Lecture # 7.3
Socket Programming - II
Internet Domain UDP Sockets

Course: Advance Operating System

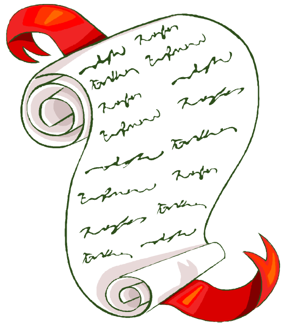
Instructor: Arif Butt

Punjab University College of Information Technology (PUCIT)
University of the Punjab

Source Code files available at: **`https://bitbucket.org/arifpucit/spvl-repo/src`**
Lecture Slides available at: **`http://arifbutt.me`**

Today's Agenda

- Recap of TCP Sockets
- What are Datagram Sockets?
- How Datagram Sockets Work?
- System Call Graph for UDP Sockets
- BSD UNIX Socket API
- Writing a UDP `echo` Client (connected / unconnected)
- Writing a UDP daytime Client (connected / unconnected)
- Writing a UDP `time` Client (connected / unconnected)
- Writing a UDP `echo` Server

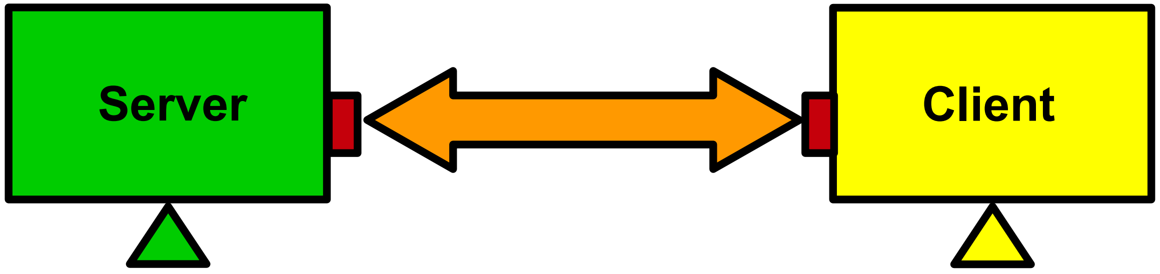


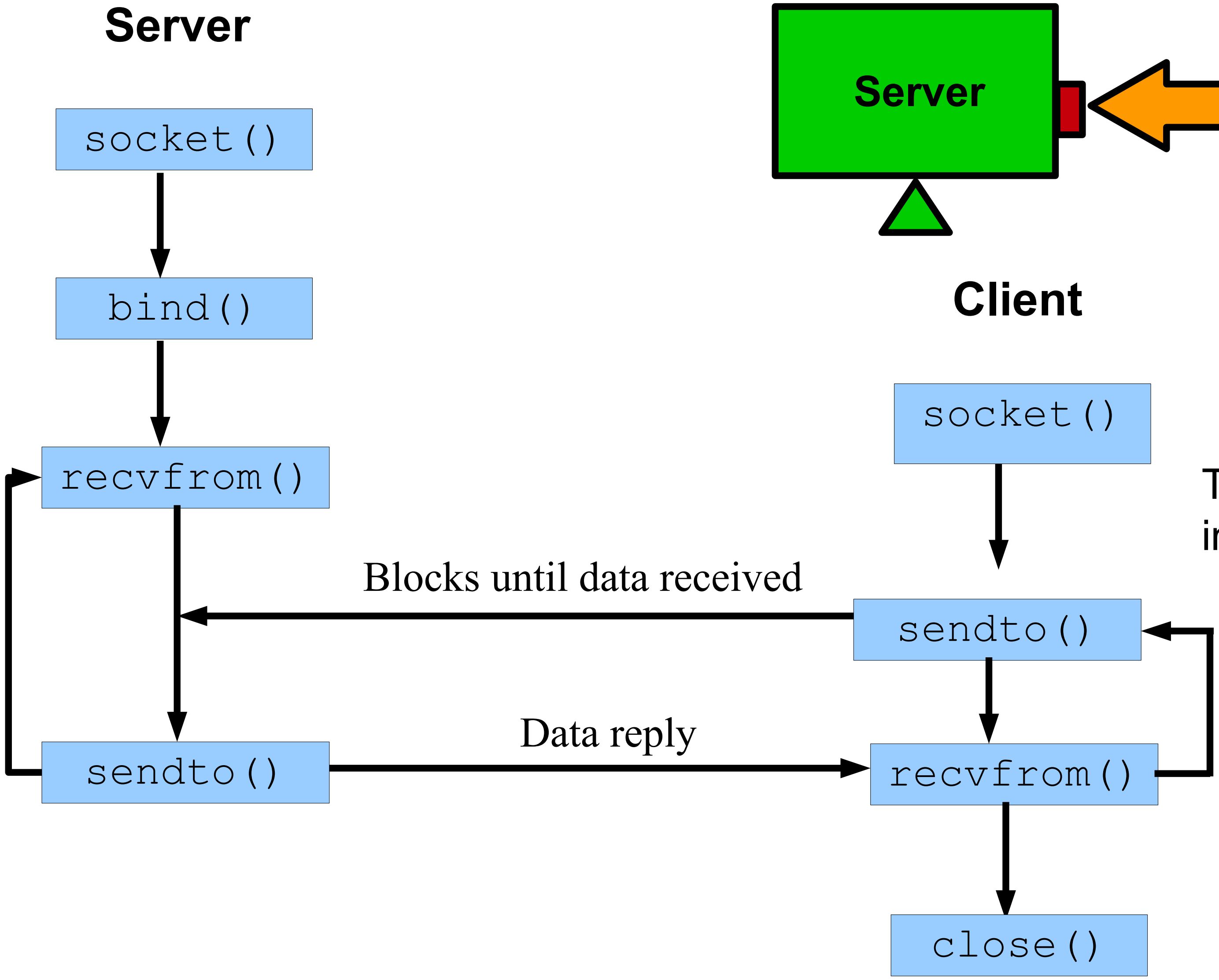
Datagram Sockets

- **Datagram sockets (SOCK_DGRAM)** provide *unreliable*, *full-duplex*, *packet-oriented* communication channel
- Messages are transmitted and received in terms of datagrams, which is a small, fixed-length packet
- A process either reads the entire message or in case of error does not read any data
- Unreliable means there is no sequence numbering, so messages may arrive out of order, be duplicated or not arrive at all
- Datagram sockets can be connected as well as unconnected

How Datagram Sockets Work? Behind the curtain

System Call Graph: UDP Sockets





The client may call `connect()` in case of connected UDP socket

Datagram Sockets

- **Connectionless Datagram Sockets:** Mostly the datagram sockets are connection-less, i.e., the client do not call `connect()`, rather address every message using `sendto()` and `recvfrom()` calls. It is possible for a process to call `sendto()` to different server processes

- **Connection-Oriented Datagram Sockets:** If a UDP client calls `connect()` and specifies the UDP server address, the socket is said to be connected. From that point onward, the client may only send to and receive from the address specified by connect. So you don't have to use `sendto()` and `recvfrom()`. You can simply use `send()` and `recv()` calls. Normally UDP sockets do not use connect. However, UDP sockets may use `connect()` multiple times to change their association

Pseudocode: UDP / Datagram Sockets

Server

```
socket()
bind()
while(1){
    read a request from some client
    send a reply to that client
}
```

Client

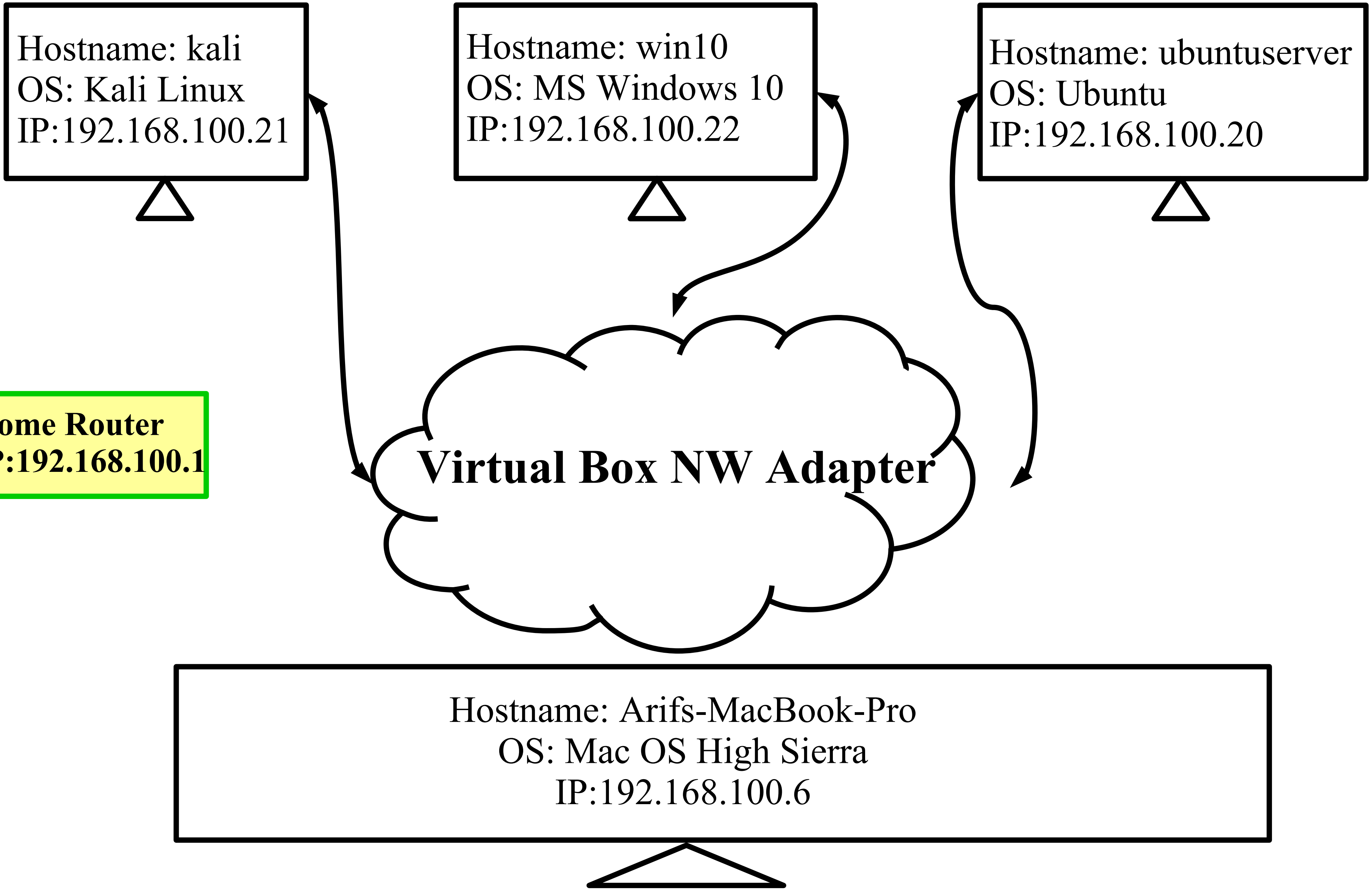
```
socket()
sendto()
recvfrom()
close()
```

Stream vs Datagram Sockets

TCP Stream Sockets	UDP Datagram Sockets
Fragment/Reassemble	No
Ordering	No
Reliable	May not arrive
Connected	Multiple senders

- **Datagram Sockets**, therefore, place less load on kernel network code and on NW traffic. Datagrams may get lost in transit, and they may arrive out of order. For these two reasons, datagram sockets are best suited to applications in which simplicity, efficiency, and speed are more important than data integrity and consistency. They are a bad choice for web, file or email servers. As they can be large documents. They are good choice for streams of music and video where a missing note or frame may not even be noticed

Lab Scenario



Home Router
IP:192.168.100.1

Internet Domain UDP Sockets

`udpechoclient.c`

(connected / unconnected)

sendto()

```
ssize_t sendto(int sockfd,const void *buf,size_t count,int flags,
              const struct sockaddr *dest_addr, socklen_t addrlen);
```

- Since datagram sockets are mostly not connected to a remote host, we need to give the destination address before we send a packet
- This call is basically the same as the call to `send()` with the addition of two other pieces of information
- The argument **dest_addr** is a pointer to a `struct sockaddr` which contains the destination IP address and port, where the calling process will send the data
- The argument **addrlen** can simply be set to `sizeof(struct sockaddr)`
- Just like with `send()`, `sendto()` returns the number of bytes actually sent (which again, might be less than the number of bytes you told it to send), or -1 on error

recvfrom()

```
int recvfrom(int sockfd, void* buf, int count, int flags, struct sockaddr* from, socklen_t* fromlen);
```

- Since datagram sockets are mostly not connected to a remote host, we need to give the address from which we want to receive a packet
- This call is basically the same as the call to `recv()` with the addition of two other pieces of information
- The last two arguments are used to obtain the address of the sender. The address of the sender's socket will be stored in **from,** and the length of that address will be stored in the integer pointed to by the last argument **fromlen**. If we are not interested in the address of the sender, then we specify both **from** and **fromlen** as NULL
- Just like with `send()`, `sendto()` returns the number of bytes actually received (which again, might be less than you requested in the **fromlen** parameter), or -1 on error
- If the remote side has closed the connection, `recvfrom()` will return 0

Internet Domain UDP Sockets

`udpdaytimeclient.c`

Internet Domain UDP Sockets

udptimeclient.c

Internet Domain UDP Sockets

`udpechoserver.c`

Assignment: Design and Code of `nc(1)`

Things To Do

If you have problems visit me in counseling hours. . . .