Chapter 13: Fundamentals of Networking and Network Protocols

13.1 Preliminaries

Similar to how `pthreads` library provides an API for multithreading, we will use the `socket` library for developing networking applications

13.2 Basic Terminologies

- Packet flow on the internet traverses through a series of *queues* that reside at the input of the routers *en route* from source to destination
- In the presence of absence other traffic, we might experience *network congestion* and packets encounter *queueing delays*

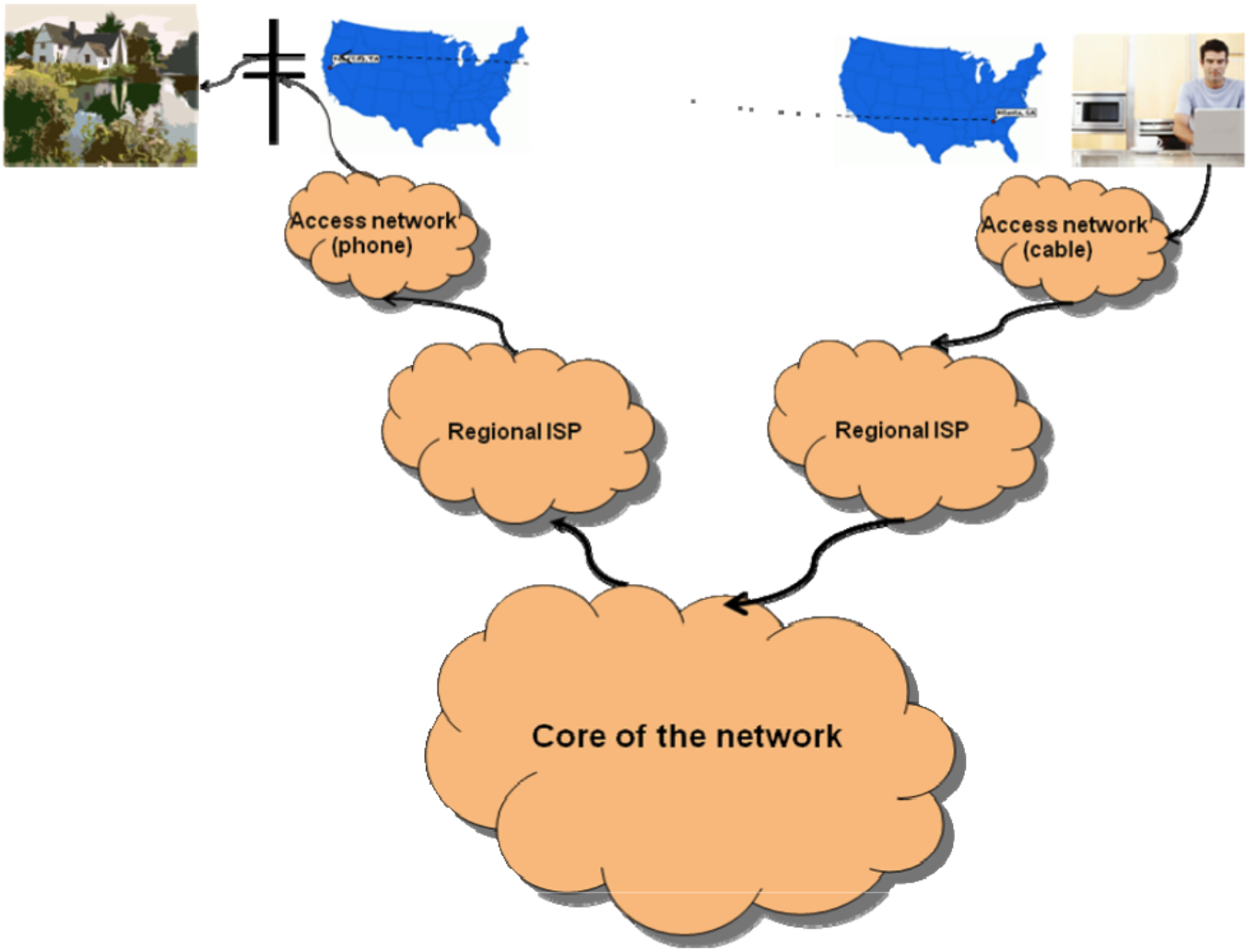


Figure 13.1: Passage of Charlie's e-mail from Atlanta, GA to Yuba City, CA

13.3 Networking Software

- ***Protocol stack*** - networking software part of the OS
  - Language that defines the syntax and semantics of messages for computers to talk to one another
  - Agreed convention

Problems We Might Face In Networking

1. **Arbitrary message size** and physical limitations of network packets
2. **Out of order** delivery of packets
3. **Packet loss** in the network
4. **Bit errors** in transmission
5. **Queuing delays** en route to the destination

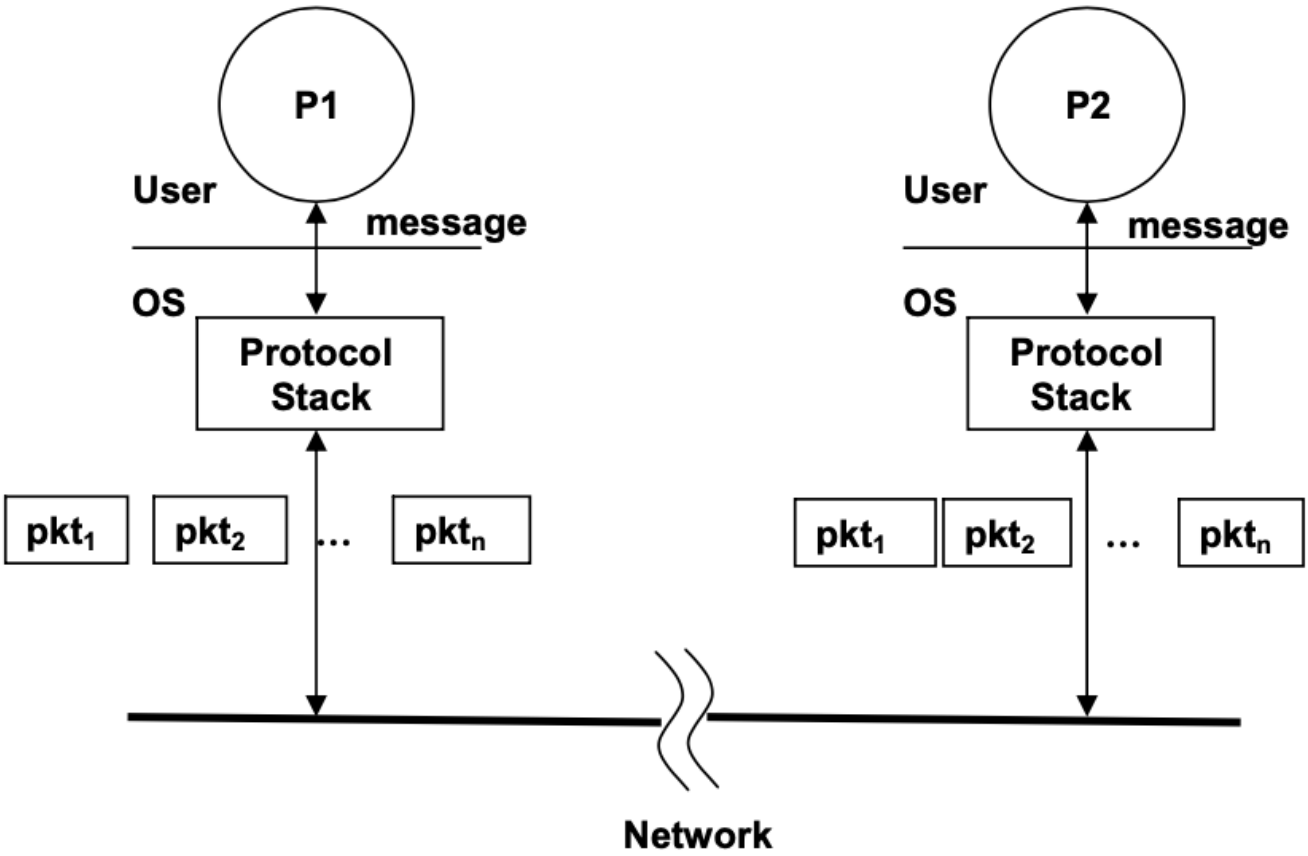


Figure 13.2: Message Exchange Between Two Network Connected Nodes

13.4 Protocol Stack

Rather than bundling all functionality into one stack, protocol layering abstracts the concerns into 5 different layers

- Outgoing messages are *pushed down* the layers of the stack
- Incoming messages are *popped up* from the bottom layers of the stack

Internet Protocol Stack

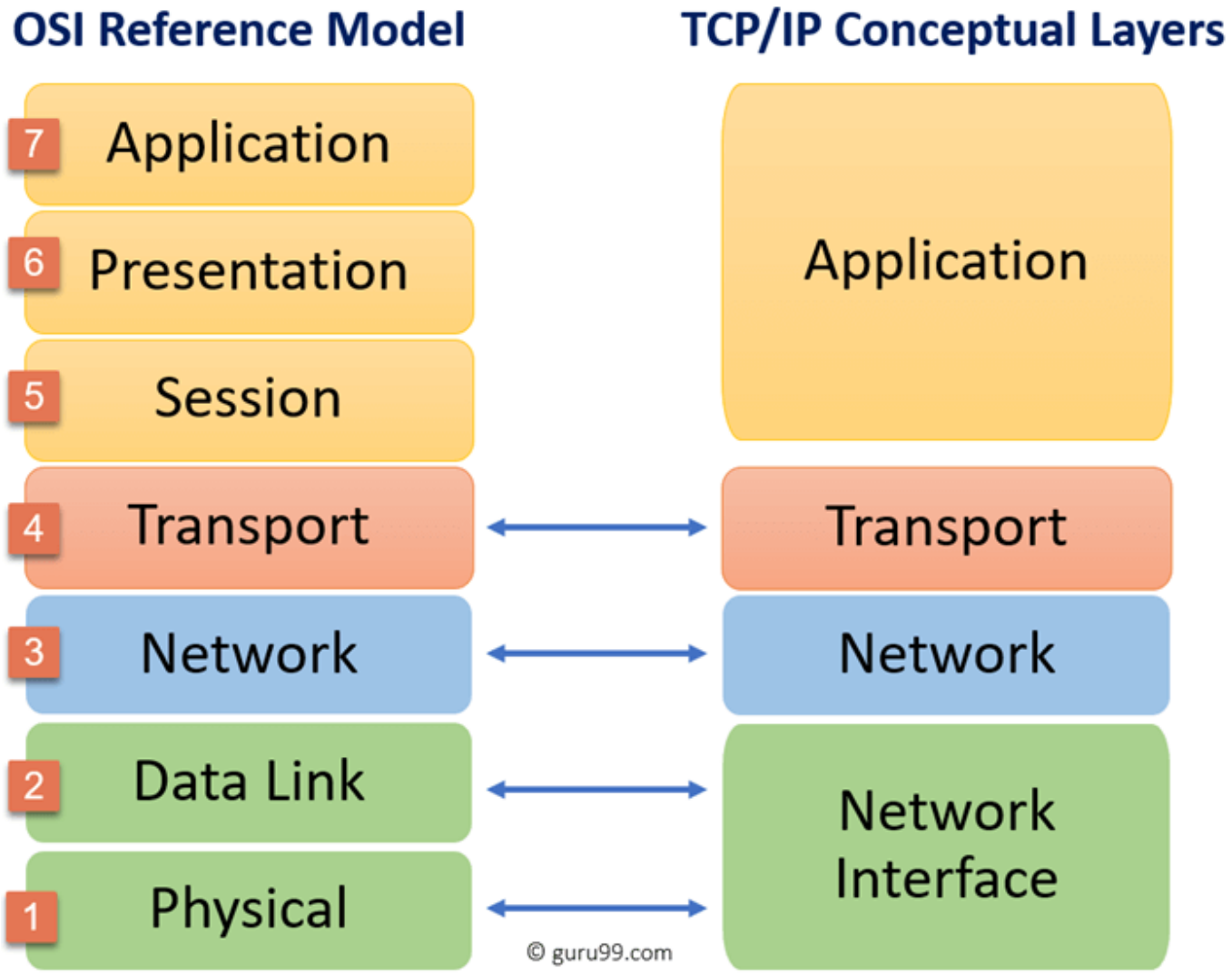


Figure 13.3: Internet Protocol Stack

Layers

- Application Layer: support for network-based applications. Examples are HTTPS for web applications, SMTP for mail, and FTP for files

- **Transport Layer**: takes application layer message and ferries it to its destination. Two biggest protocols are TCP and UDP
- **Network Layer**: responsible for routing a packet from source to destination
    - On sending side, given a packet from the transport layer, the network layer finds a way to get the packet to the intended destination address
    - At receiving end, network layer passes packet up to the transport layer, which is responsible for collating this packet into the message to be delivered to the application layer
    - Similar to the postal service. Protocol is generically called *Internet Protocol (IP)*
- **Link Layer**: ferrying the IP packets between nodes on the Internet through which a packet has to be routed from source to destination
    - Network layer hands the IP packet to the appropriate link layer depending on the next hop to be taken (sometimes even breaking up the packet into *fragments*)
    - Link layer delivers the fragments to the next hop, where they are passed up to the network layer. This repeats until packet reaches destination
    - Somewhat analogous to last stop delivery. Examples are Ethernet, Token Ring, and IEEE 802.11
- **Physical Layer**: responsible for physically (electrically, optically, etc.) moving the bits of the packet from one node to the next
    - Link layer protocol may use multiple physical media to ferry the bits
    - Different mediums have different layer protocols (e.g. Ethernet may use a different physical layer protocol for a twisted pair of copper wires, a coaxial cable, an optical fiber, etc.)

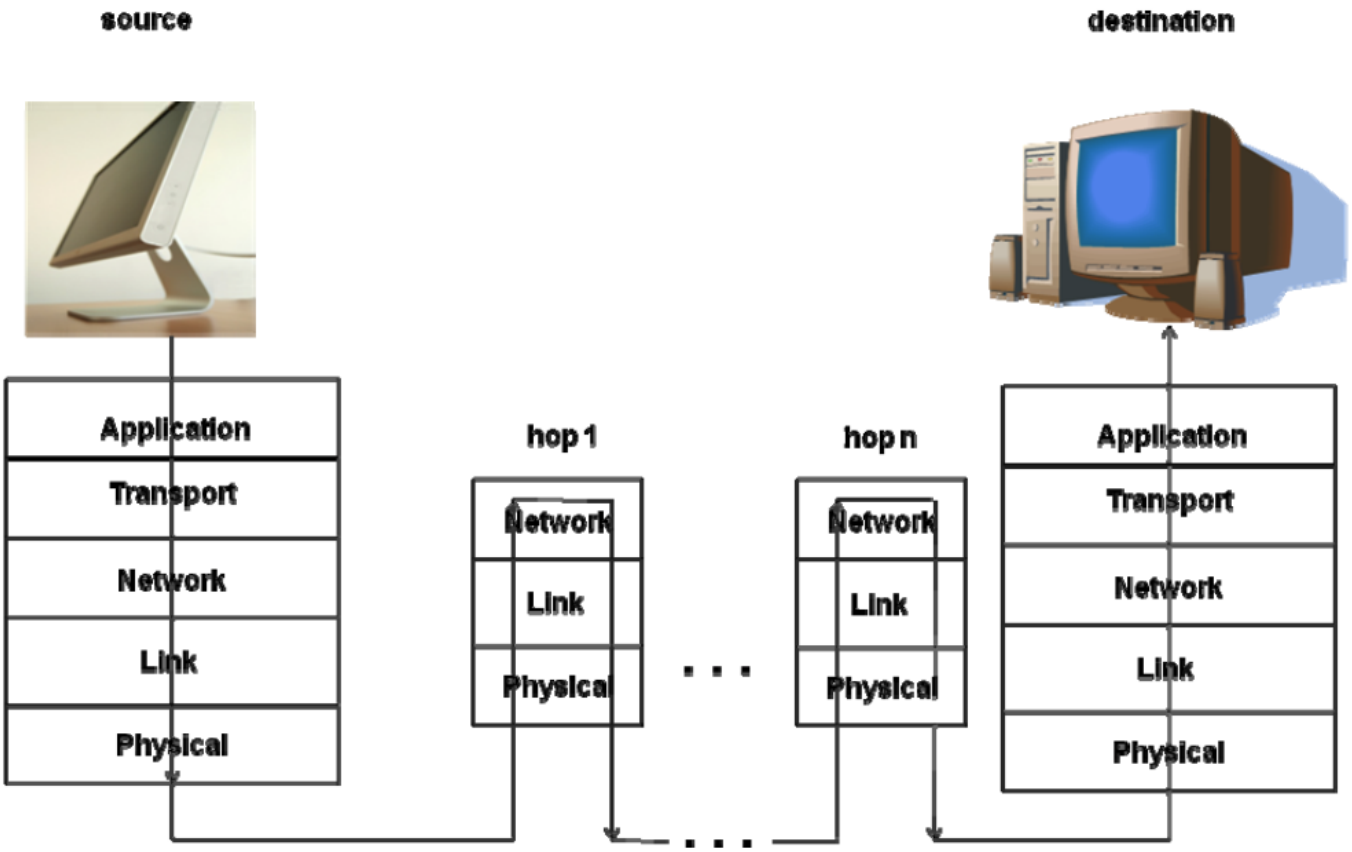


Figure 13.4: Passage of a packet through the network

OSI Model

International Standards Organization (ISO) came up with a 7-layer model of the *Open Systems Interconnection (OSI)* suite

- Refer to figure 13.3 above

Session Layer

- Manager of a specific communication session between two end points
- Example: you have several simultaneous instant messaging sessions
    - Session layer abstracts out the details of the transport from the application, and presents a higher-level interface to the application (for example the Unix socket abstraction

Presentation Layer

- Subsumes some functionality that may be common across several applications
- Example: formatting the textual output on your display window is independent of the specific client program (AOL, MSN, etc.) you may be using for IM

    - presentation functions (such as locally echoing text characters, formatting, and character conversion) that are not dependent on the internal details of the application itself belong in the presentation layer

Practical Issues With Layering

- OSI model is a great reference, though it is almost never implemented that way in real life
- Protocols such as HTTP, FTP, and SMTP subsume layers 7-5 of the OSI model

13.5 Application Layer

Generally, any network application has two parts

1. **Client**: part that sits on end devices such as handhelds, cellphones, laptops, and desktops
2. **Server**: part that provides the expected functionality of some network service (e.g., a search engine)

Application-layer protocols may be tailored for different classes of network applications

- Web apps use HTTPS
- Email uses SMTP
- File transport uses FTP

13.6 Transport Layer

Transport layer provides an API so application layer can send and receive data on the network

Expected functionality

1. Support arbitrary data size at the application level
2. Support in-order delivery of data
3. Shield the application from loss of data
4. Shield the application from bit errors in transmission

Transport layer may view the data from the application layer either as a *byte stream* or as a *message*

- Stream – TCP
- Message – UDP

Transport layer at the source has to break up the data into ***packets*** commensurate with the hardware limitations of the network

- Peer transport layer at the destination has to assemble the packets into the original message
- This is called ***scatter/gather***

Round Trip Time - time taken for a small message to be sent from the sender to the receiver and back again to the sender

- Estimation of the cumulative time delay experienced at the sender for sending a packet and receiving an acknowledgment

Stop and Wait Protocols

1. Sender sends a packet and waits for ACK
2. When recipient receives packet, send an ACK to Sender with the unique signature
3. Sender waits for a certain amount of time (a.k.a. timeout), then retransmits packet if ACK doesn't appear

We use the sequence number to determine if we received an ACK for the current packet or a previous duplicate packet

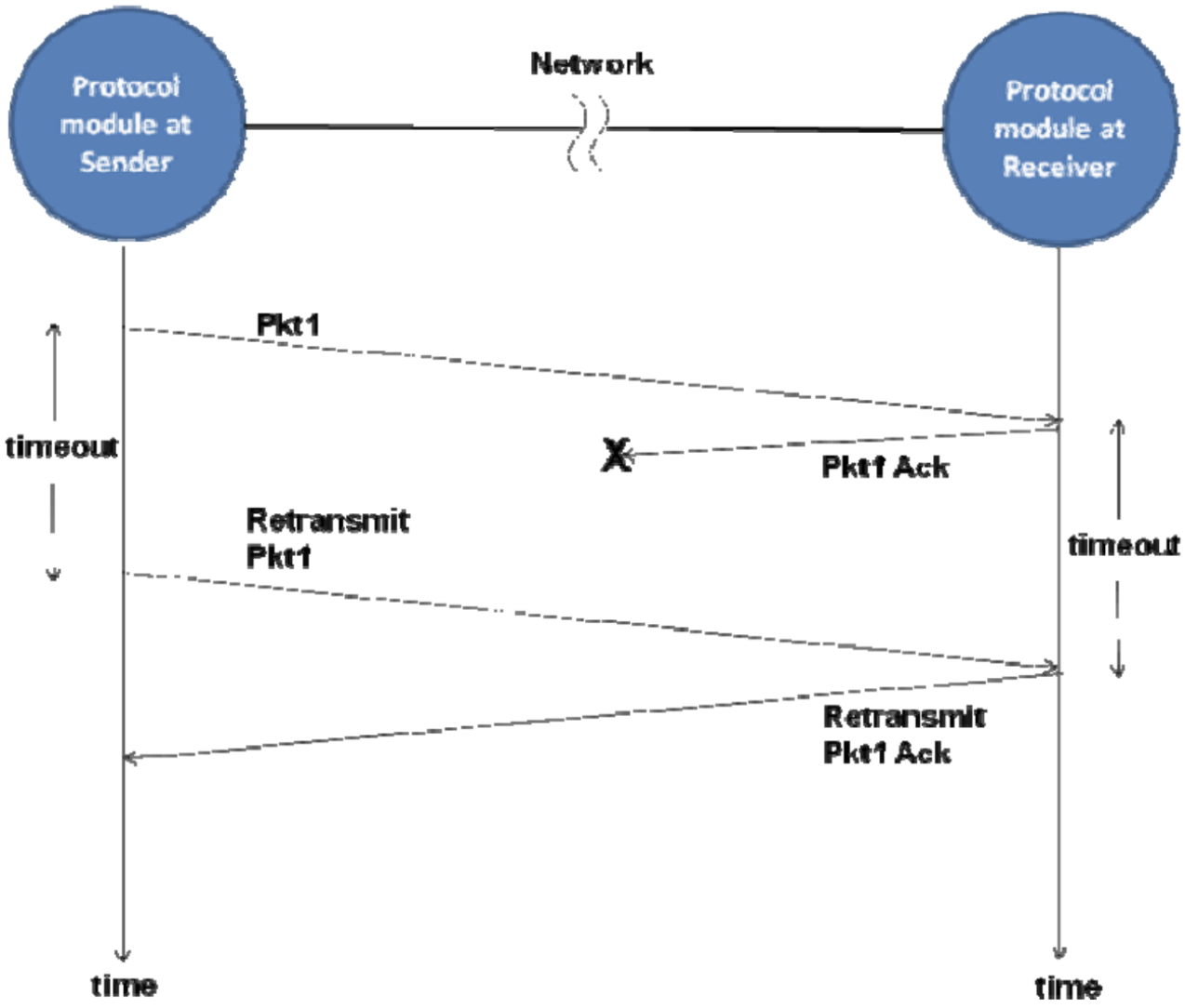


Figure 13.5: Destination retransmits the acknowledgement for previous packet if it receives the same packet

---

Example

A message has 10 packets, and the RTT for a packet is 2 ms. What is the total time for message transmission?

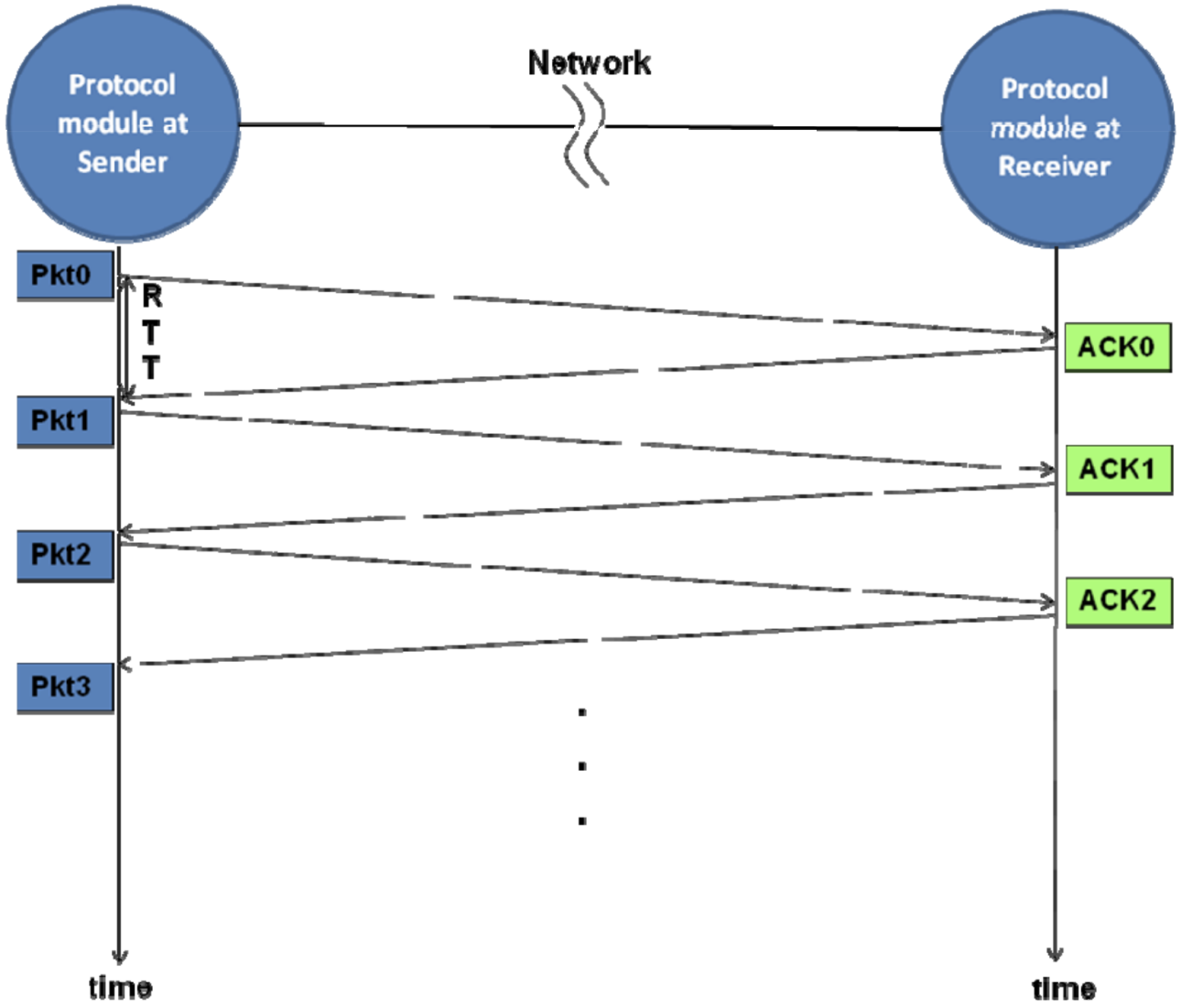


Figure 13.6: Timeline of packet sending in Stop-and-wait protocol

$\text{Total time} = 10 \cdot RTT = 10 \cdot 2 = 20ms$

Pipelined Protocol

There is a lot of *dead time* on the network while the sender is waiting for the ACK to arrive

If the network is reliable, we can blast packets right after each other

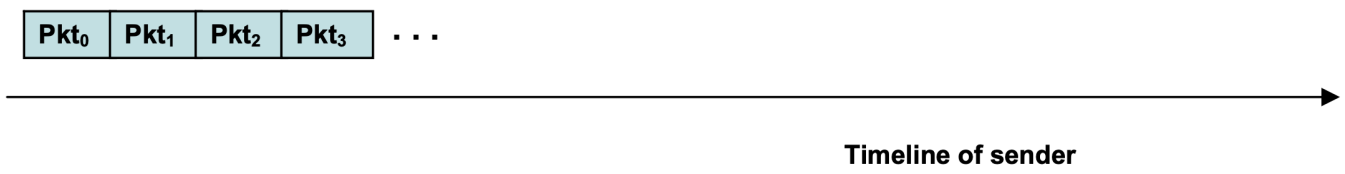


Figure 13.7: Pipelining packet transmission with no ACKs

Example

A message has 10 packets, and the time to send a packet from source to destination is 1 ms. How much time is required to complete the transmission with the pipelining with no-acks protocols?

Total time of transmission = 1 ms

Reliable Pipelined Protocol

The source sends a set of packets (called a window) before expecting an acknowledgment

The destination acknowledges each packet individually as before but the good news is that the sender does not have to wait for acknowledgements for all the outstanding data packets before starting to send again

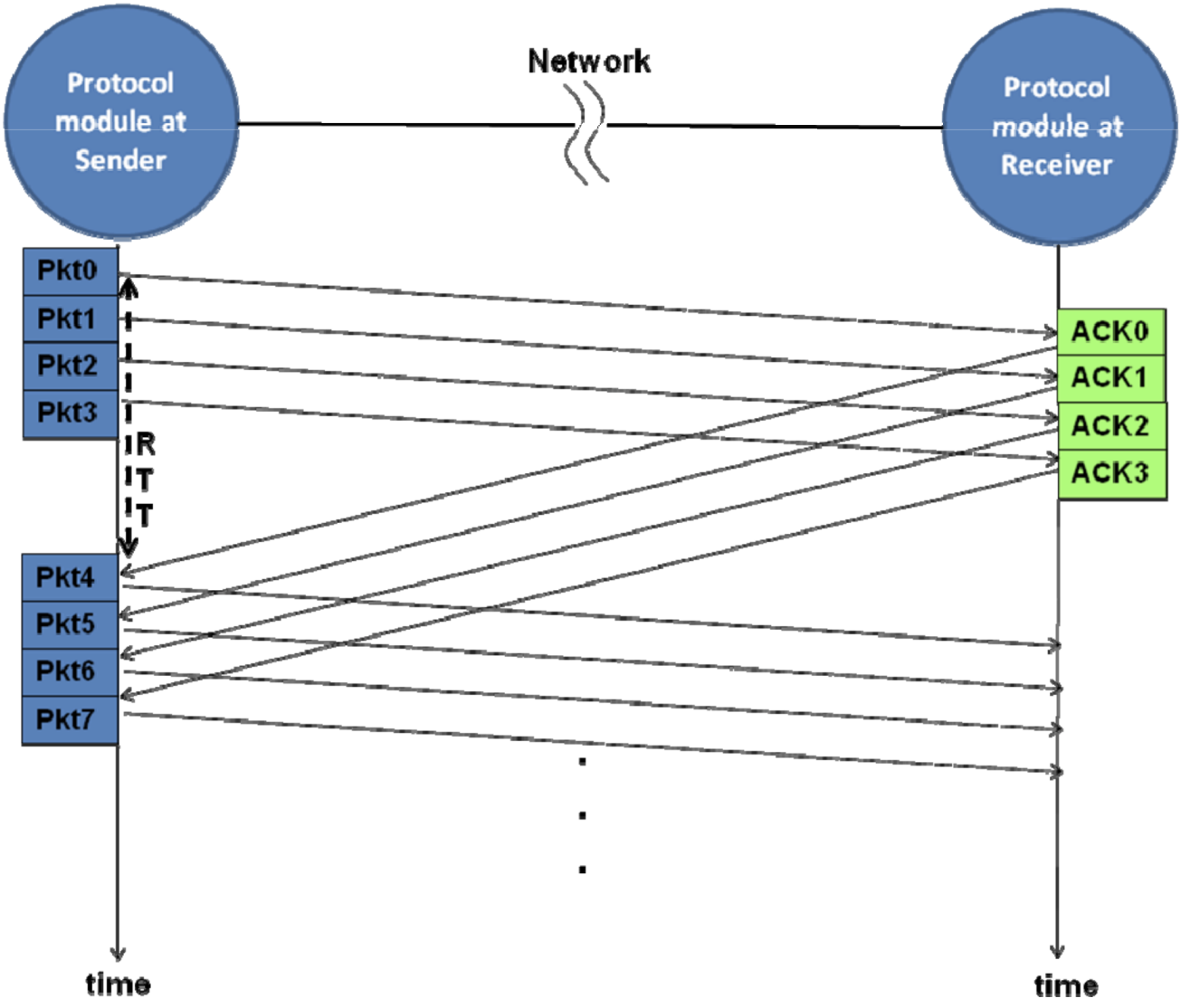


Figure 13.8: Reliable Pipelined Transmission with Acks (window size=4)

In the figure above, as soon as the sender receives ACK0, it is able to send Pkt4-7.

Sliding Window

- As soon as an ACK for the first red packet in the active window is received, the active window moves one-step to the right (the first white packet becomes a blue packet)
- Since the active window slides over the space of sequence numbers (from left to right) as time progresses, we refer to this as a sliding window protocol

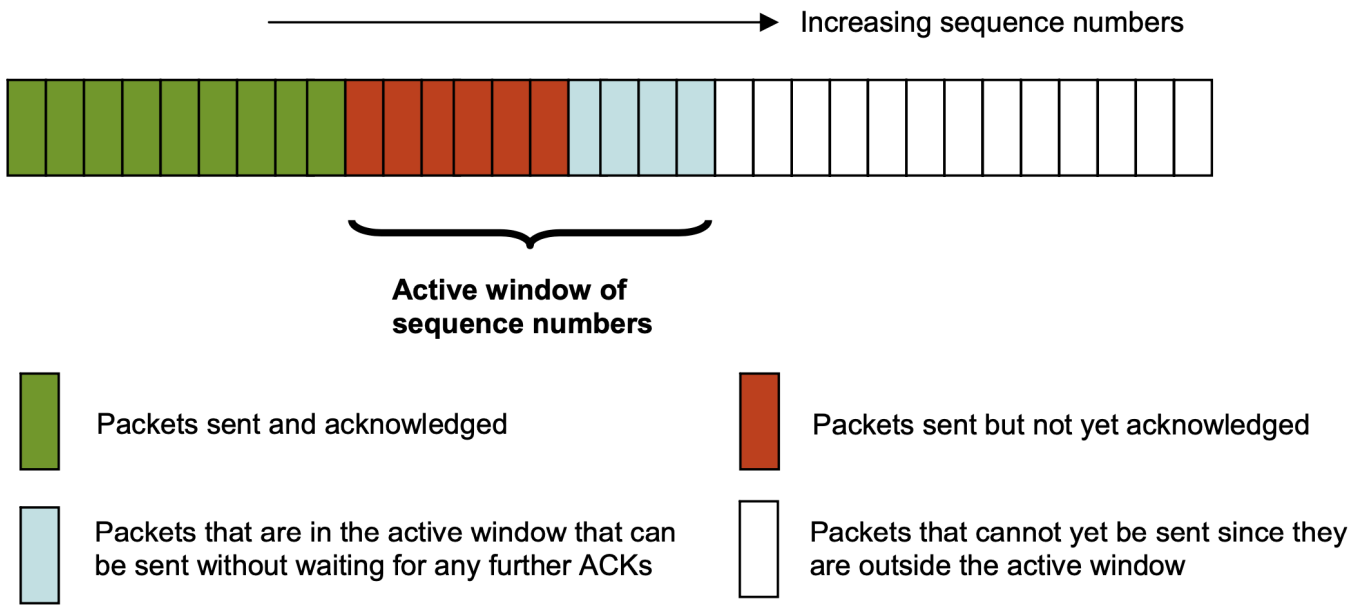
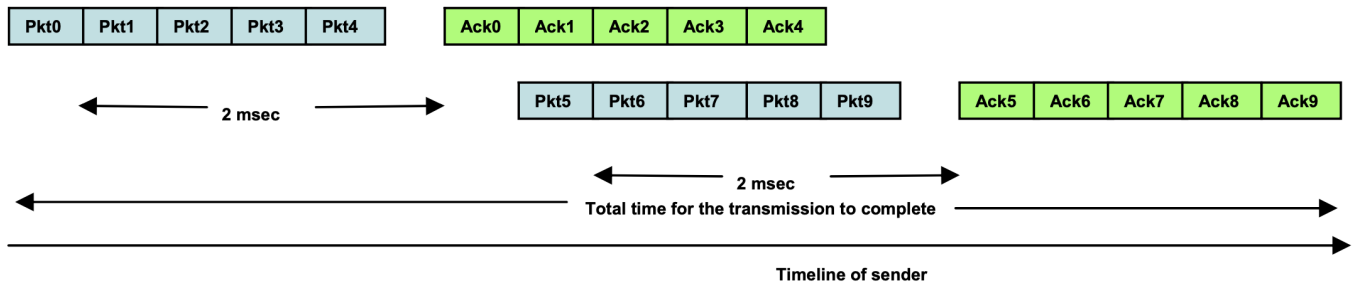


Figure 13.9: Active window (window size=10) of a sliding window protocol

---

Example

A message has 10 packets, and the RTT is 2 ms. How much time is required to complete the transmission with the sliding window protocol with a window size of 5?



With a sliding window size of 5, we need 2 cycles to fully transmit all the packets

$\text{Total time} = 2 \cdot RTT = 2 \cdot 2 = 4$

Turningpoint Question

Question: The timeout for packet retransmission
Answer: is a function of the expected latency for transmission to the destination

TCP

TCP involves first setting up a connection between the two endpoints of communication

- Involves sending streams of data
- Full duplex connection (both sides can simultaneously send and receive messages)

Connection setup: three way handshake

- Client sends the server a connection request message (which has a special field to indicate that it is a connection request), with information about the initial sequence-number it plans to use for its data packets
- The server sends an acknowledgement message to the connection request (once again, with the special field to indicate it is part of the connection establishment 3-way handshake), with information about the initial sequence-number the server plans to use for its data packets
- The client allocates resources (packet buffers for windowing, timers for retransmission, etc.) and sends an acknowledgement (which is the final leg of the three-way handshake). Upon receiving this acknowledgement, the server allocates resources for this connection (packet buffers for windowing, timers for retransmission, etc.).

Reliable Data Transport: protocol guarantees that the data handed to it from the upper layers will be faithfully delivered in order to the receiving end without any loss or corruption of the data

Congestion Control: sender self-regulates its flow by observing the network congestion, and dynamically adjusting its window size to avoid the buildup of queues in the routers

and thus reducing the network congestion

Connection Teardown: two endpoints agree to tear down the connection

- Client sends a connection teardown request (with a special field to distinguish it from normal data) to the server. The server sends an ACK.
- Server sends its own teardown connection request (with a special field to distinguish it from normal data) to the client. The client sends an ACK. The client de-allocates client-side resources associated with this connection. Upon receiving the ACK, the server de-allocates the server- side resources associated with this connection. The connection is officially closed.

UDP

- Connection-less
- No ACKs or Windowing
- Suited for environments with low chance of packet loss and applications tolerant to packet loss
- Unreliable
- Message may arrive out of order
- May contribute to network congestion
- No guarantees on delay or transmission rate

13.7 Network layer

Functionalities we need in this layer

1. Routing Algorithms: determine a route for packets to take from source to destination
2. Service Model: The network layer should forward an incoming packet on an incoming link to the appropriate outgoing link based on the available routing information

Dijkstra's Link State Routing Algorithm

Link state = the cost associated with each physical link at any given node

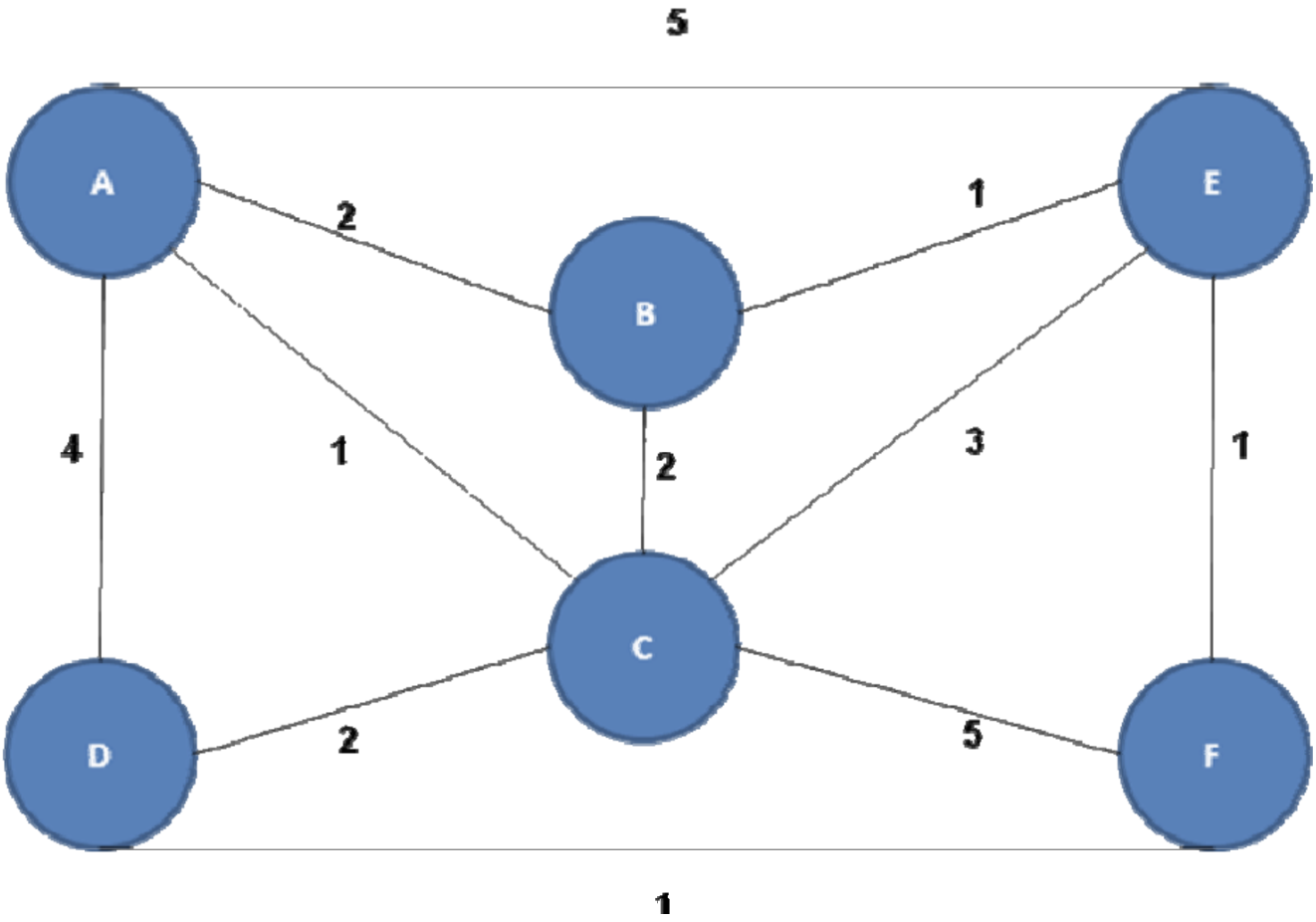


Figure 13.10: An example network

All nodes in the network have complete information about the state of the network (connectivity and costs associated with the links)

Just uses Dijkstra's to find shortest path

Distance Vector Algorithm

This algorithm is *asynchronous* and works with *partial knowledge* of the link-state of the network. Each node maintains a ***routing table***, called a ***distance vector table***. This table has the *lowest cost route of sending a packet to every destination in the network*, through each of the *immediate neighbors* physically connected to this node

	Cost through immediate neighbors			
Destination	A	B	C	F
A	5(EA)	3(BA)	4(ECA)	5(EFDCA)
B	7(EAB)	1(EB)	5(ECB)	6(EFDCB
C	6(EAC)	3(EBC)	3(EC)	4(EFDC)
D	8(EACD)	4(EBEFD)	5(ECD)	2(EFD)
F	9(EABEF)	2(EBEF)	7(ECBEF)	1(EF)

Figure 13.11: Distance Vector Table for Node E

A node recomputes its DV table if

1. node observes a change in the link-state for any of the immediate neighbors (for example, due to congestion, let us say the link-state from E to B becomes 5 from 1)
2. node receives an update on the least-cost route from an immediate neighbor

Hierarchical Routing

Split the internet into groups called ***autonomous systems***

- Routers within an AS may run one of LS or DV protocols for routing among the hosts that are within an AS
- Gateway routers of different ASs communicate with one another using a different protocol called ***Border Gateway Protocol*** (BGP)

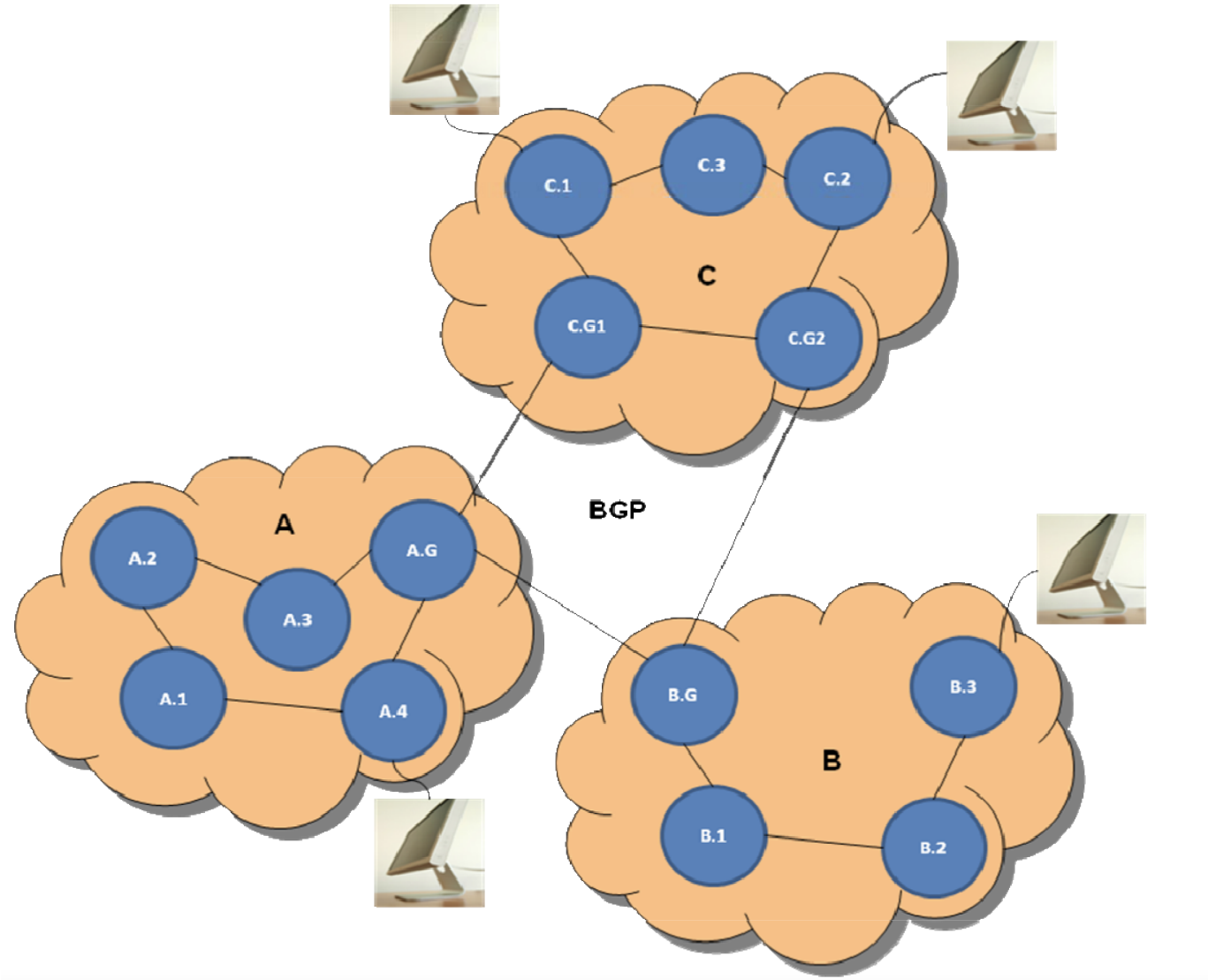


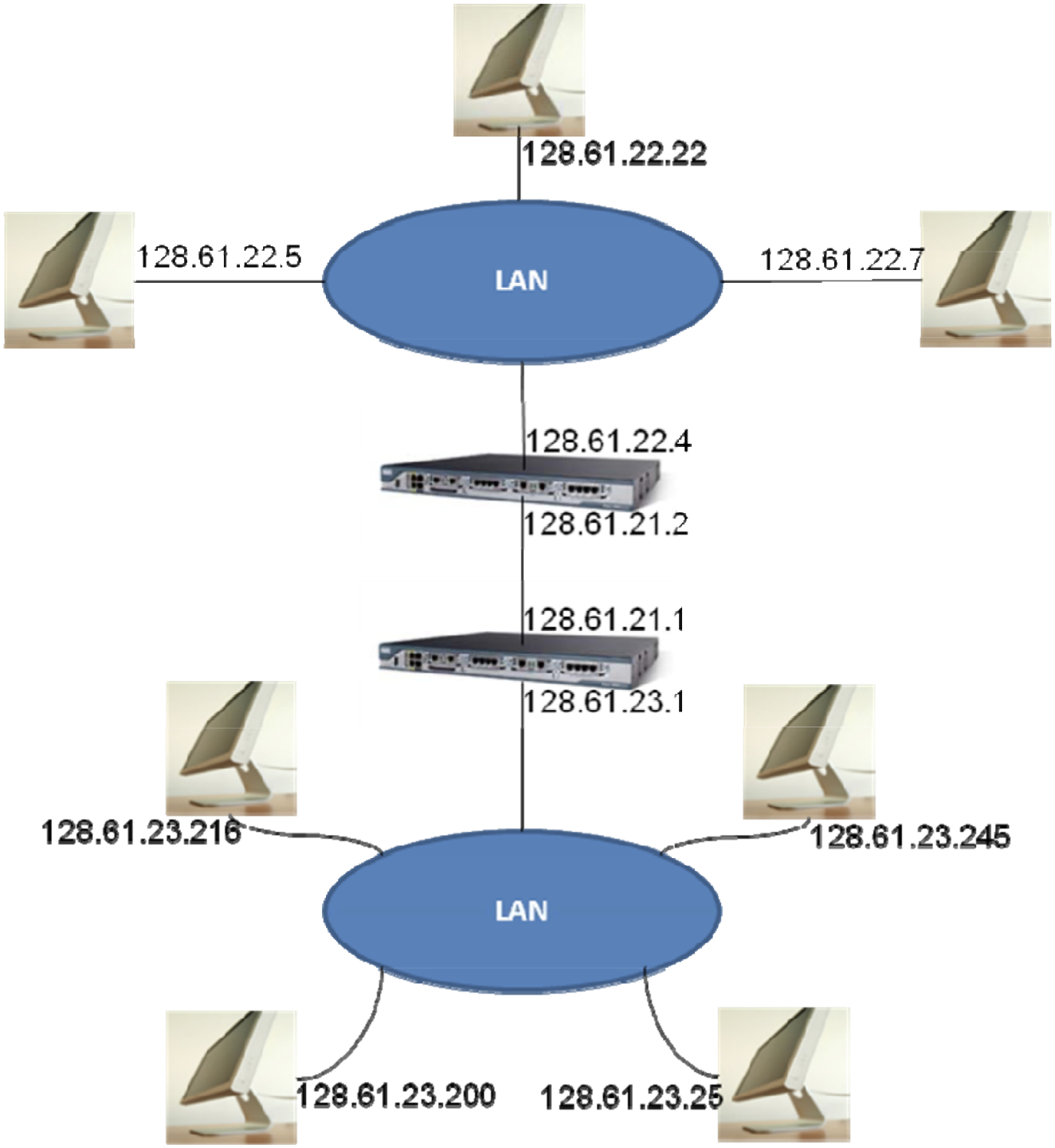
Figure 13.12: Three different ASs coexisting with Intra- and Inter-AS protocols

To find if hosts are on the same network, look at the top 24 bits of the IPv4 address

---

Example

How many IP networks are in this Figure?



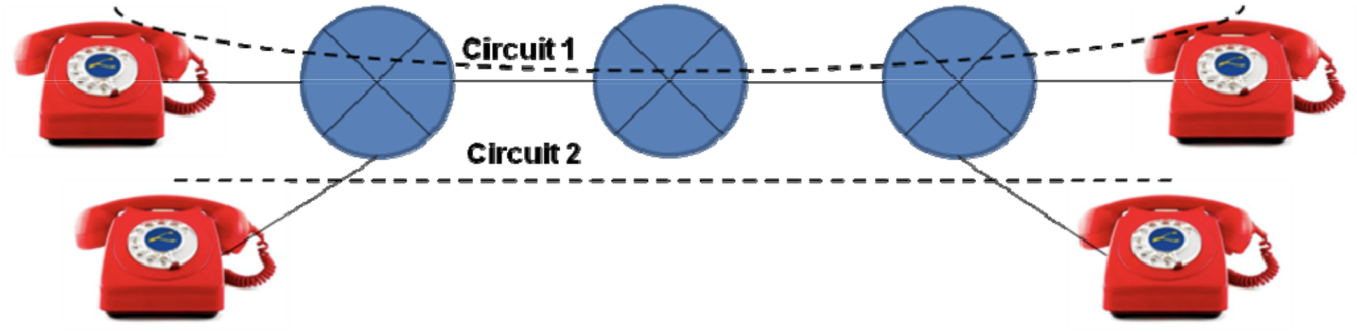
Answer:

There are **three** IP networks in this figure. One for the lower half of the figure connecting the 4 hosts on the lower LAN with the bottom router (network address: `128.61.23.0/24`), second connecting the two routers together (network address: `128.61.21.0/24`), and the third connecting the top 3 hosts on the upper LAN with the top router (network address: `128.61.22.0/24`).

Circuit Switching vs Packet Switching

Circuit Switching is analogous to making reservations for trips at every leg of your journey

- If you don't show up at any leg of your journey, your ticket goes to someone else



Packet switching is analogous to just showing up at every leg of your journey

- You might have to wait for empty seats
- This is why we might have packet loss
- ***Datagram*** service model = every packet is individually stamped with the destination address, and the routers en route look at the address to decide a plausible route to reach the destination
- ***Virtual Circuit*** service model = establish a route from source to destination during the call set up phase (called a virtual circuit)
    - Source gets a virtual circuit number, which it can use for the duration of the call to send packets along the virtual circuit
    - Switches en route maintain a table (let us call it VC table) that contains information pertaining to each virtual circuit currently managed by the switch, namely, incoming link, outgoing link
    - Once the message transmission is complete, the network layer at the source executes a call teardown that deletes the corresponding entry in each of the switches en route to the destination

Data Forwarding

Routing is akin to the decision of which path to take and forwarding is the act of doing it

When the transport layer gives it a packet, the network layer determines the next hop the packet must take in its journey towards the ultimate destination

- This is called forwarding
- Forwarding table contains the next hop IP address given a destination IP address

13.8 Link Layer and Local Area Network

Link layer is responsible for acquiring the physical medium for transmission, and sending the packet over the physical medium to the destination host

- Deals with frames instead of packets
- Protocols fall into two categories
    - Random access
    - Taking turns

The part of the link layer protocol that deals with gaining access to the physical medium is usually called Media Access and Control (MAC for short) layer.

Ethernet

Arbitration logic decides who has control of the bus when multiple units require simultaneous access to the bus

Ethernet connects an arbitrary number of units together in an office setting over several hundreds of meters. Thus, the designers of Ethernet had to think of some other way of arbitrating among units competing to use the medium simultaneously to deal with the twin problems of large distances and arbitrary number of units

CSMA/CD

Carrier Sense Multiple Access/Collision Detection

- Tries to counter problems of distance and arbitrary number of devices being connected to the Ethernet cable
- We would look around to make sure no one is talking; then we will start to talk; if two or more people try to say something at the same time we will shut up and try again when no one else is talking

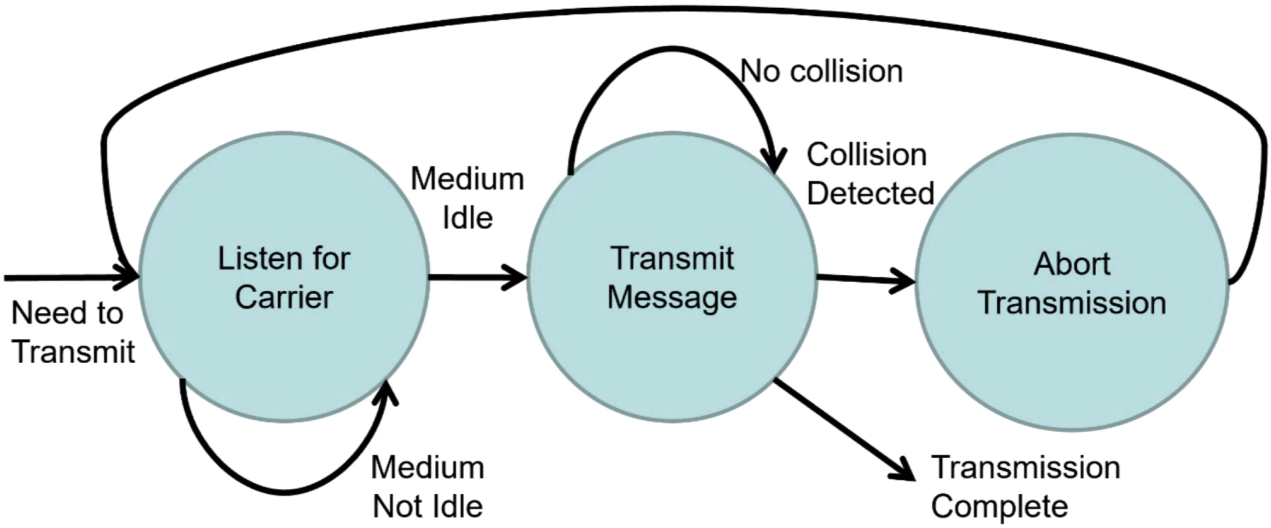


Figure 13.13: State Transitions in CSMA/CD

1. A *station* (i.e computer) wanting to transmit *senses* the *medium* (i.e., the cable) whether there is an ongoing frame transmission. If yes, then wait until idle then start. If idle, start transmission immediately. Absence of any electrical activity = idleness
2. Multiple stations sense the cable simultaneously for idleness. Simultaneously, multiple units may come to the same conclusion that the medium is idle, and start their respective frame transmissions. This is a problem.
3. **Collision Detection:** Each station, after starting a frame transmission, listens for a collision. If what it observes (via listening) is different from this activity, then it knows that some other station has also assumed idleness of the medium. The transmitting station immediately aborts the transmission, and sends out a noise burst to warn all other stations that a collision has occurred. The station then waits for a *random amount* of time before repeating the cycle of sense, transmit, and observe for collision. This random amount grows exponentially (***exponential backoff***) with more and more collisions encountered

Collision domain: set of computers that can hear the transmissions of one another

The protocol uses *base band signaling*, i.e., the frame is transmitted digitally on the medium (i.e., the cable) directly as 0's and 1's.

Broadband refers to simultaneous analog transmission of multiple messages on the same medium

- Different services may use different frequencies

IEEE 802.3

Ethernet uses a specific standard of the CSMA/CD protocol called IEEE 802.3

- digital frame transmission uses Manchester code
- logic 0 as a low followed by a high transition
- logic 1 as a high followed by a low transition
- low signal is -0.85 volts and high signal is +0.85 volts and idle is 0 volts

Manchester coding technique ensures that there is a voltage transition in the middle of each bit transmission thus enabling synchronization of the sender and receiver stations

- there is always electrical activity when there is a frame transmission on the wire
- If there is no voltage transition in the duration of a bit transmission then a station assumes the medium is idle
- Since Ethernet uses *base band signaling technique*, there can only be one frame at a time on the medium

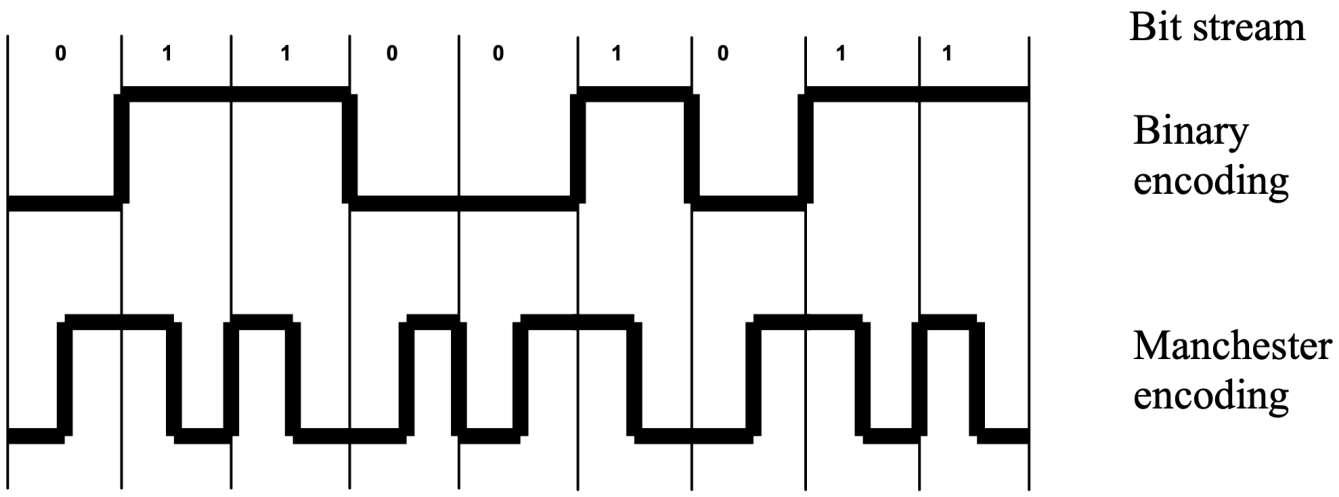


Figure 13.14: Manchester Encoding

While early Ethernet used CSMA/CD, most Ethernet LANs in use today are *switched Ethernets*

- No collision in switched ethernet
- Ethernet as a link layer protocol continues to use the same wire format (i.e., frame headers) for compatibility

Wireless LAN and IEEE 802.11

We introduce CSMA/CA

- CA stands for collision avoidance
- Stations cannot determine if there was a collision on the medium
    - Detecting collisions assumes that a station can simultaneously send its own transmission and receive to verify if its transmissions is being interfered with by transmissions from other stations
    - Easy to detect in wired networks, hard in wireless

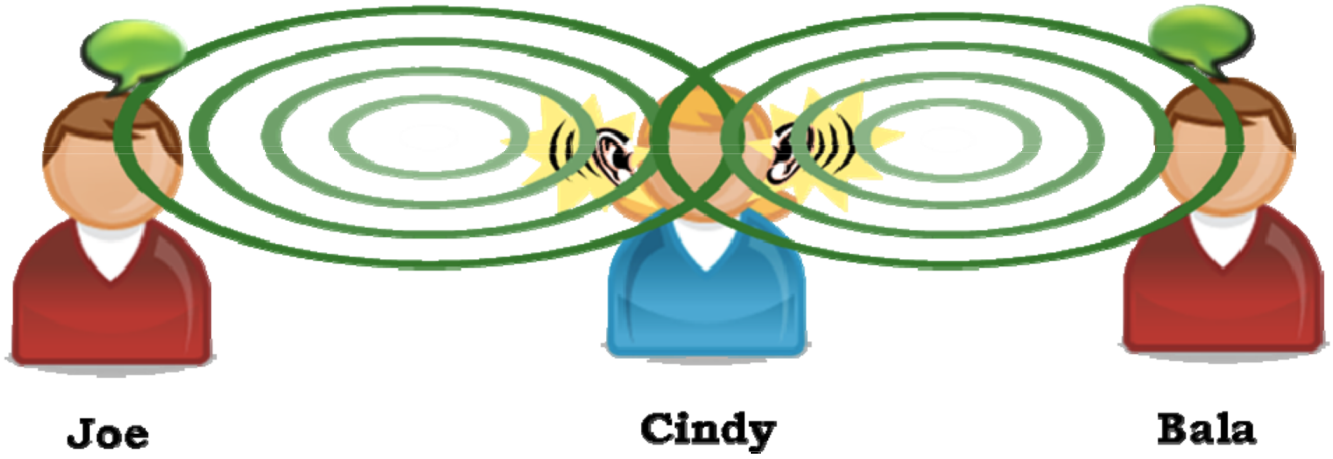


Figure 13.15: Hidden Terminal Problem

Cindy can hear Joe and Bala; Joe can hear Cindy; Bala can hear Cindy; however, ***Joe and Bala cannot hear each other*** - Joe may try to talk to Cindy simultaneous with Bala talking to Cindy. There will be a collision at Cindy but neither Bala nor Joe will realize this

To avoid collisions, we send a short ***Request To Send (RTS)*** control packet on the medium to the desired destination

- Destination responds with a ***Clear To Send (CTS)*** control packet

Token Ring

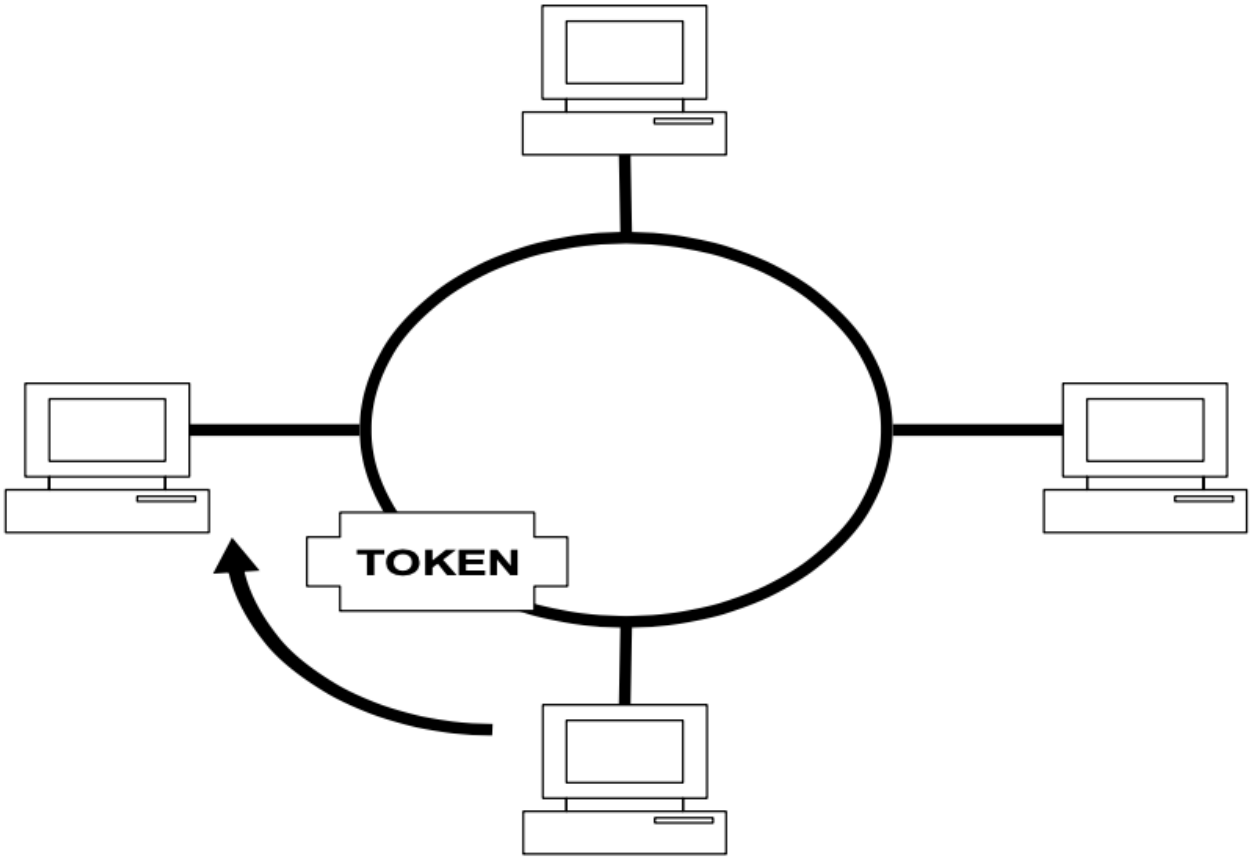


Figure 13.16: Token Ring

A ***token***, a special bit pattern, keeps circulating on the wire

- A node that wishes to send a frame, waits for the token, grabs it, puts its frame on the wire, and then puts the token back on the wire
- Each node examines the header of the frame
- If the node determines that it is the destination for this frame then the node takes the frame
- Somebody has to be responsible for removing the frame and regenerating the token
    - Usually, it is the sender of the frame that should remove the frame and put pack the token on the wire
    - If a token ring network spans a geographically large area, it would make sense for the destination node to remove the frame and put back the token on the wire

By design, there are no collisions in a Token Ring Network

- Node has to wait for the token to send a frame = high latency

- If a node dies, then the LAN is broken
- If for some reason the token on the ring is lost or corrupted, then the LAN cannot function anymore

Link Layer Protocol	Features	Pros	Cons
Ethernet	Member of random access protocol family; opportunistic broadcast using CSMA/CD; exponential backoff on collision	Simple to manage; works well in light load	Too many collisions under high load
Token Ring	Member of taking turns protocol family; Token needed to transmit	Fair access to all competing stations; works well under heavy load	Unnecessary latency for token acquisition under light load

Other Link Layer Protocol

FDDI

- Fiber Distributed Data Interface
- Originally conceived for fiber-optic physical medium, and was considered especially suitable as a high-bandwidth backbone network for connecting several islands of Ethernet-based LANs together in a large organization such a university campus or a big corporation

ATM

- Asynchronous Mode Transfer
- Ability to provide guaranteed quality of service through bandwidth reservation on the links and admission control to avoid network congestion
- Connection-oriented link layer protocol and provides many of the network layer functionality as well that a transport layer protocol can be directly implemented on top of it

13.9 Networking Hardware

Component	Function
Host	A computer on the network; this is interchangeably referred to as node and station in computer networking parlance
NIC	Network Interface Card (corresponds with layer 2 of OSI model)
Port	End-point on a repeater/hub/switch for connecting a component (corresponds with layer 1)
Collision Domain	Term used to signify the set of computers that can interfere with one another
Repeater	Boosts signal strength on an incoming port and reproduces bit stream on an outgoing port; used in LANs and WANs; corresponds to layer 1 (physical) of the OSI model
Hub	Connects computers together to form a single collision domain, serving as a multi-port repeater; corresponds to layer 1 (physical) of the OSI model
Bridge	Connects independent collision domains, isolating them from one another; typically 2-4 ports; uses MAC addresses to direct the message on an incoming port to an outgoing port; corresponds to layer 2 (data link) of the OSI model
Switch	Similar to a bridge but works for several ports, corresponds to layer 2 of OSI model
Router	Essentially a switch but has expanded capabilities to route a message from the LAN to the Internet; corresponds to layer 3 (network) of the OSI model
VLAN	Virtual LAN; capabilities in modern switches allow grouping computers that are physically distributed and connected to different switches to form a LAN; VLANs make higher level network services such as broadcast and multicast in Internet subnets feasible independent of the physical location of the computers; corresponds to layer 2 (data link) of the OSI model

13.11 Data Structures For Packet Transmission

SYN: This signals the start of a new byte-stream allowing the two endpoints to synchronize their starting sequence number for the transmission
FIN: This signals the end of transmission of this byte-stream
ACK: This signals that the header includes a piggy-backed ACK, and therefore the acknowledgement number field of the header is meaningful
URG: This field indicates that there is "urgent" data in this segment. For example, if you hit Ctrl-C to terminate a network program, the application level protocol will translate that to an urgent message at the TCP level.

13.12 Message Transmission Time

1. **Processing delay at the Sender** (S): This is the cumulative time spent at the sender in the various layers of the protocol stack and includes
2. **Transmission delay** (T_w): This is the time needed to put the bits on the wire at the sending end
3. **Time of flight** (T_f): This term simplifies a number of things going on in the network from the time a message is placed on the wire at the sending end to the time it is finally in the network interface at the receiving end
4. **Processing Delay at the Receiver** (R): This is the mirror image of the processing delay incurred by the sender and has all the elements of that delay due to the physical, link, network, and transport layers.

$$\text{Total time for transmission} = S + T_w + T_f + R$$

$$\text{Throughput} = \text{message size} \div \text{end to end latency}$$

13.15 Network Programing Using Unix Sockets

The socket creation call on Unix takes on three parameters domain, type, protocol.

- Domain: this parameter helps pick the protocol family to use for the communication
- Type: this parameter specifies the application property desired such as datagram or stream
- Protocol: this parameter specifies the protocol that belongs to the protocol family (given by the domain parameter) that satisfies the desired property (given by the type parameter)

Interprocess communication using sockets follow a client/server paradigm

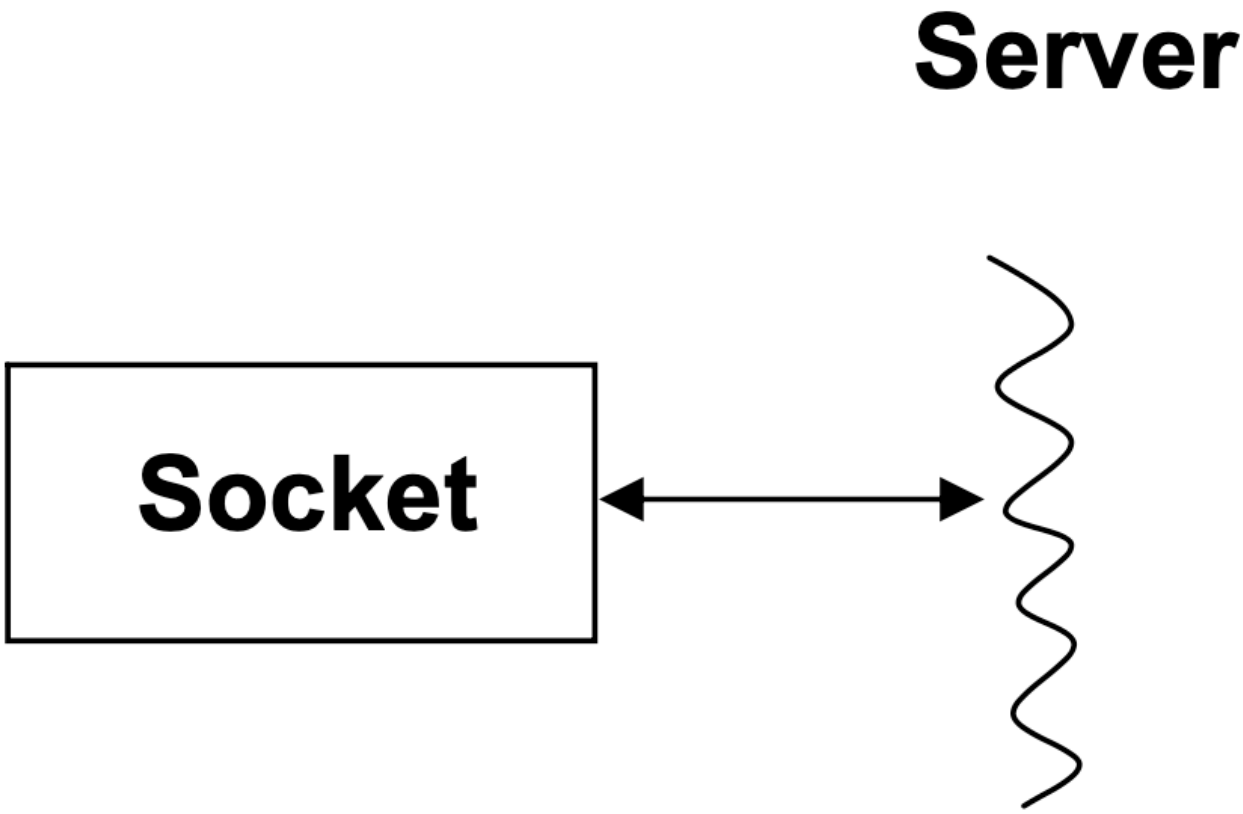


Figure 13.15: Server side Socket call

Procedure For Server

1. Create a socket for communication
2. The system call to associate a name with a socket is `bind`. The name has two parts:

```
<host address, port number>
```

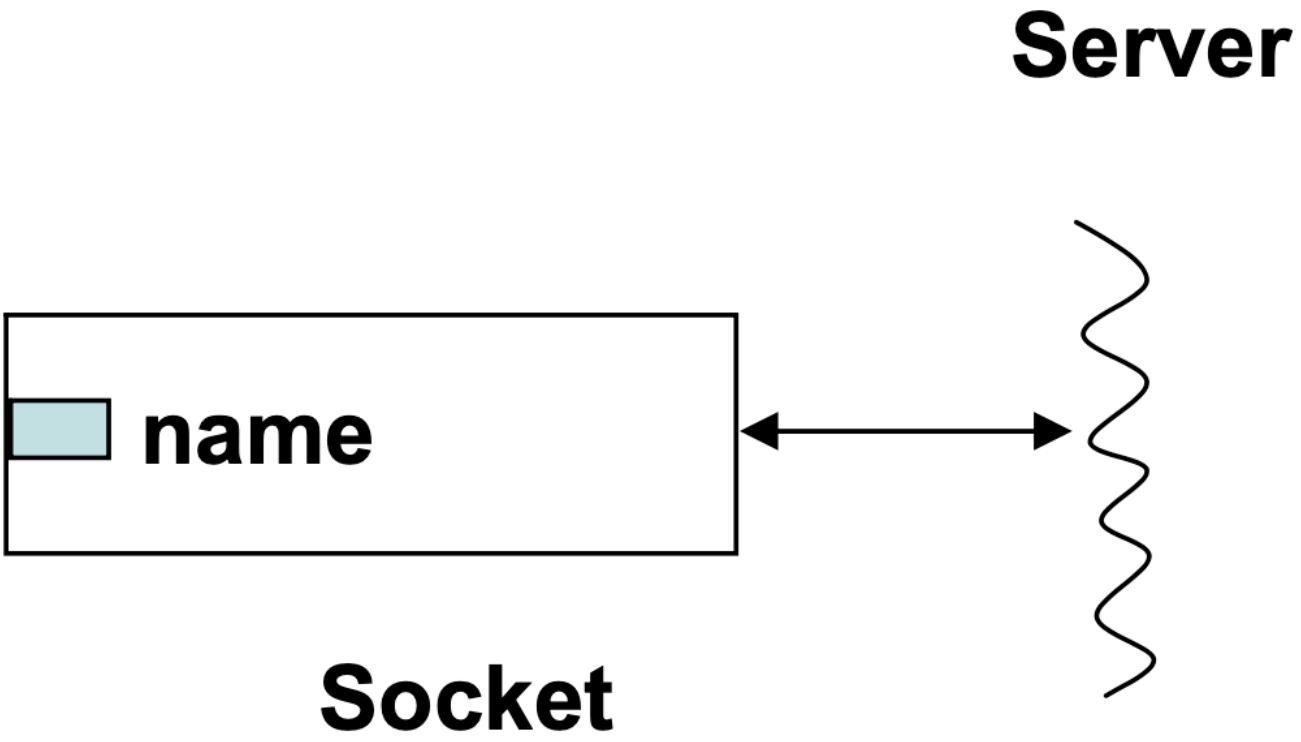


Figure 13.16: Server side Socket After Bind

3. Execute `listen` on the server socket

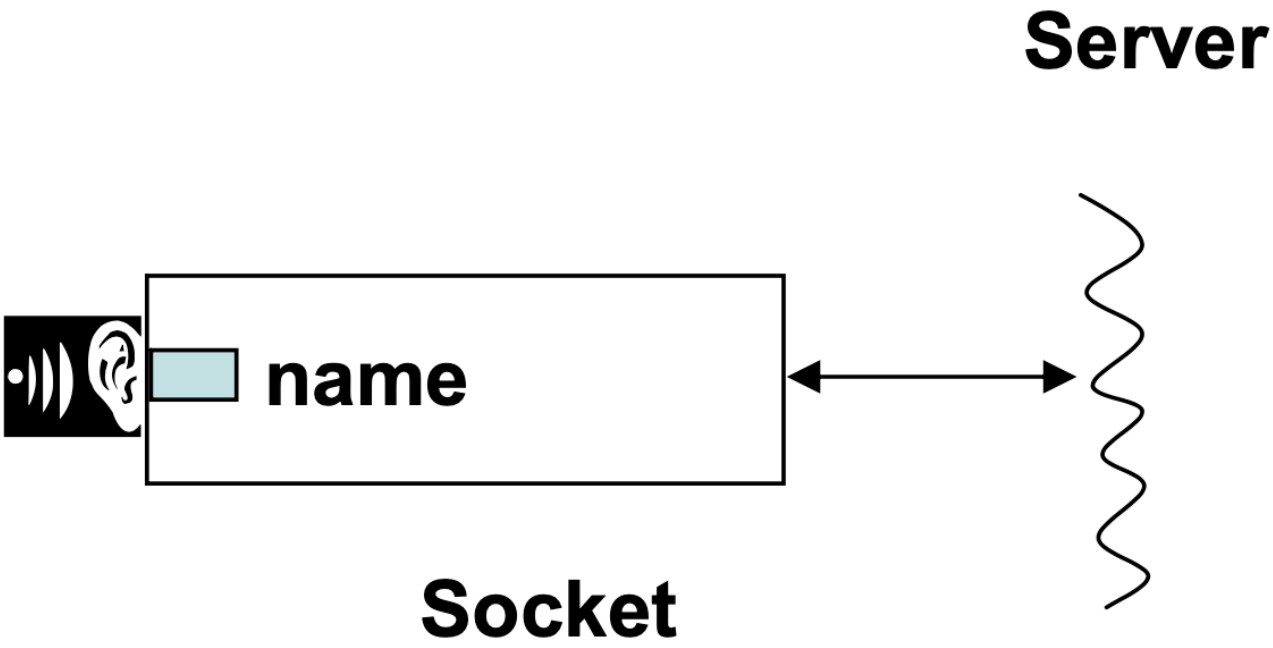


Figure 13.17: Server side Socket After Listen

4. Complete reception of an incoming request using `accept`

Procedure For Client

1. Create a client side socket
2. Call `connect` to the server `name`

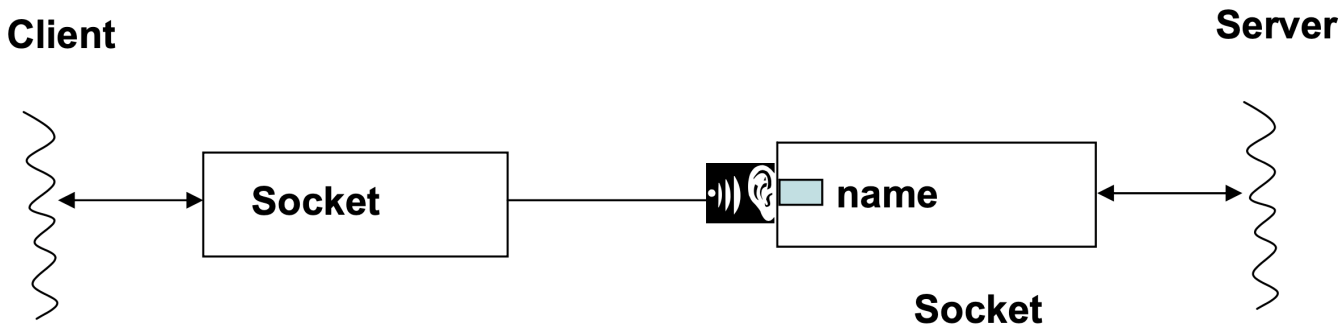


Figure 13.18: Client-Server relationship

Once accepted, the OS creates a new *data socket*

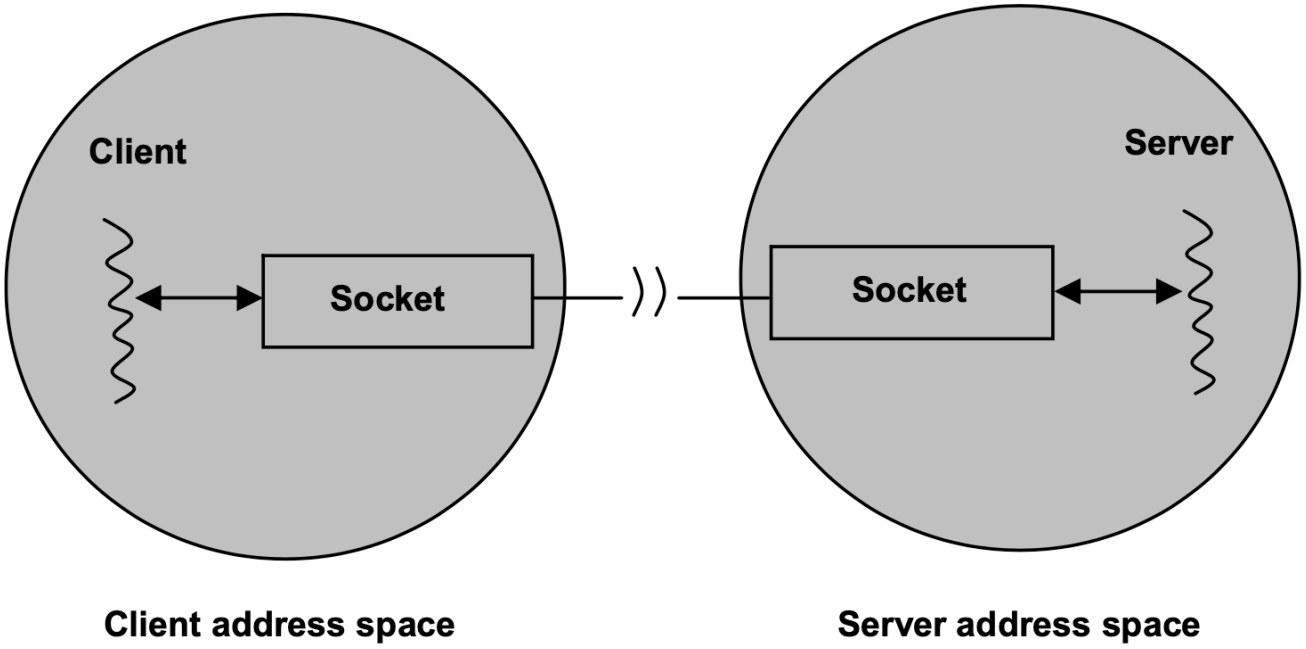


Figure 13.19: Sockets in Client-Server Address spaces

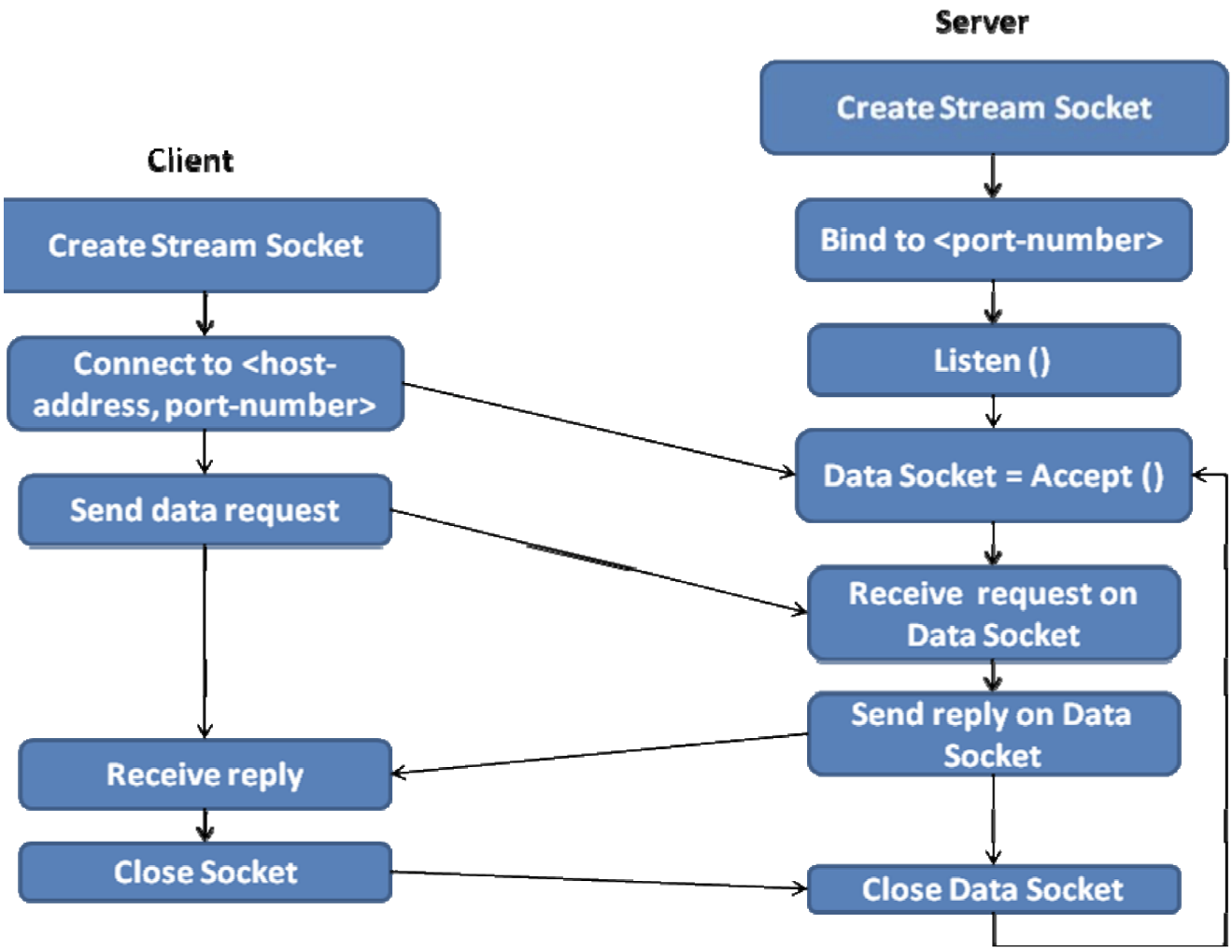


Figure 13.20: Data Communication on a Stream Socket

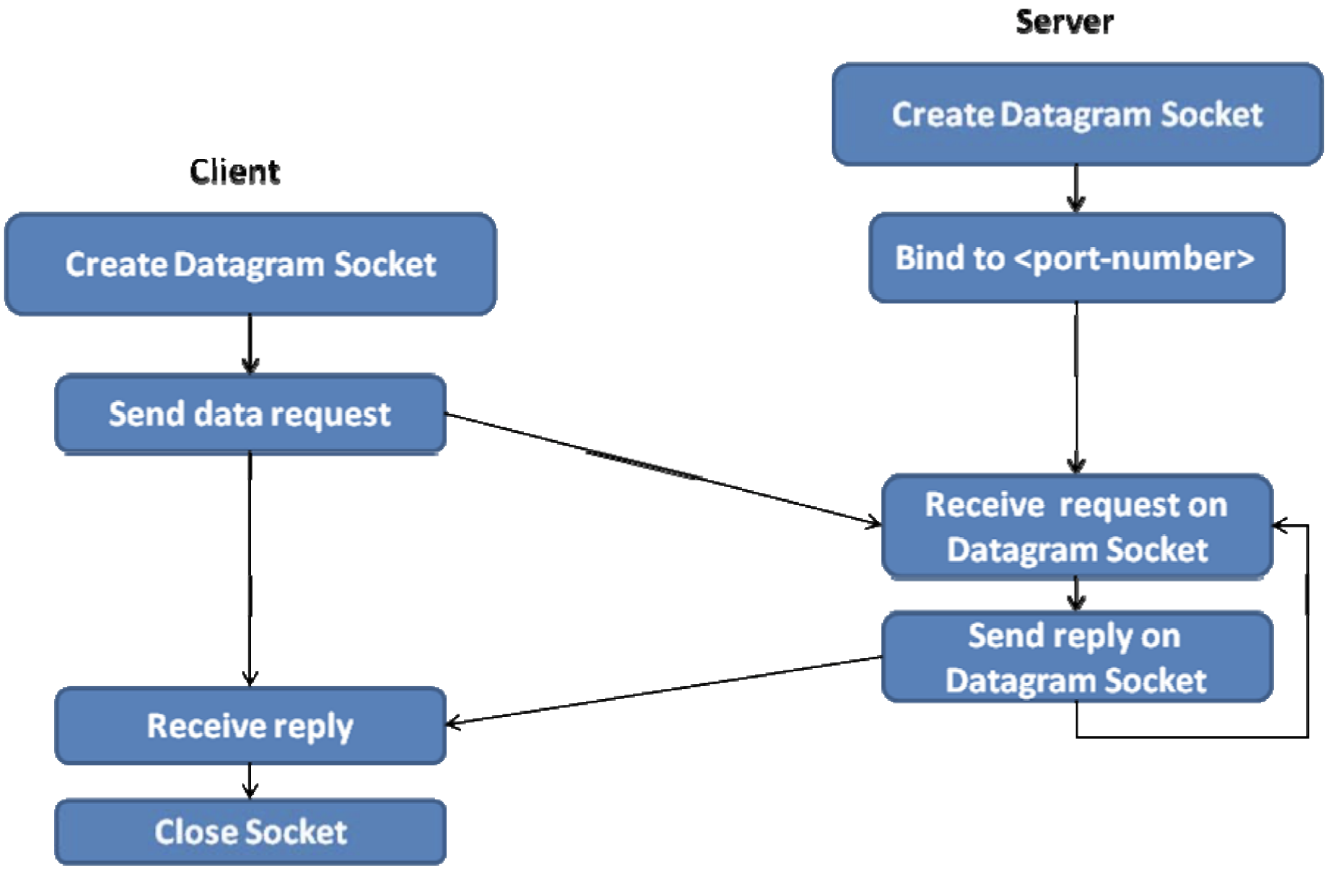


Figure 13.20: Data Communication on a Datagram Socket