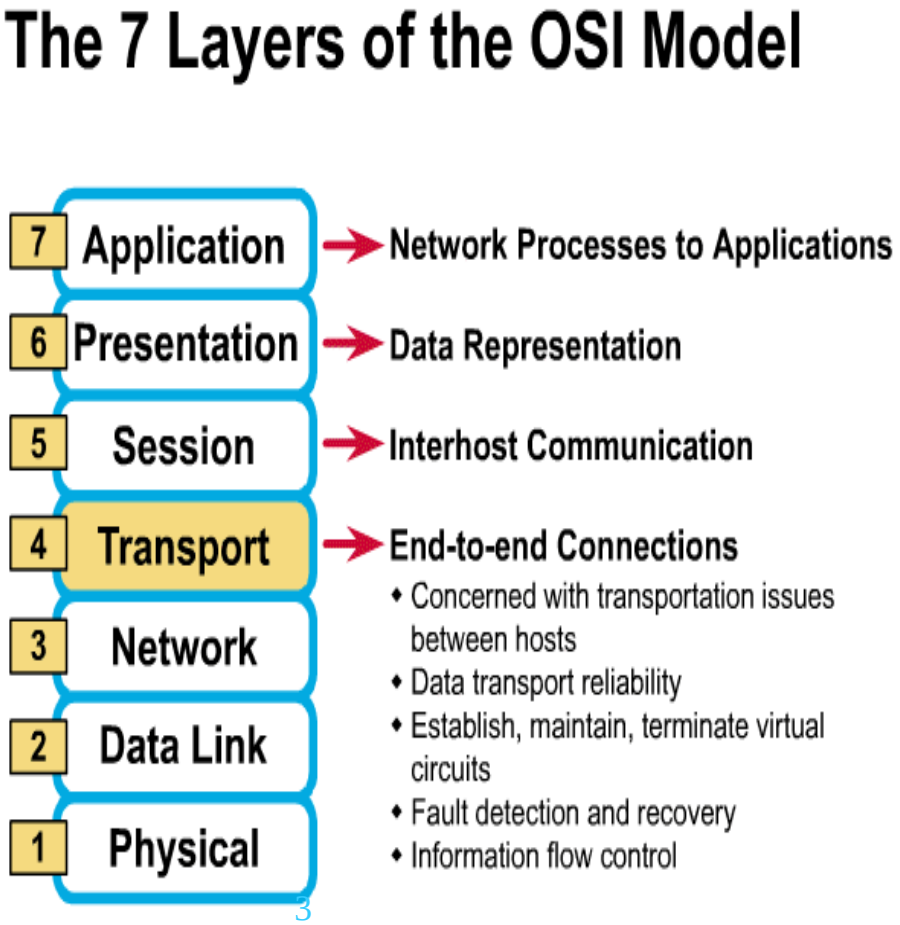
- BGP messages exchanged between peers over TCP connection
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

# 4. Transport Layer

- Provides reliable data delivery
- It's the TCP in TCP/IP

- Receives info from upper layers and segments it into packets

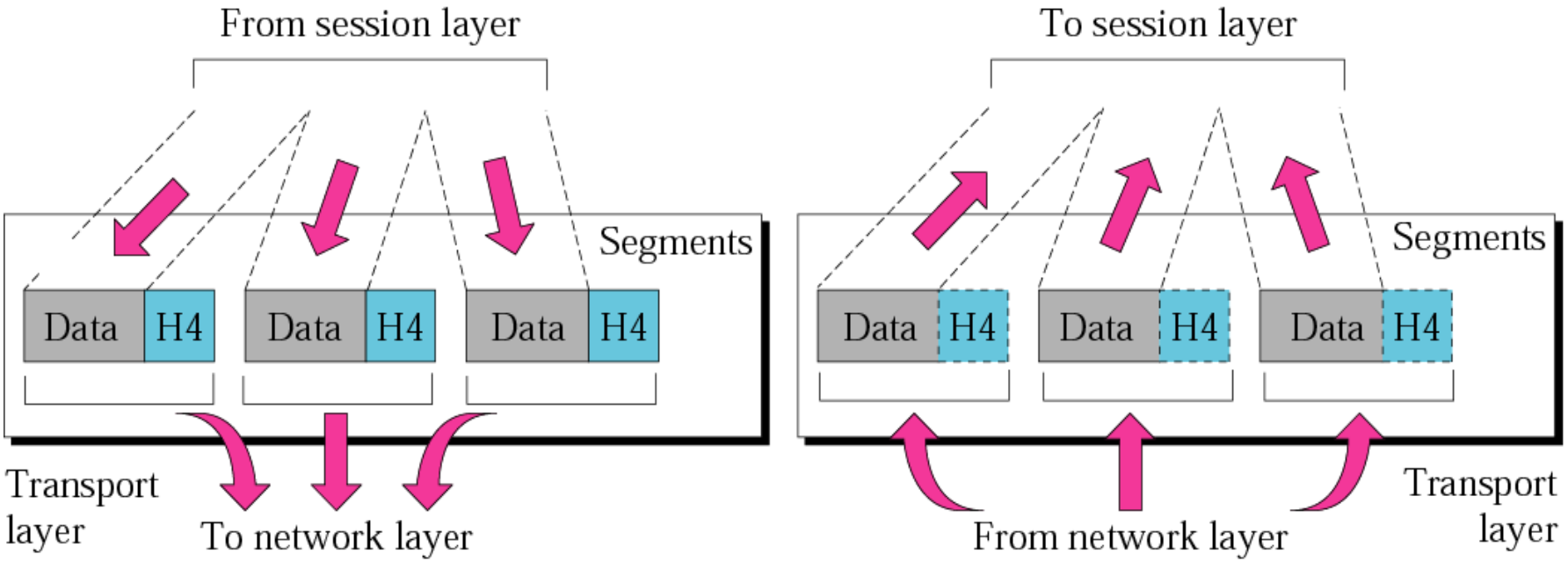
- Can provide error detection and correction



# OSI model

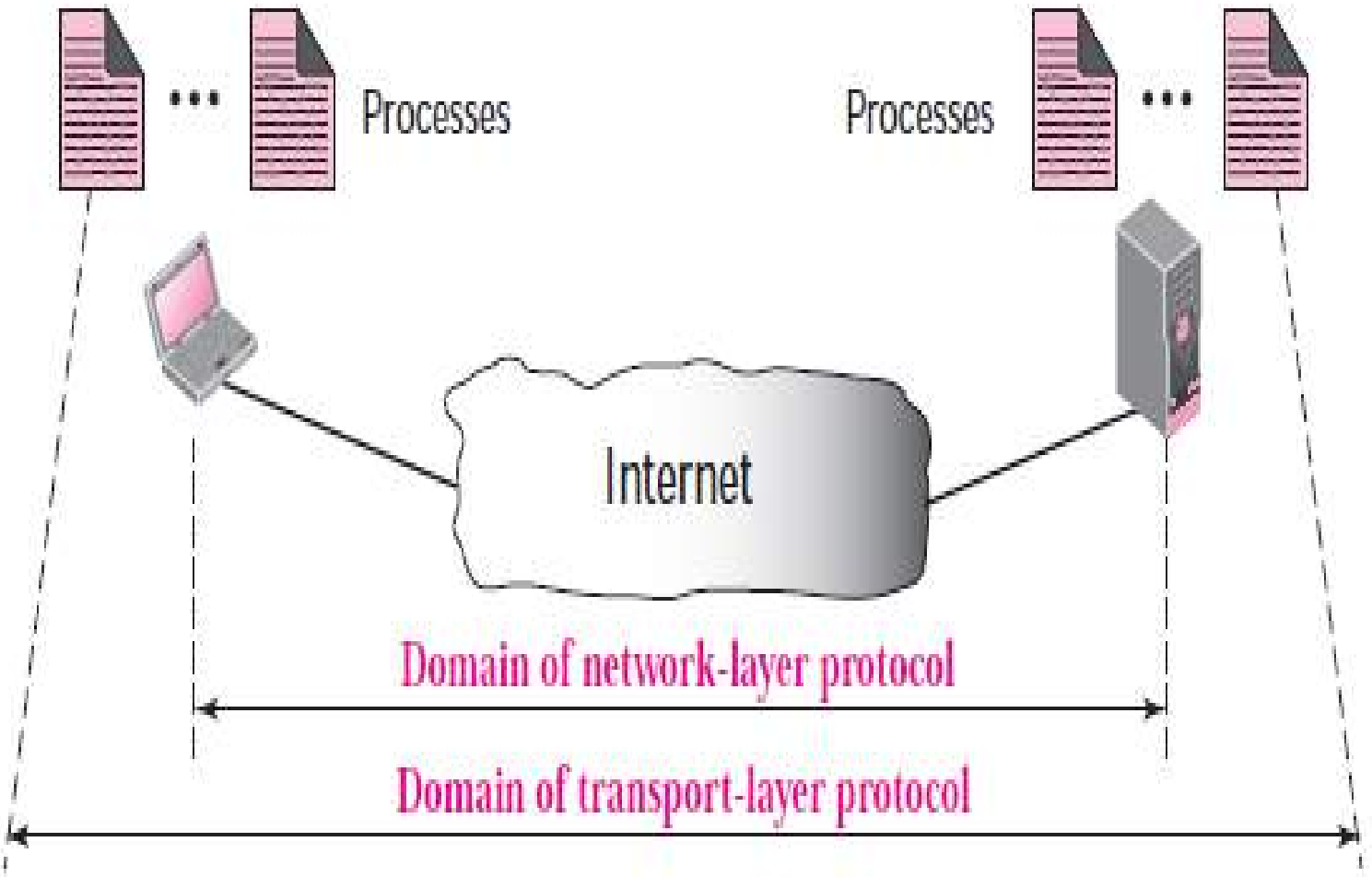
| Layer | Name | Example protocols |
|---|---|---|
| 7 | Application Layer | HTTP, FTP, DNS, SNMP, Telnet |
| 6 | Presentation Layer | SSL, TLS |
| 5 | Session Layer | NetBIOS, PPTP |
| 4 | Transport Layer | TCP, UDP |
| 3 | Network Layer | IP, ARP, ICMP, IPSec |
| 2 | Data Link Layer | PPP, ATM, Ethernet |
| 1 | Physical Layer | Ethernet, USB, Bluetooth, IEEE802.11 |

# Transport layer




***The transport layer is responsible for the delivery of a message from one process to another.***

# Transport layer Protocols



# Why Flow and Error control

- ✓ For **reliable** and **efficient** data communication a great deal of coordination is necessary between two machines. Some of these are necessary because of the following
- ✓ **Constraints:**
- Both sender and receiver have limited speed. (receive, send process data).
- Both sender and receiver have limited memory(storage) .

- ✓ **Requirements:**
- A fast sender should not **overwhelm** a slow receiver, which must perform a certain amount of processing before passing the data on to the higher-level software.
- If error occur during transmission, it is necessary to devise mechanism to correct it.

# What is Flow Control

- **Flow Control** is a technique so that transmitter and receiver with different speed characteristics can communicate with each other.
- To control the flow of data, the receiver needs to send some **feedback** to the sender to inform the latter it is overwhelmed with data.

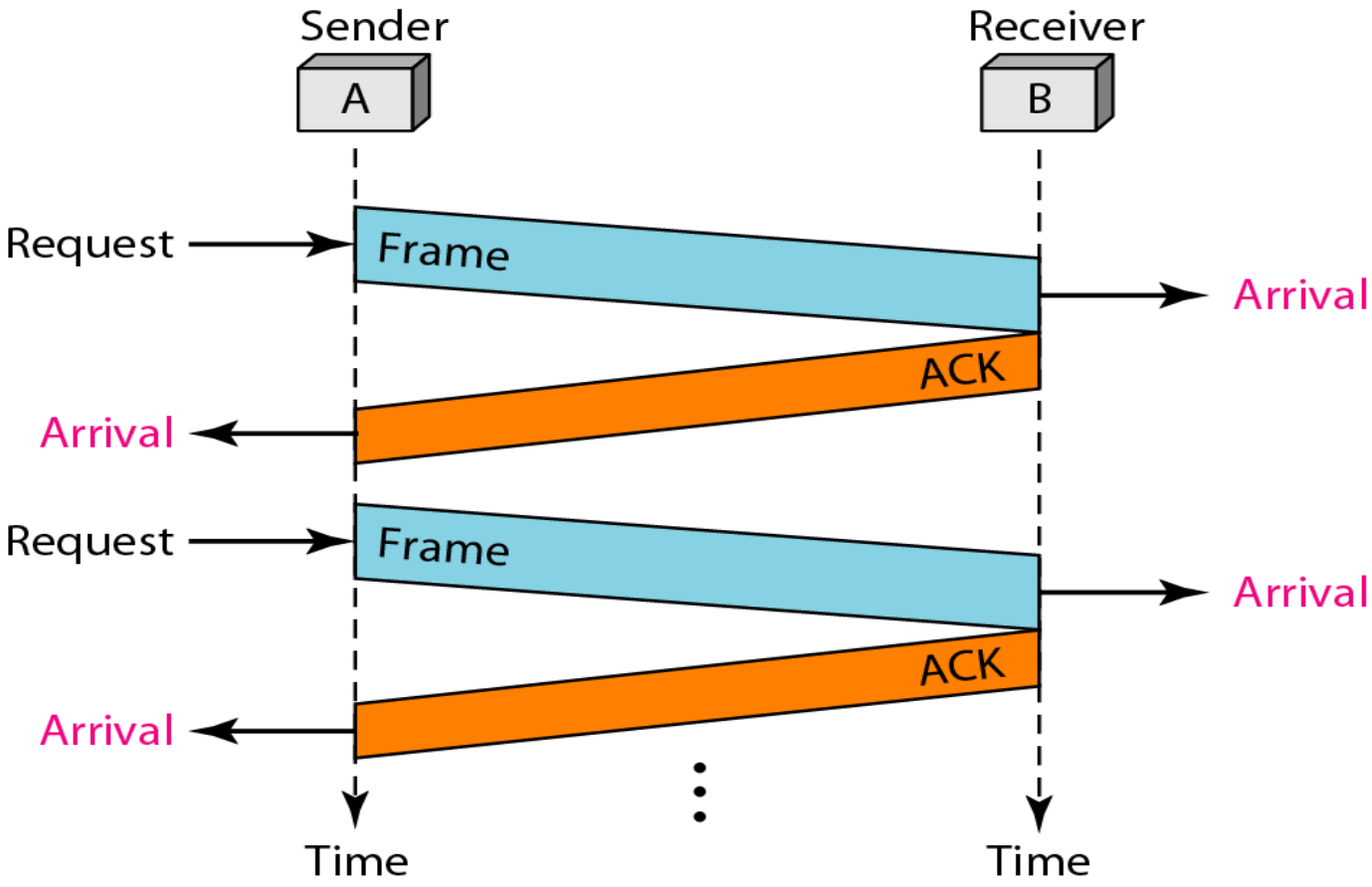
# FLOW CONTROL

- **ACK** is a packet sent by one host in response to a packet its has received
- **Time out** is a signal that an ACK not came from receiver the sender will retransmission this packet
- **Propagation delay** is define as delay between transmission and receipt.
- **Propagation time** can be used to estimate time out period.

# Stop-and-Wait

- This is the **simplest** form of flow control.
- After receiving the frame, the receiver indicates its willingness to accept another frame by sending back an **ACK** frame acknowledging the frame just received.
- The sender must **wait** until it receives the ACK frame **before** sending the next data frame.

# Stop-and-Wait



# Disadvantage of stop and wait

- At any time there is only **one frame** that sent and waiting to be ACK1
- This is **not** good for transmission **media.**
- To improve efficiency **multiple frames** should be transmit while send is wait to one ACK this called sliding window

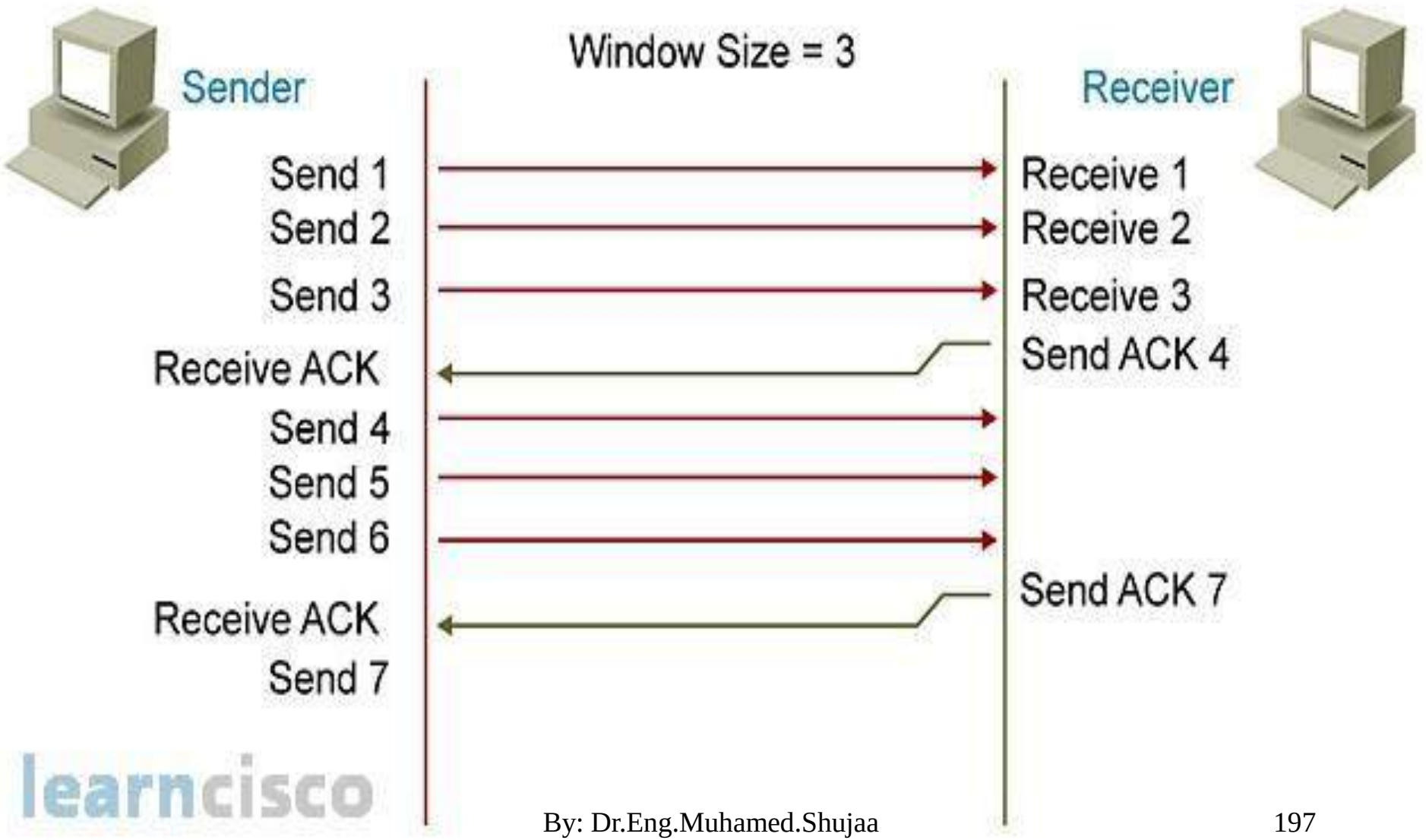
# Sliding Window flow control

- With the use of multiple frames for a **single** message, the stop-and-wait protocol does not perform well. Only **one frame** at a time can be in transit. In stop-and-wait flow control, if $a > 1$, *serious inefficiencies result.*
- Efficiency can be greatly improved by allowing **multiple frames** to be in transit at the **same time.**

# Sliding Window(sender)

- To keep track of the frames, sender station sends **sequentially** numbered frames .
- **Window** meaning number of frames that sender can transmit in same time.
- Size of window can be variable.

# Sliding Window Flow control



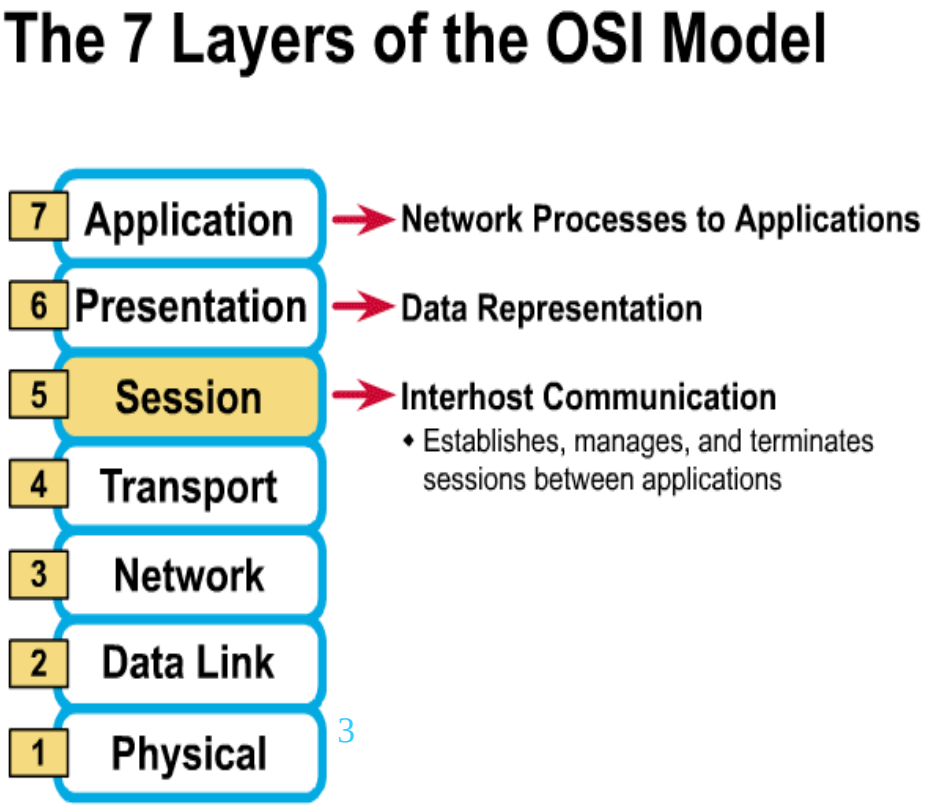
# Congestion Control

## Reasons of congestions:

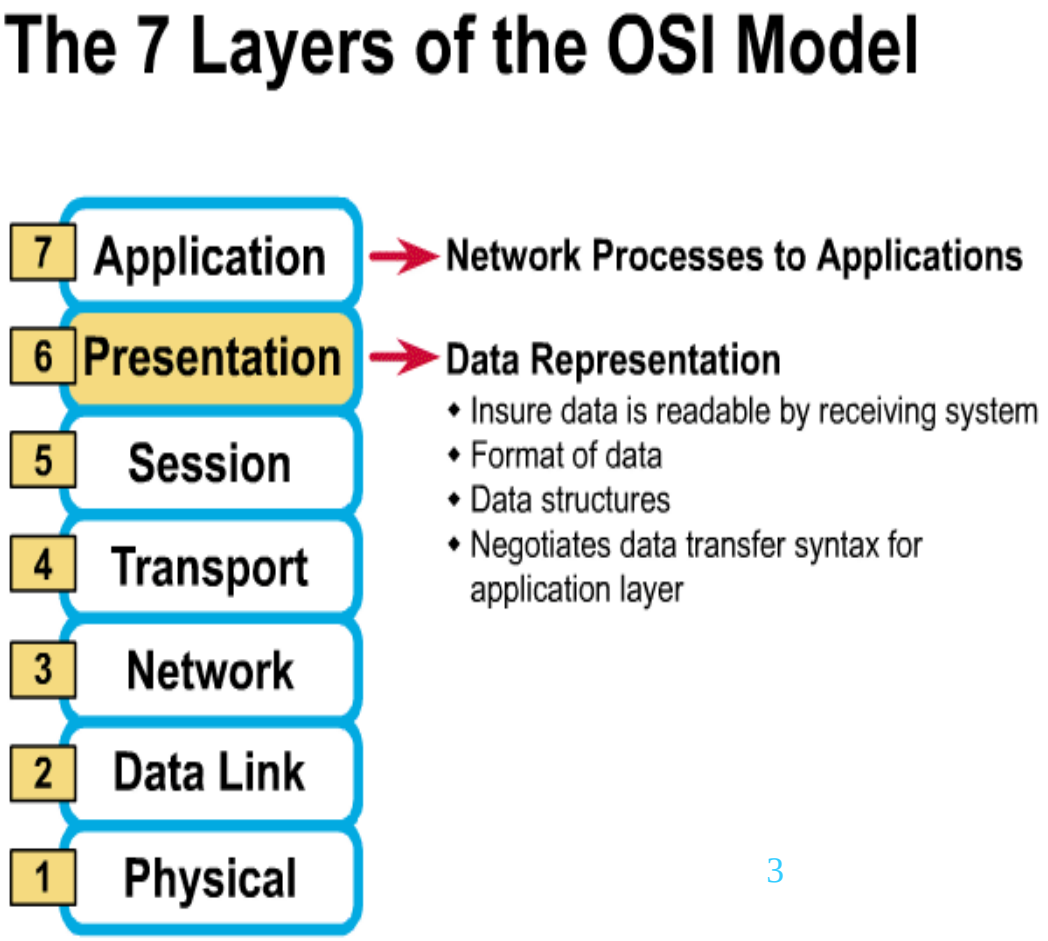
- Load on the network.
- The number of packets sent is greater than the capacity of the network.
- Low Bandwidth.
- Slow Processor.

# 5. Session Layer

- Allows applications to maintain an ongoing session
- Where is it on my computer?
  - Workstation and Server Service (MS)
  - Windows Client for NetWare (NetWare)



# 6.Presentation Layer




3

# SECURE SHELL PROTOCOL (SSH)

- Another popular **remote login** application program is **Secure Shell (SSH).**
- SSH, like TELNET, uses **TCP** as the underlying transport protocol, but SSH is **more secure** and provides more services than TELNET.
- Covers authentication, encryption.
- Solve the security issues at **remote login** of **Telnet.**
- Solve the security issues during **file transfer** at **FTP**.

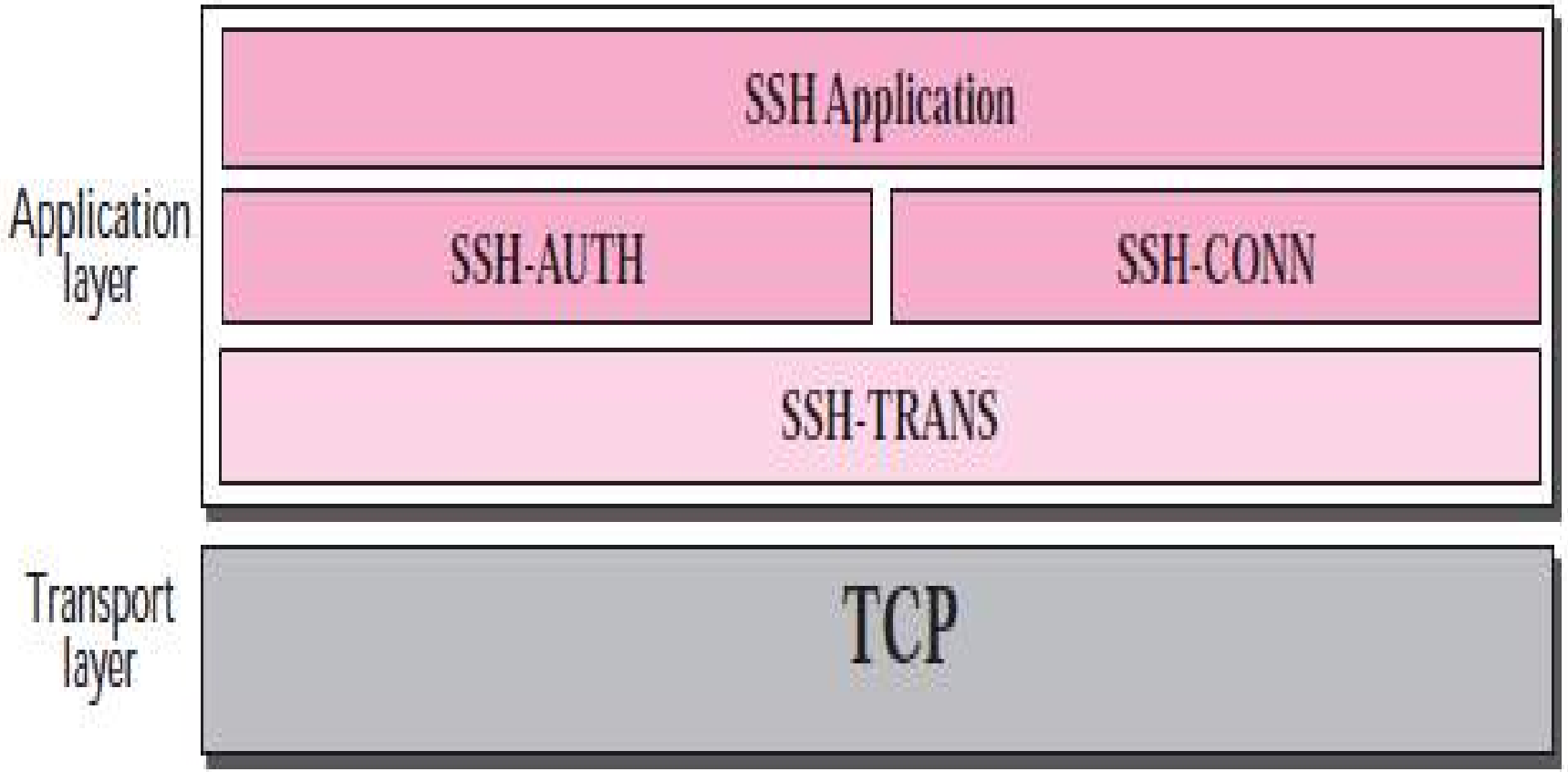
# SSH

- There are two versions of SSH: **SSH-1** and **SSH-2**, which are totally incompatible.
- The first version, SSH-1 is now deprecated because of **security problems** in it.
- SSH is a proposed application-layer protocol with four components.
- In this lecture, we discuss only SSH-2.

# SSL vs SSH

- SSL is **TCP**-based and always used in **WEB applications**, with HTTP.
- SSH is **TCP**-based and always used with **Telnet** and **FTP**

# SSH-2 components



# ***SSH Transport-Layer Protocol (SSH-TRANS)***

- Privacy or confidentiality of the message exchanged. السرية والامان
- Data integrity. سلامة البيانات
- Server authentication. التوثيق
- Compression of the messages that improve the efficiency of the system and makes attack more difficult. ضغط البيانات

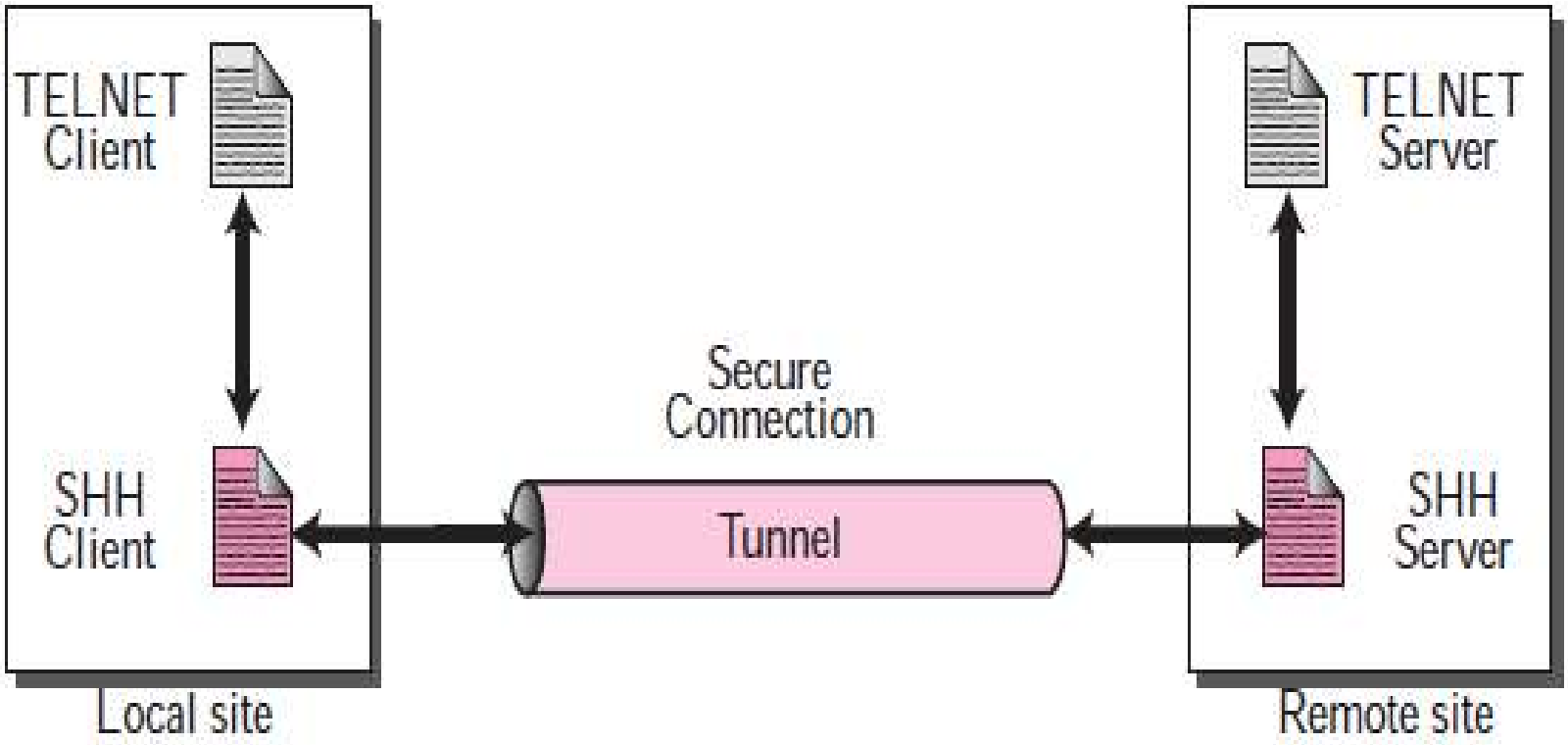
# ***SSH Authentication Protocol (SSH-AUTH)***

- After a secure channel is established between the client and the server and the server is authenticated for the client, SSH can call another software that can authenticate the client for the server.

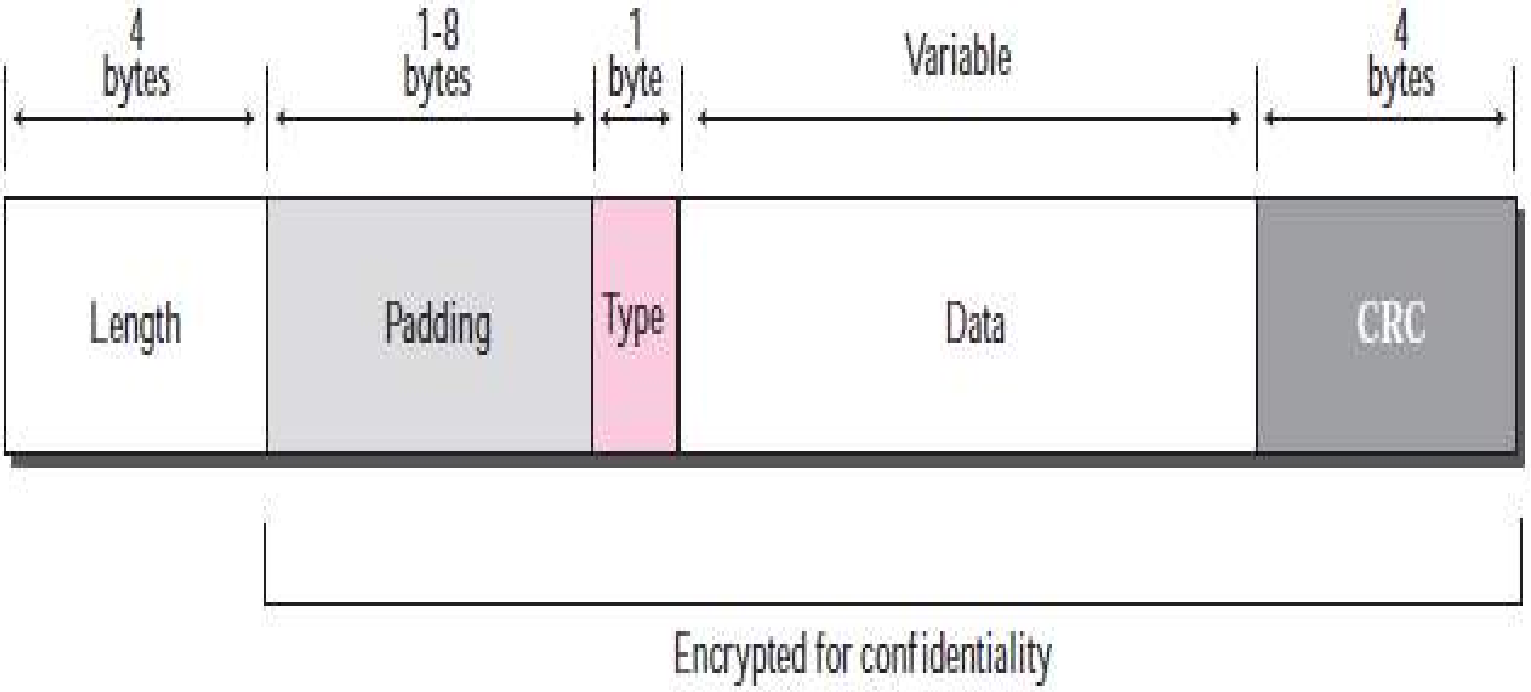
# ***SSH Applications***

- Remote login is one of the services that can use the SSH-CONN protocols; other applications, such as a **file transfer** application can use one of the logical channels for this purpose

# Port Forwarding



# Format of the SSH Packets



# SSH Packet

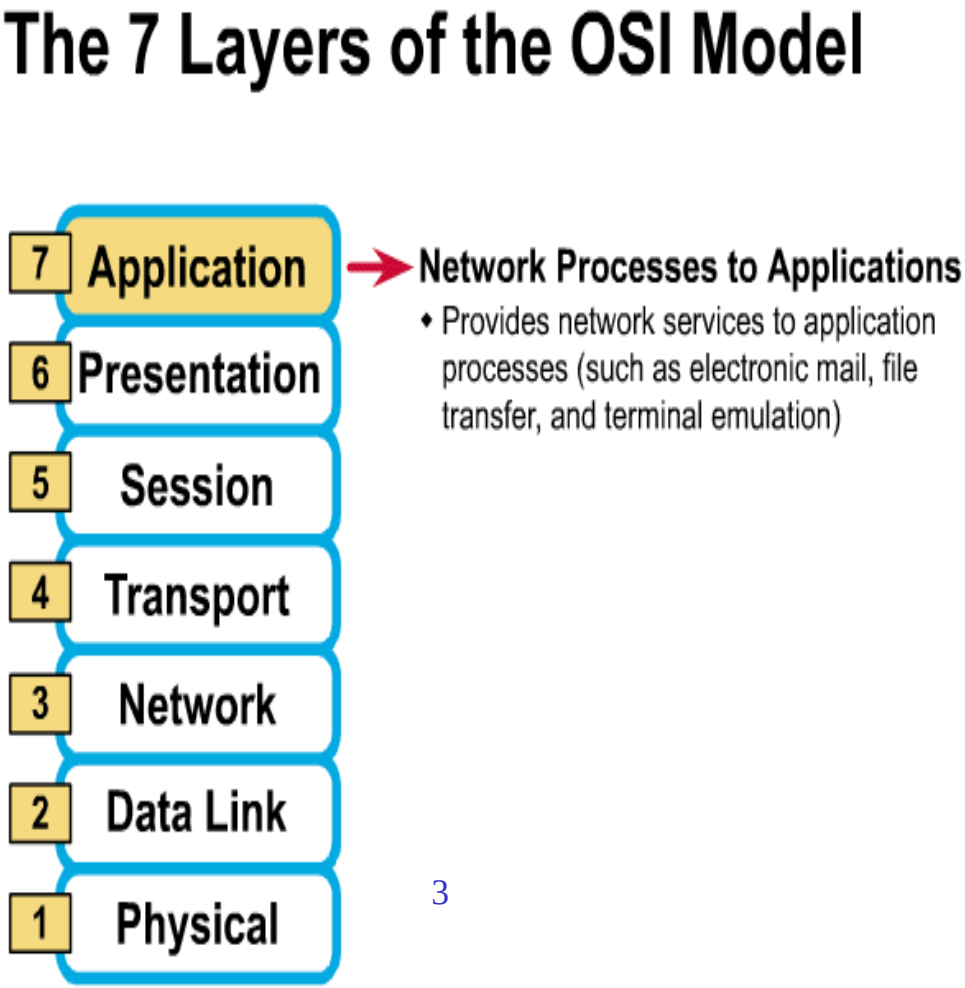
- **Length. This 4-byte field defines the length of the packet including the type, the** data, and the CRC field, **but not** the padding and the length field.
- **Padding. One to eight bytes of padding is added to the packet to make the attack** on the security provision more difficult.

# SSH Packet

- **Type.** This one-byte field defines the type of the packet used by SSH protocols.
- **CRC.** The cyclic redundancy check filed is used for **error detection**.

# 7. application

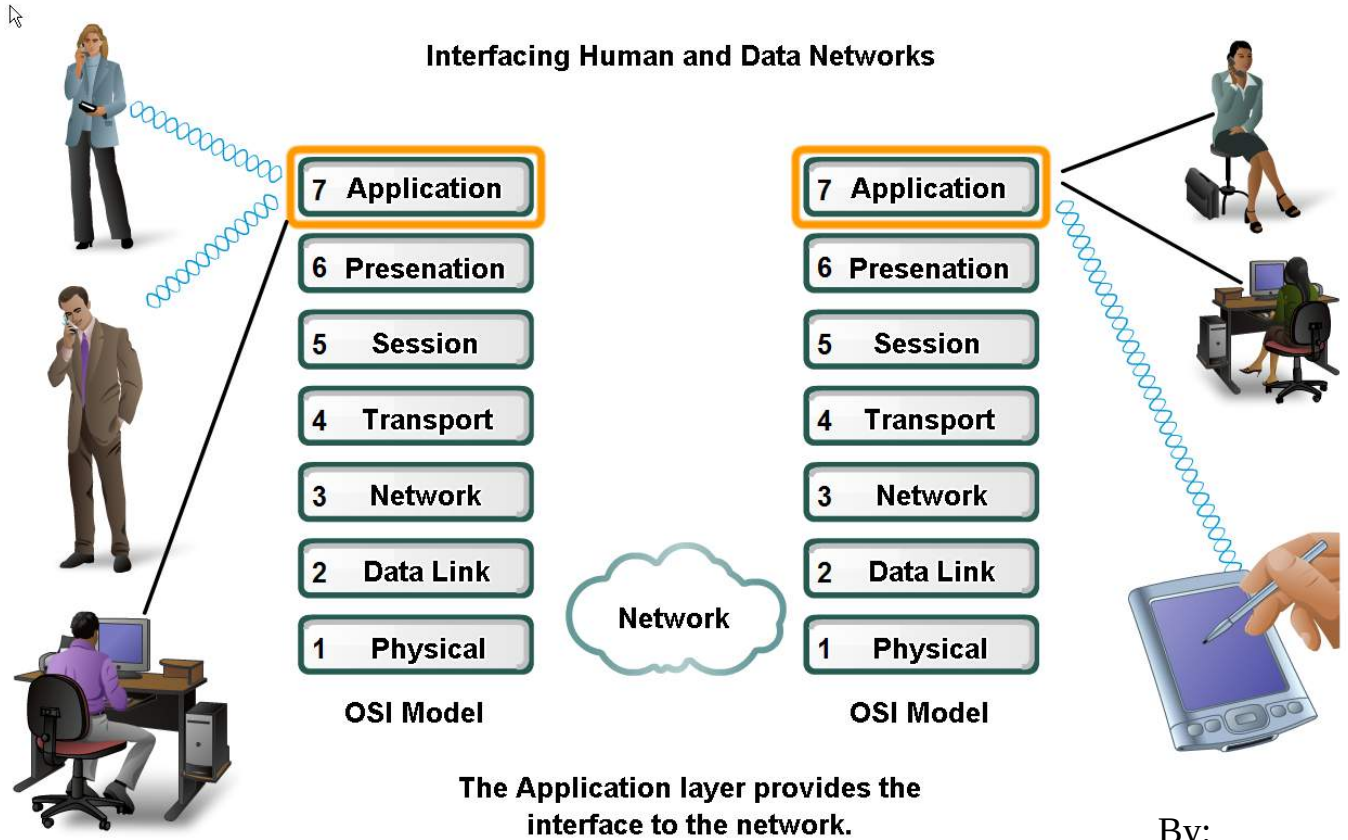
- Gives end-user applications access to network resources
- Where is it on my computer?
  - Workstation or Server Service in MS Windows

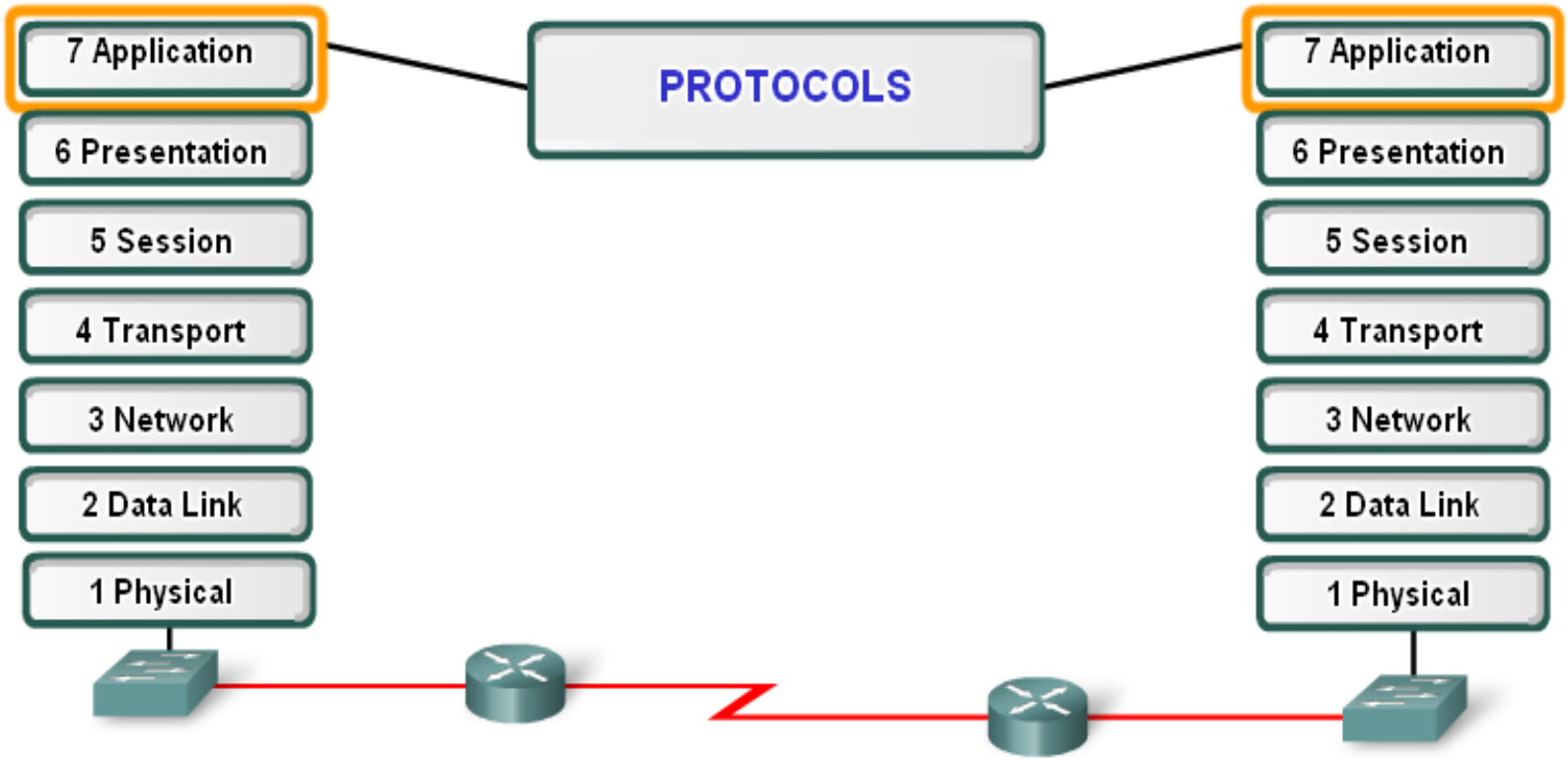



3

## Applications Layer – allows user to interface with the network!

# Application layer



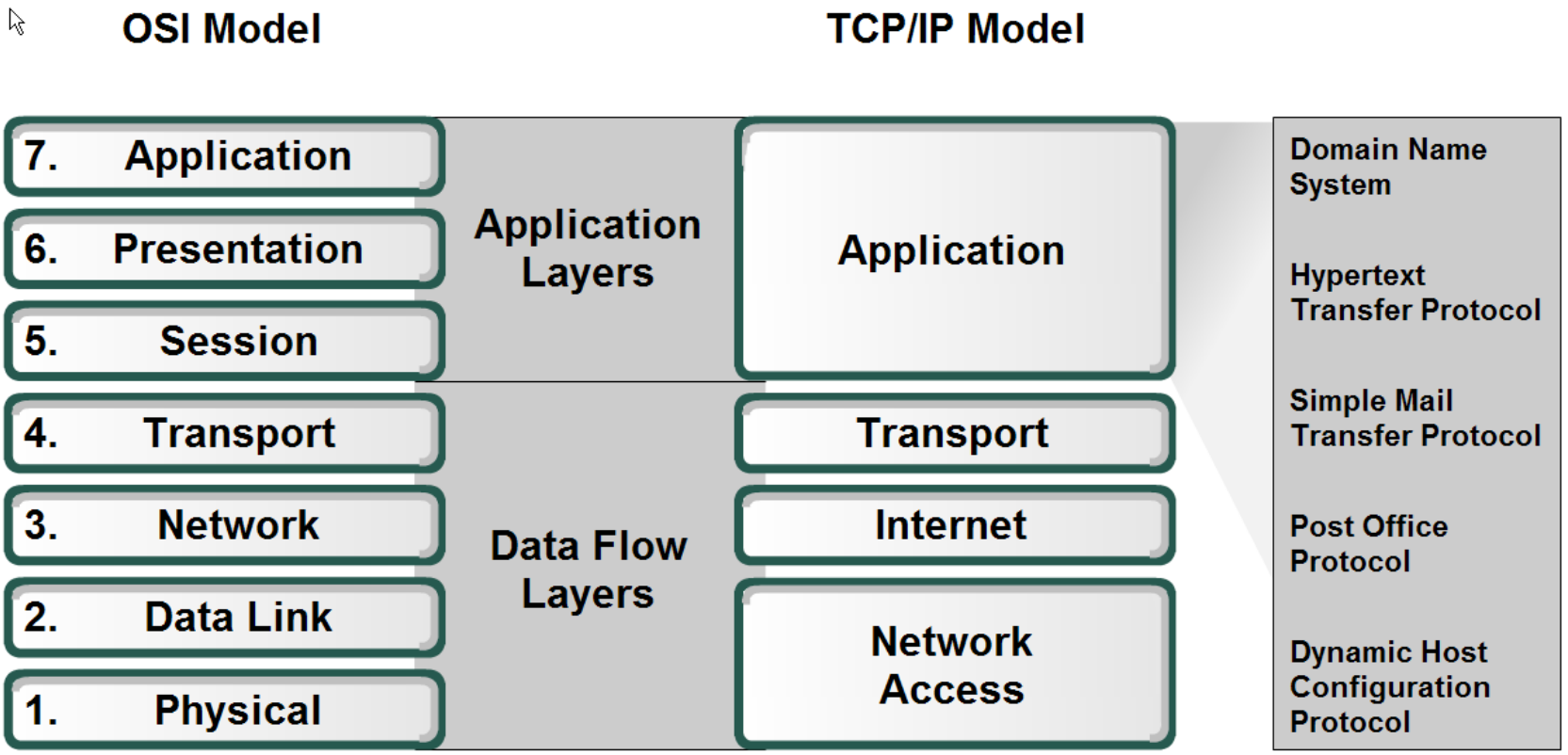



Application layer protocols provide the rules for communication between applications.

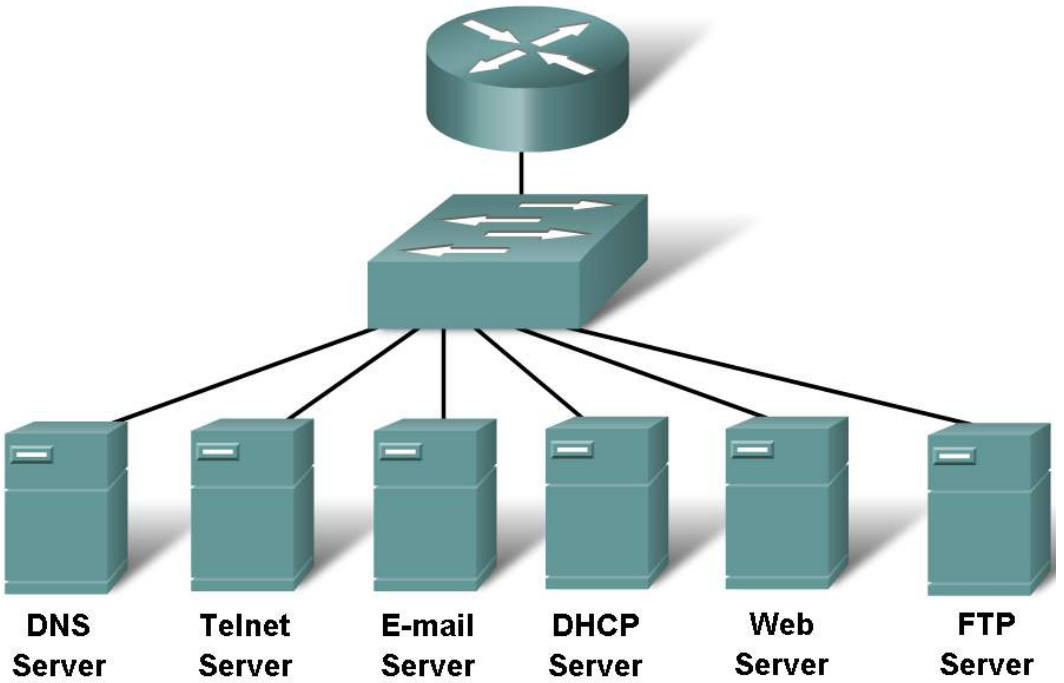
**Protocols:**

- Define processes on either end of the communication
- Define the types of messages
- Define the syntax of messages
- Define the meaning of any informational fields
- Define how messages are sent and the expected response
- Define interaction with the next lower layer

# Application Layer – Provides the interface between the applications on either end of the network.



# Protocols and networks



# Protocols

- DNS – Matches domain names with IP addresses
- HTTP – Used to transfer data between clients/servers using a web browser
- SMTP & POP3 – used to send email messages from clients to servers over the internet
- FTP – allows the download/upload of files between a client/server
- Telnet – allows users to login to a host from a remote location and take control as if they were sitting at the machine (virtual connection)
- DHCP – assigns IP addresses, subnet masks, default gateways, DNS servers, etc. To users as they login the network

# Application layer software

- 2 types
  - Applications – Provide the human (user) interface. Relies on lower layers to complete the communication process.
  - Services – establish an interface to the network where protocols provide the rules and formats that govern how data is treated..

# How data requests occur & are filled in application layer?

- Client/server model
- Application layer services and protocols
- Peer-to-peer networking and applications

# Peer-to-Peer (P2P) Network Model

- Two or more computers are connected and are able to share resources **without having a dedicated server**
- Every end device can function as a client or server on a ‘per request’ basis
- Difficult to enforce security and policies
- User accounts and access rights have to be set individually on each peer device

# P2P Applications

- Unlike P2P networks, a device can act as both the client and server within the same communication
- Each device must provide a user interface and run a background service.
- Can be used on P2P networks, client/server networks and across the internet.