COMPUTER NETWORKS

Sivaraman Eswaran Ph.D.
Department of Computer Science and Engineering

COMPUTER NETWORKS

Application Layer

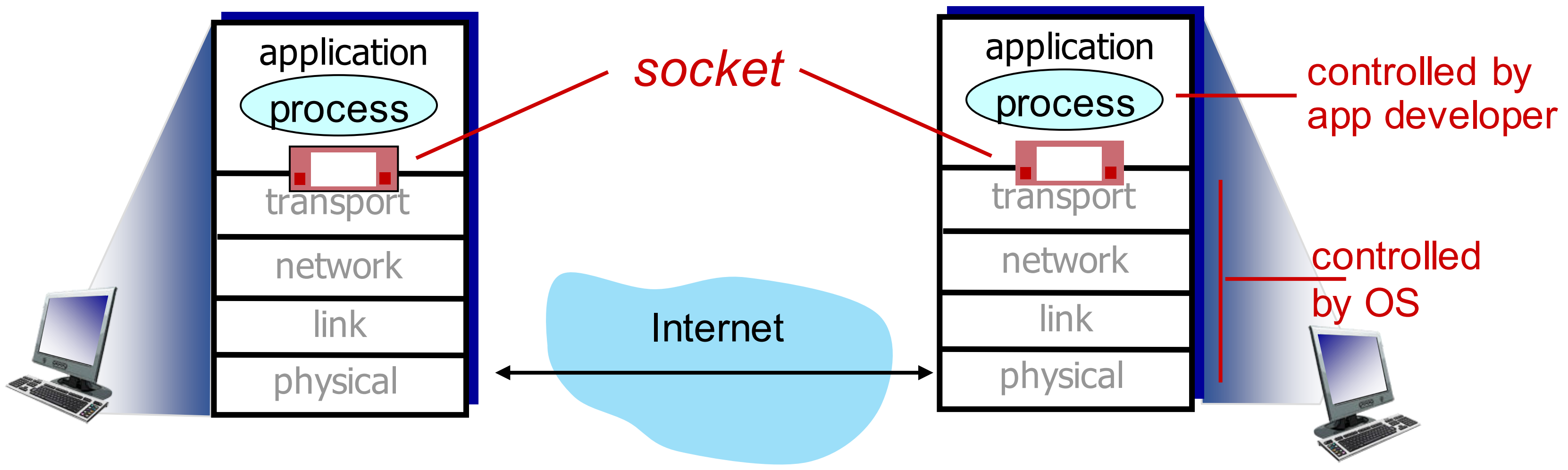
Sivaraman Eswaran Ph.D.
Department of Computer Science and Engineering

Unit – 2 Application Layer

Socket Programming

goal: learn how to build client/server applications that communicate using sockets

socket: door between application process and end-end-transport protocol



Two socket types for two transport services:

- *UDP:* unreliable datagram
- *TCP:* reliable, byte stream-oriented

Application Example:

1. client reads a line of characters (data) from its keyboard and sends data to server
2. server receives the data and converts characters to uppercase
3. server sends modified data to client
4. client receives modified data and displays line on its screen

Socket Programming with UDP

UDP: no "connection" between client & server

- no handshaking before sending data
- sender explicitly attaches IP destination address and port # to each packet
- receiver extracts sender IP address and port# from received packet

UDP: transmitted data may be lost or received out-of-order

Application viewpoint:

- UDP provides *unreliable* transfer of groups of bytes ("datagrams") between client and server

Client/Server socket interaction: UDP

server (running on serverIP)

create socket, port= x:
serverSocket =
socket(AF_INET,SOCK_DGRAM)

read datagram from
serverSocket

write reply to
serverSocket
specifying
client address,
port number

client

create socket:
clientSocket =
socket(AF_INET,SOCK_DGRAM)

Create datagram with server IP and
port=x; send datagram via
clientSocket

read datagram from
clientSocket

close
clientSocket

COMPUTER NETWORKS

Example app: UDP client

Python UDPClient

include Python's socket library

```
from socket import *
serverName = ‘hostname’
serverPort = 12000
```

create UDP socket for server

```
clientSocket = socket(AF_INET,
                      SOCK_DGRAM)
```

get user keyboard input

```
message = raw_input(’Input lowercase sentence:’)
```

attach server name, port to message; send into socket

```
clientSocket.sendto(message.encode(),
                    (serverName,  serverPort))
```

read reply characters from socket into string

```
modifiedMessage, serverAddress =
                    clientSocket.recvfrom(2048)
```

print out received string and close socket

```
print modifiedMessage.decode()
clientSocket.close()
```

Example app: UDP server

Python UDPServer

```
                                                            from socket import *
                                                            serverPort = 12000
                                  create UDP socket  →      serverSocket = socket(AF_INET, SOCK_DGRAM)
            bind socket to local port number 12000  →      serverSocket.bind(('', serverPort))
                                                            print ("The server is ready to receive")
                                       loop forever  →      while True:
Read from UDP socket into message, getting           →          message, clientAddress = serverSocket.recvfrom(2048)
client's address (client IP and port)
                                                                modifiedMessage = message.decode().upper()
      send upper case string back to this client     →          serverSocket.sendto(modifiedMessage.encode(),
                                                                                    clientAddress)
```

PES UNIVERSITY ONLINE

THANK YOU

Sivaraman Eswaran Ph.D.
Department of Computer Science and Engineering

sivaramane@pes.edu

+91 80 6666 3333 Extn 834