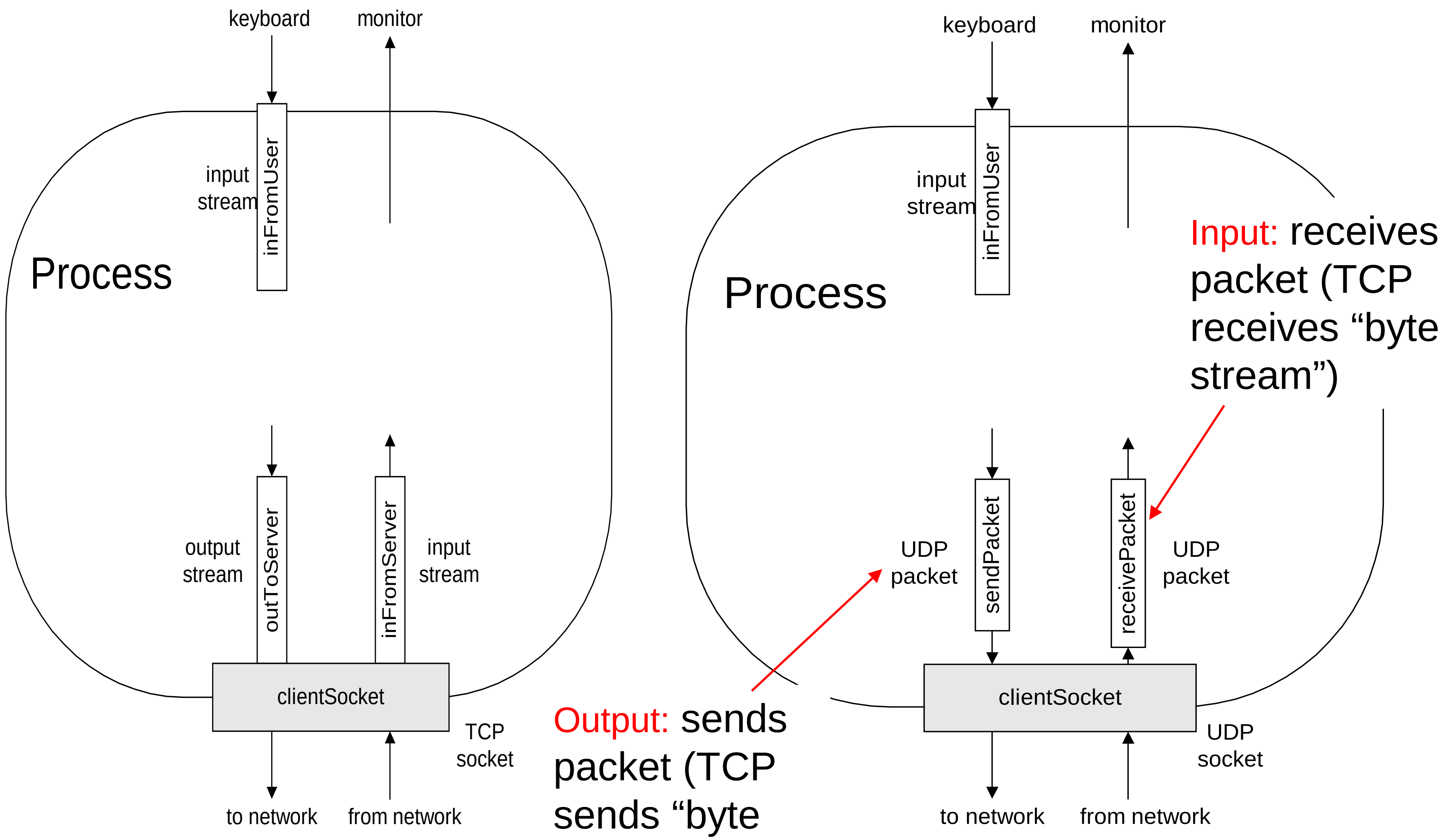
CPSC 441
UDP Socket Programming

Department of Computer Science
University of Calgary

TCP Vs UDP Socket



Input: receives packet (TCP receives “byte stream”)

Output: sends packet (TCP sends “byte stream”)

What this means in terms of programming so far..

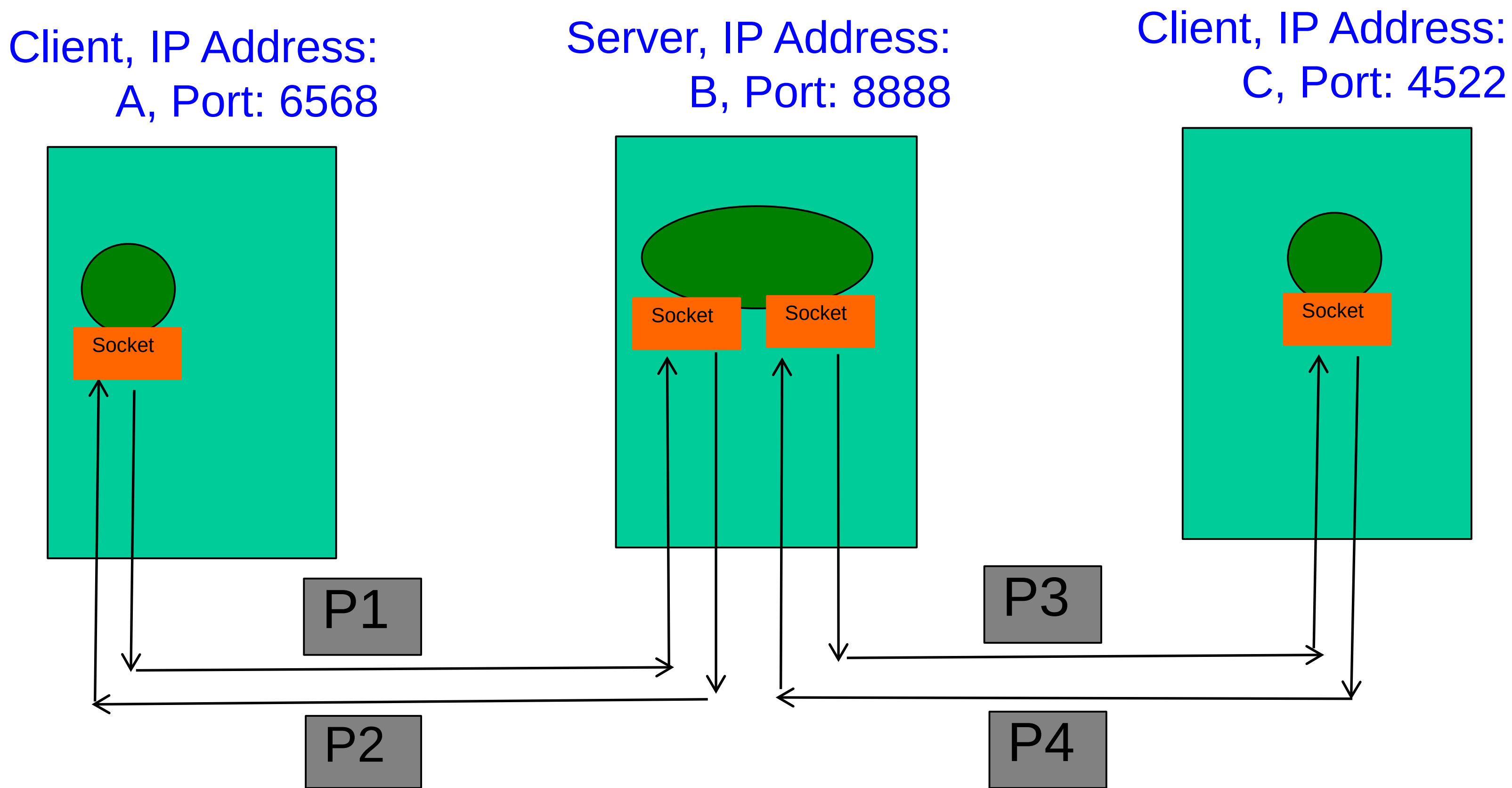
- No read loop as follows!

```
while(byteread != -1)
{

          byteread = socket.getInputStream().read();
          .
          .
          .
          .

 }
```

TCP Socket Revisited

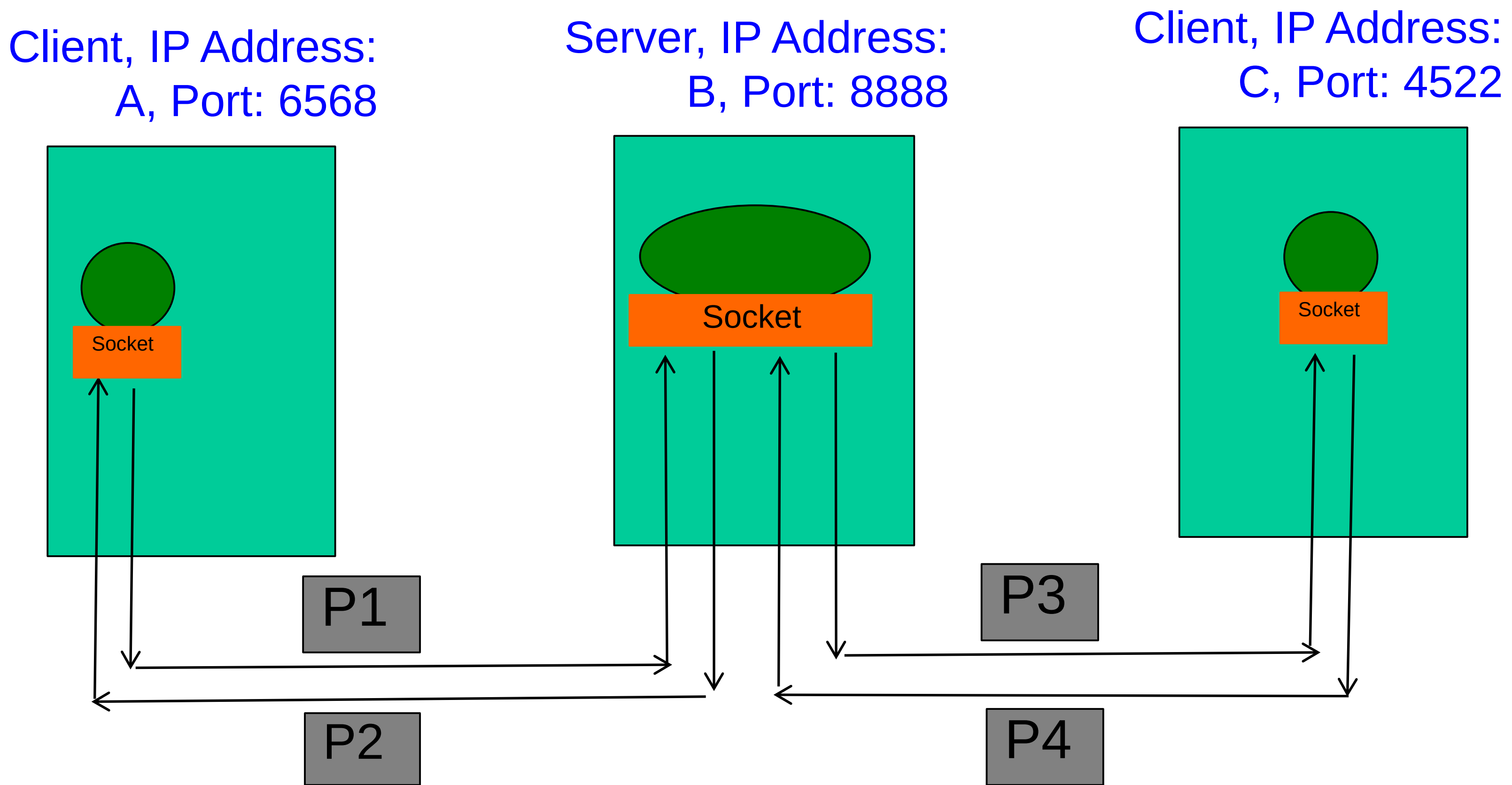
Client port number is dynamically allocated

- Consider the multi-threaded server we discussed earlier
- Find the source/destination IP addresses and port numbers of packets P1, P2, P3 and P4


Client, IP Address: A, Port: 6568
Server, IP Address: B, Port: 8888
Client, IP Address: C, Port: 4522
Socket
Socket
Socket
Socket
P1
P2
P3
P4

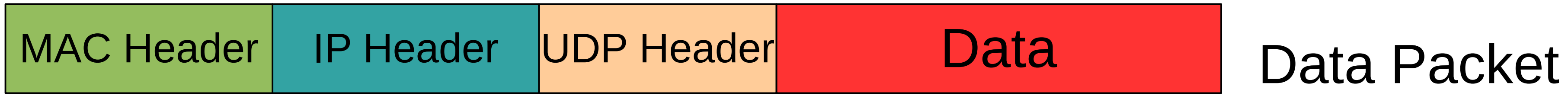
UDP Socket

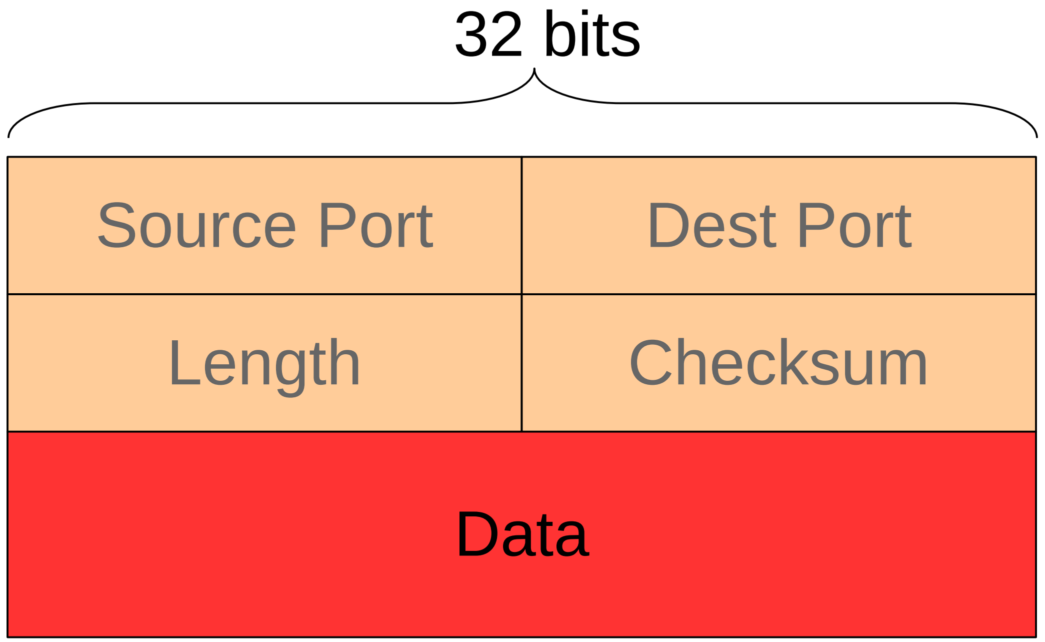
- A single socket at server end
- It is the responsibility of server application to differentiate messages from different clients



Client, IP Address: A, Port: 6568
Server, IP Address: B, Port: 8888
Client, IP Address: C, Port: 4522
Socket
Socket
Socket
P1
P2
P3
P4

UDP – Packet Format





- 2 bytes for source/destination ports (0-65536)
- Length (bytes) = header + data
- Checksum of header and data
- Data = variable length
  - multiple of 4 bytes, padding done by kernel

Java Classes

- DatagramSocket
- DatagramPacket

Example: Java client (UDP)

```
import java.io.*;
import java.net.*;

class UDPClient {
    public static void main(String args[]) throws Exception
    {
     try
     {
      BufferedReader inFromUser =
      new BufferedReader(new InputStreamReader(System.in));

      DatagramSocket clientSocket = new DatagramSocket();

      InetAddress IPAddress = InetAddress.getByName(”localhost");

      byte[] sendData = new byte[1024];
      byte[] receiveData = new byte[1024];

      String sentence = inFromUser.readLine();

      sendData = sentence.getBytes(“US-ASCII”);
```

Create input stream

Create client socket

Translate hostname to IP address using DNS

See the difference with TCP!
Only local port considered

Example: Java client (UDP), cont.

Create datagram with data-to-send, length, IP addr, port

Send datagram to server

Read datagram from server

```
DatagramPacket sendPacket =
   new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

clientSocket.send(sendPacket);

DatagramPacket receivePacket =
   new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(receivePacket);

String modifiedSentence =
    new String(receivePacket.getData(), “US-ASCII”);

System.out.println("FROM SERVER:" + modifiedSentence);
clientSocket.close();
}
catch(Exception e)
{
 // handle exception
}
  }
}
```

Example: Java server (UDP)

```
import java.io.*;
import java.net.*;

class UDPServer {
  public static void main(String args[]) throws Exception
    {
      try
     {
      DatagramSocket serverSocket = new DatagramSocket(9876);


      while(true)
        {
          byte[] receiveData = new byte[1024];
          byte[] sendData  = new byte[1024];

         DatagramPacket receivePacket =
          new DatagramPacket(receiveData, receiveData.length);
         serverSocket.receive(receivePacket);
```

Create datagram socket at port 9876

Create space for received datagram

Receive datagram

Example: Java server (UDP), cont

```
          String sentence = new String(receivePacket.getData(), “US-ASCII”);

          InetAddress IPAddress = receivePacket.getAddress();

          int port = receivePacket.getPort();

          String capitalizedSentence = sentence.toUpperCase();

          sendData = capitalizedSentence.getBytes(“US-ASCII”);

          DatagramPacket sendPacket =
             new DatagramPacket(sendData, sendData.length, IPAddress,  port);

          serverSocket.send(sendPacket);
        }
      }
      catch(Exception e)
      { // handle exception
      }
    }

}
```

Get IP addr port #, of sender

Create datagram to send to client

Write out datagram to socket

End of while loop, loop back and wait for another datagram

Summary: Socket programming *with UDP*

UDP: no "connection" between client and server

- no handshaking
- sender explicitly attaches IP address and port of destination to each packet
- server must extract IP address, port of sender from received packet

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

application viewpoint

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