ALGONQUIN COLLEGE

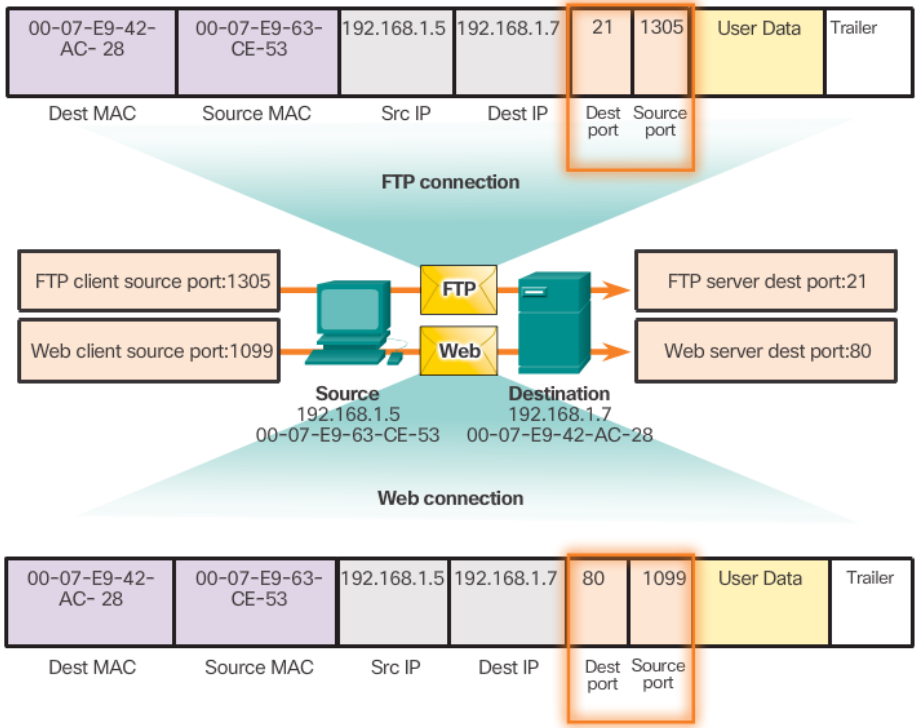
Socket Programming

Objectives

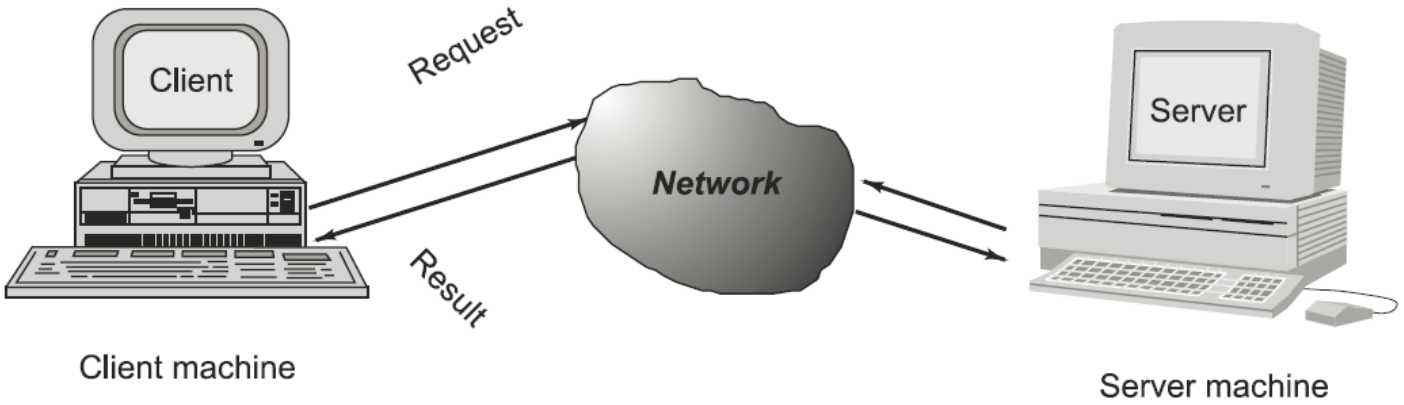
- To explain terms: TCP, IP, stream-based communications, and packet-based communications.
- To create servers using server sockets and clients using client sockets.
- To develop an example of a client/server application.

Socket Pairs

- The combination of the source IP address and source port number, or the destination IP address and destination port number, is known as a socket.
- The socket is used to identify the server and service being requested by the client.
- Two sockets combine to form a soc
192.168.1.7:80)
- Sockets enable multiple processes running on a client and multiple connections to a server process to be distinguished from each other.
- The source port number acts as a return address for the requesting application.
- It is the transport layer's job to keep track of active sockets.

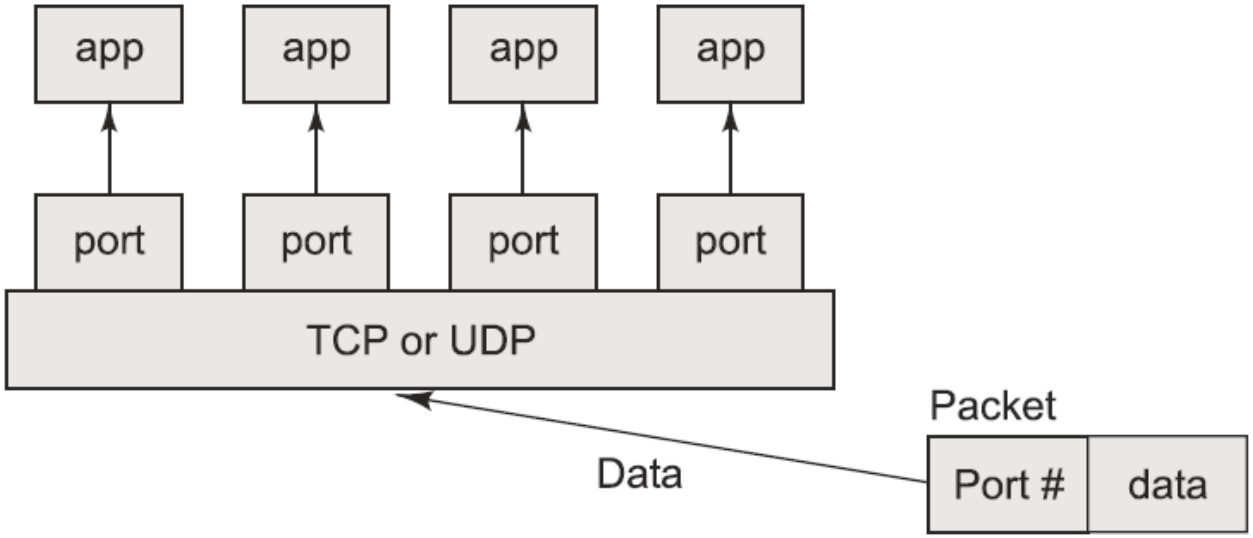


Client/Server Communication



Client – Server Communication

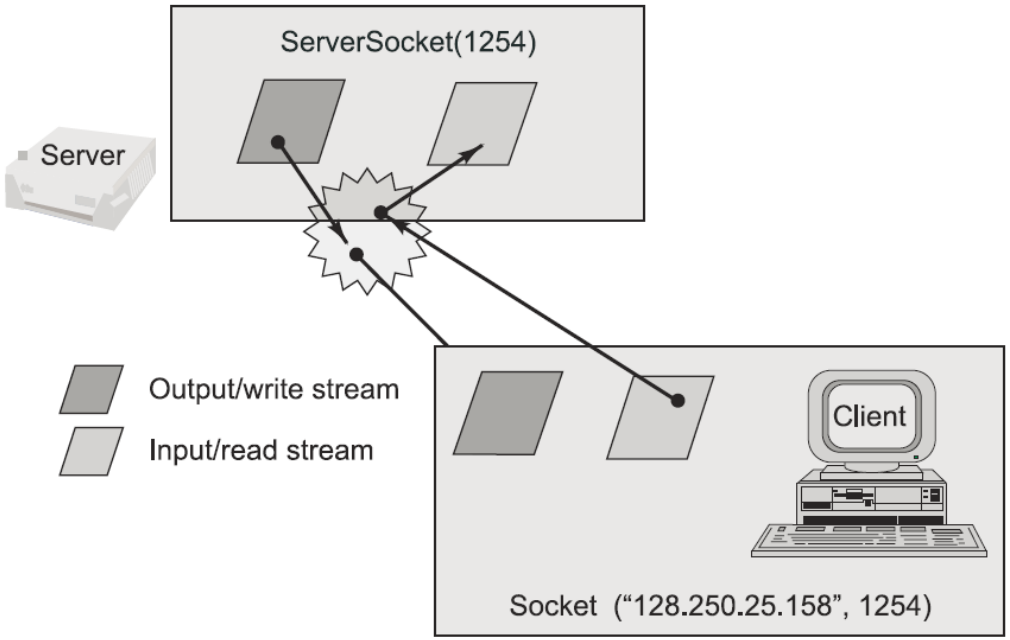
Client/Server Communication



TCP/UDP mapping of incoming packets to appropriate port/process

TCP/IP Socket Programming

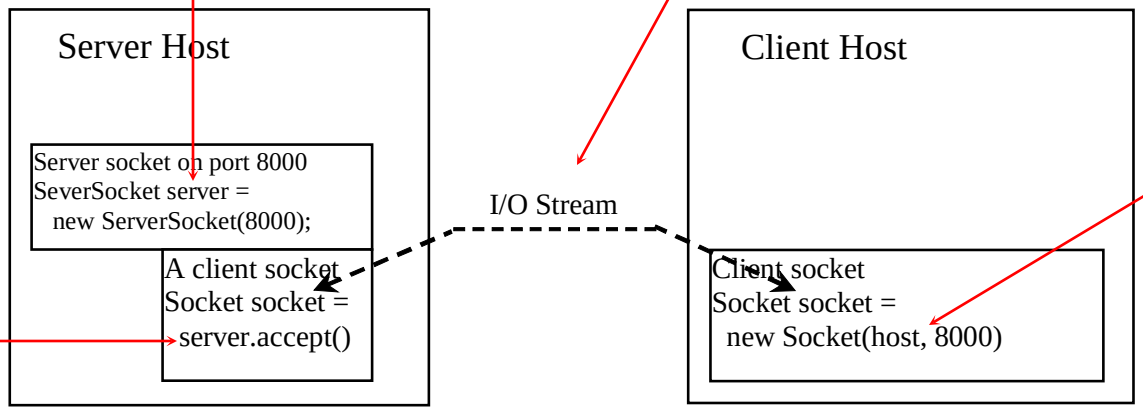
- The two key classes from the `java.net` package used in creation of server and client programs are:
  - `ServerSocket`
  - `Socket`
- Socket-based client and server programming



TCP/IP Socket Programming

The server must be running when a client starts. The server waits for a connection request from a client. To establish a server, you need to create a server socket and attach it to a port, which is where the server listens for connections.

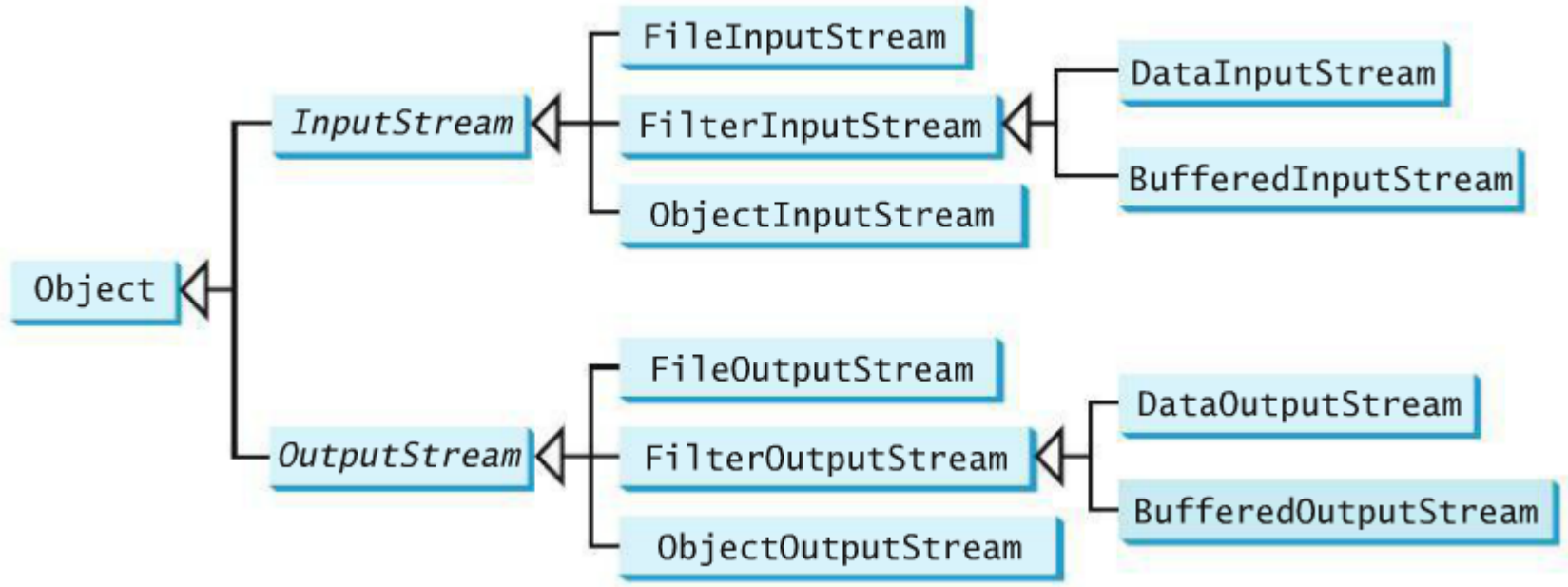
After the server accepts the connection, communication between server and client is conducted the same as for I/O streams.



After a server socket is created, the server can use this statement to listen for connections.

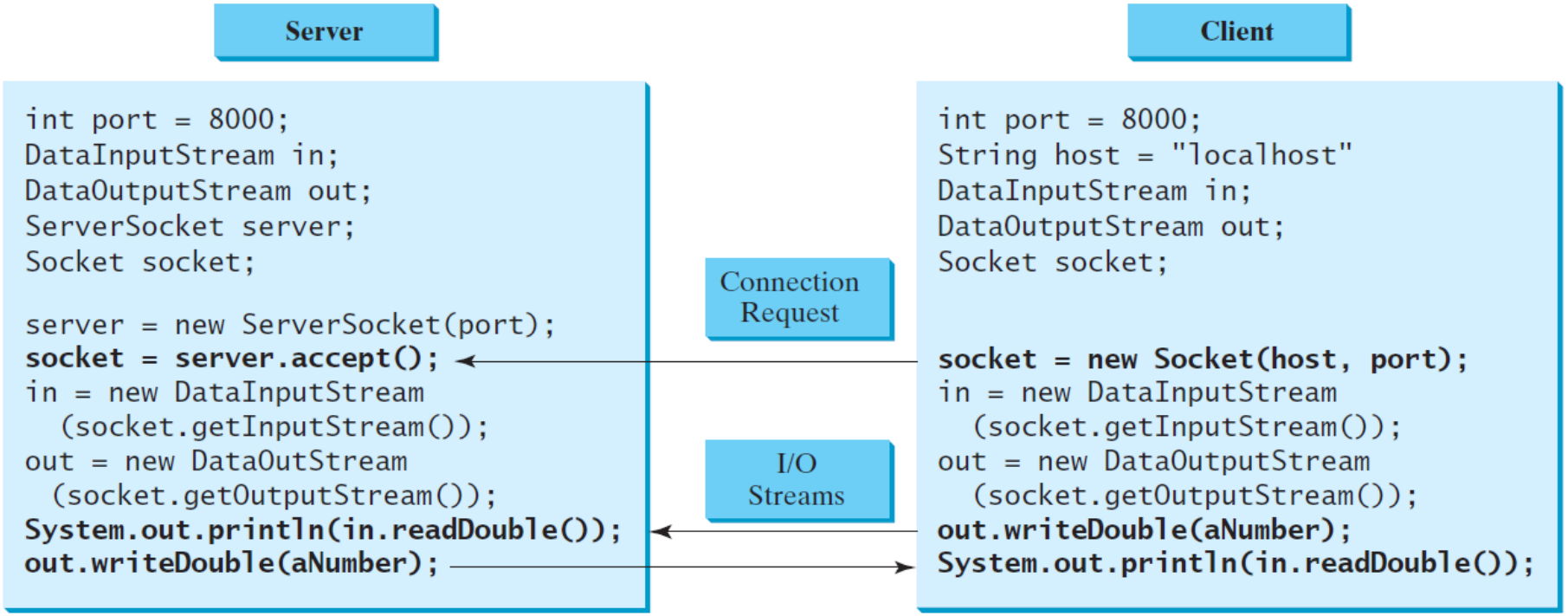
The client issues this statement to request a connection to a server.

Binary I/O Stream



`InputStream, OutputStream` (abstract class), and their subclasses are for performing binary I/O

Data Transmission through Sockets



InputStream input = socket.getInputStream();

OutputStream output = socket.getOutputStream();

Stream Socket vs. Datagram Socket

Stream socket

- A dedicated point-to-point channel between a client and server.
- Use TCP (Transmission Control Protocol) for data transmission.
- Lossless and reliable.
- Sent and received in the same order.

Datagram socket

- No dedicated point-to-point channel between a client and server.
- Use UDP (User Datagram Protocol) for data transmission.
- May lose data and not 100% reliable.
- Data may not received in the same order as sent.

UDP Socket Programming

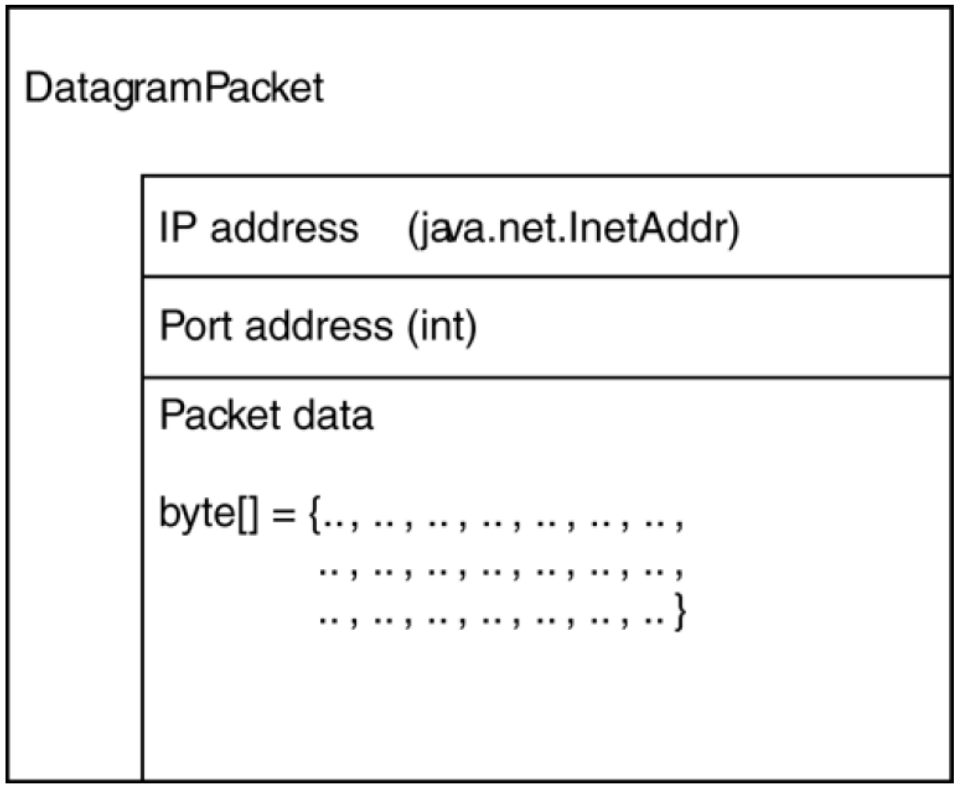
The two key classes from the `java.net` package used in creation of server and client programs are:

- `DatagramPacket`
- `DatagramSocket`

DatagramPacket Class

Two reasons to create a new `DatagramPacket`:

- To send data to a remote machine using UDP
- To receive data sent by a remote machine using UDP



`DatagramPacket` representation of a UDP packet:

DatagramPacket Class

- Constructor of `DatagramPacket`:
  - To receive incoming UDP packets:

```
DatagramPacket(byte[] buffer, int length);
```

  - To send to a remote machine:

```
DatagramPacket(byte[] buf, int length,
InetAddress address, int port);
```

- Key methods of `DatagramPacket`

`byte[] getData()`: Returns the data buffer.

`int getLength()`: Returns the length of the data to be sent or the length of the data received.

`void setData(byte[] buf)`: Sets the data buffer for this packet.

`void setLength(int length)`: Sets the length for this packet.

DatagramPacket

The DatagramPacket class represents a datagram packet. Datagram packets are used to implement a connectionless packet delivery service. Each message is routed from one machine to another based solely on information contained within the packet.

java.net.DatagramPacket	
length: int	A JavaBeans property to specify the length of buffer.
address: InetAddress	A JavaBeans property to specify the address of the machine where the package is sent or received.
port: int	A JavaBeans property to specify the port of the machine where the package is sent or received.
+DatagramPacket(buf: byte[], length: int, host: InetAddress, port: int)	Constructs a datagram packet in a byte array buf of the specified length with the host and the port for which the packet is sent. This constructor is often used to construct a packet for delivery from a client.
+DatagramPacket(buf: byte[], length: int)	Constructs a datagram packet in a byte array buf of the specified length.
+getData(): byte[]	Returns the data from the package.
+setData(buf: byte[]): void	Sets the data in the package.

DatagramSocket

DatagramSocket

The DatagramSocket class represents a socket for sending and receiving datagram packets. A datagram socket is the sending or receiving point for a packet delivery service. Each packet sent or received on a datagram socket is individually addressed and routed. Multiple packets sent from one machine to another may be routed differently, and may arrive in any order.

Create a server DatagramSocket

To create a server DatagramSocket, use the constructor DatagramSocket(int port), which binds the socket with the specified port on the local host machine.

Create a client DatagramSocket

To create a client DatagramSocket, use the constructor DatagramSocket(), which binds the socket with any available port on the local host machine.

DatagramSocket Class

- Constructor:

`DatagramSocket()`throws java.net.SocketException: Client DatagramSocket

`DatagramSocket(int port)`throws java.net.SocketException: Server DatagramSocket

- Key methods of `DatagramSocket`

`void send(DatagramPacket p):` Sends a datagram packet from this socket.

`void receive(DatagramPacket p):` Receives a datagram packet from this

Sending and Receiving a DatagramSocket

Sending

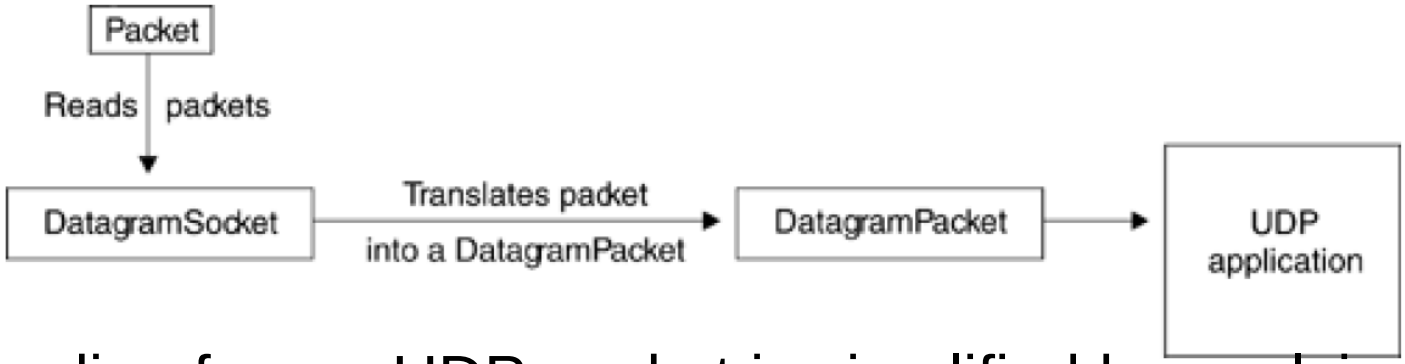
To send data, you need to create a packet, fill in the contents, specify the Internet address and port number for the receiver, and invoke the send(packet) method on a DatagramSocket.

Receiving

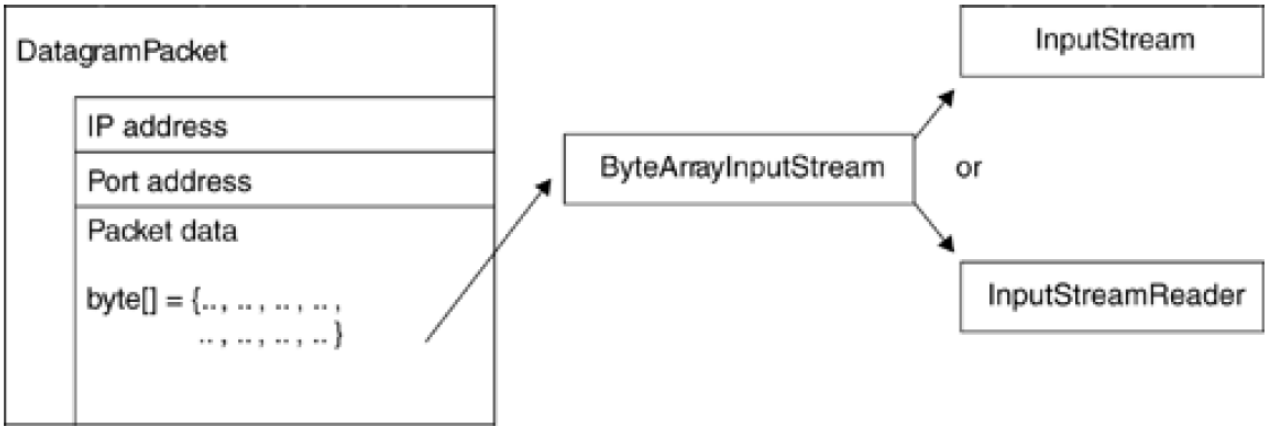
To receive data, create an empty packet and invoke the receive(packet) method on a DatagramSocket.

Listening for UDP Packets

UDP packets are received by a DatagramSocket and translated into a DatagramPacket object.

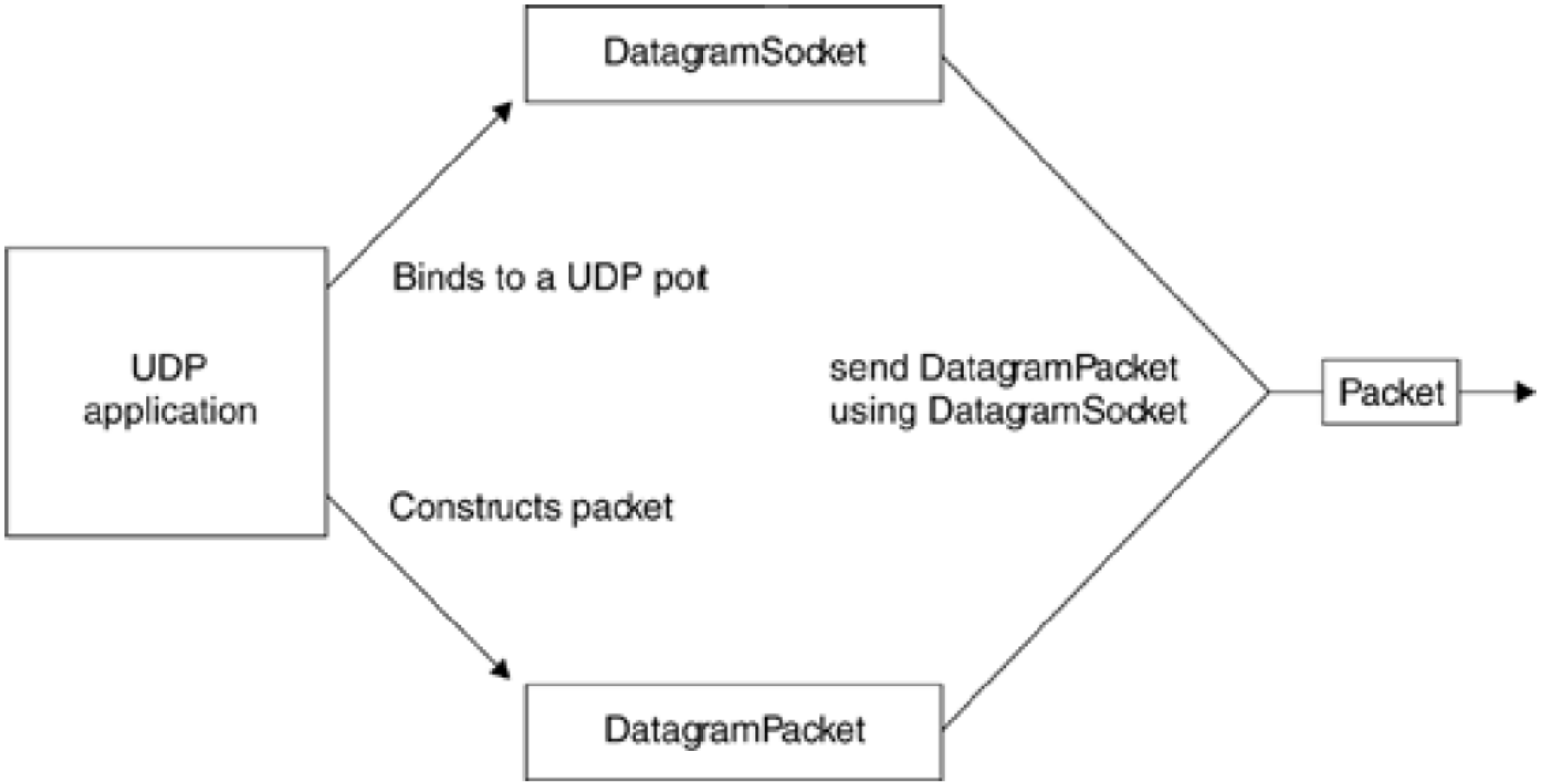


Reading from a UDP packet is simplified by applying input streams



Sending UDP Packets

Packets are send using DatagramSocket



Datagram Programming

Datagram programming is different from stream socket programming in the sense that there is no concept of a ServerSocket for datagrams. Both client and server use DatagramSocket to send and receive packets.

Designate one a server