TCP

Tr. Control Protocol

i) Connection Oriented

ii) Reliable (ordering)

iii) Error Control Maintaining ↓ checksum

iv) Slow transmission

v) More overhead (20–60 bytes)

vi) Flow Control, Congestion Control is needed

vii) Re-transmission possible

UDP

i) Connectionless

ii) Less Reliable (No ordering)

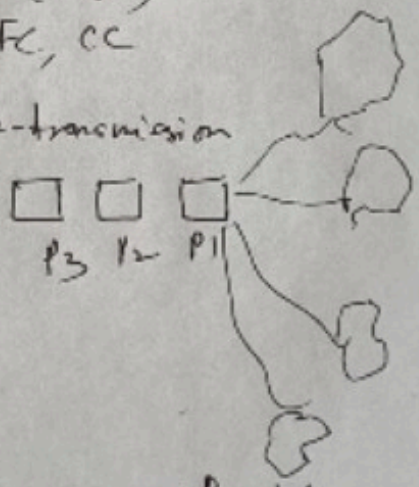
iii) Error Control optional ↓ checksum (16 bit)

iv) Fast transmission

v) Less overhead (8 byte)

vi) No FC, CC

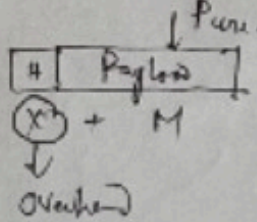
vii) No re-transmission



TCP	UDP
HTTP	DNS
FTP	BOOTP, DHCP
	RIP

Reliability

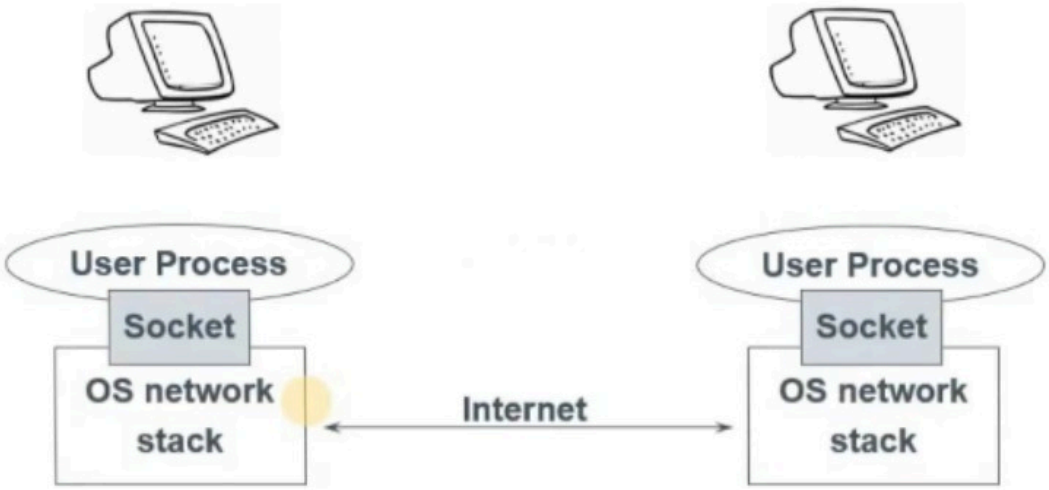
Fast transmission



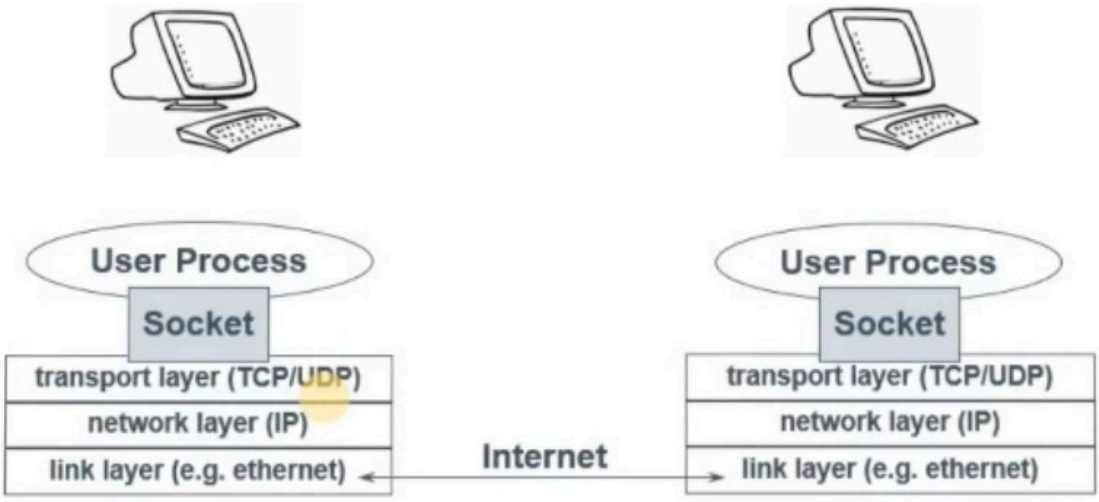
TCP / UDP Sockets

Socket and Process Communication

Socket and Process Communication

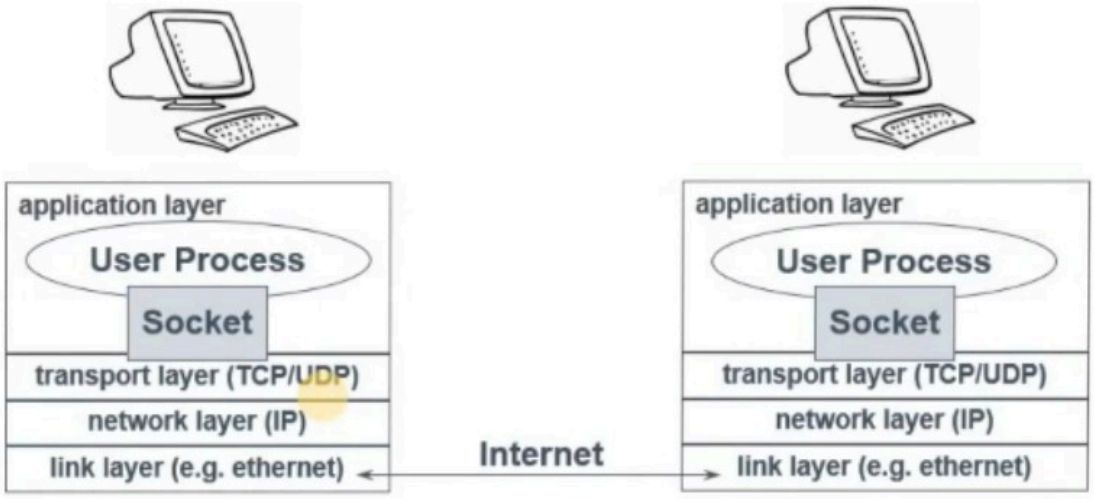


Socket and Process Communication



The interface that the OS provides to its networking subsystem

Socket and Process Communication



The interface that the OS provides to its networking subsystem

What is Socket?

- A socket is an endpoint in communication between two computers across a computer network.
- Its a virtual thing, and it does not mean any hardware.
- A socket is uniquely identified by an IP address and a Port.
- These (IP, Port) couple uniquely refers to an application.

Two Types of Application Processes Communication

- Datagram Socket (UDP)
  - Collection of messages
  - Best effort
  - Connectionless
- Stream Socket (TCP)
  - Stream of bytes
  - Reliable
  - Connection-oriented

Transmission Control Protocol (TCP): Stream Socket

TCP	Telephone Call
• Reliable – guarantee delivery	• Guaranteed delivery
• Byte stream – in-order delivery	• In-order delivery
• Connection-oriented – single socket per connection	• Connection-oriented
• Setup connection followed by data transfer	• Setup connection followed by conversation

Example TCP applications
Web, Email, Telnet

User Datagram Protocol (UDP): Datagram Socket

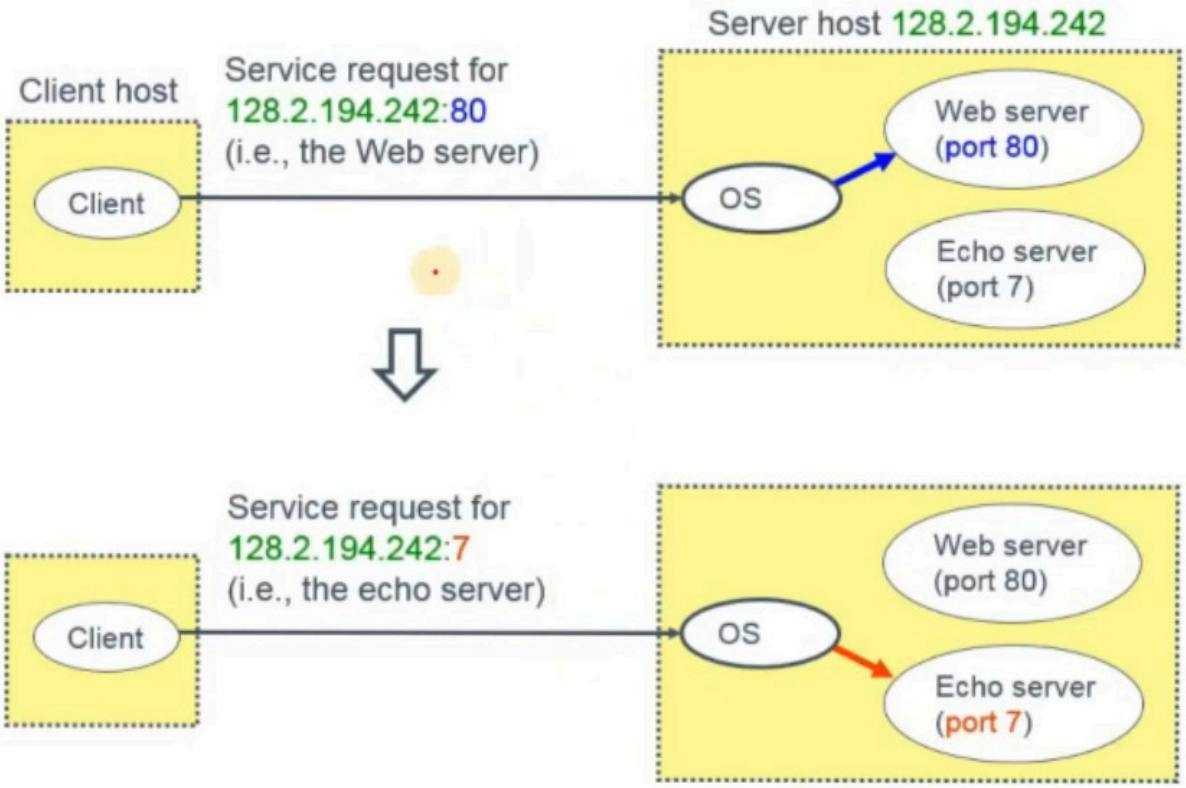
UDP	Postal Mail
• Single socket to receive messages	• Single mailbox to receive letters
• No guarantee of delivery	• Unreliable
• Not necessarily in-order delivery	• Not necessarily in-order delivery
• Datagram – independent packets	• Letters sent independently
• Must address each packet	• Must address each mail

Example UDP applications
Multimedia, voice over IP (Skype)

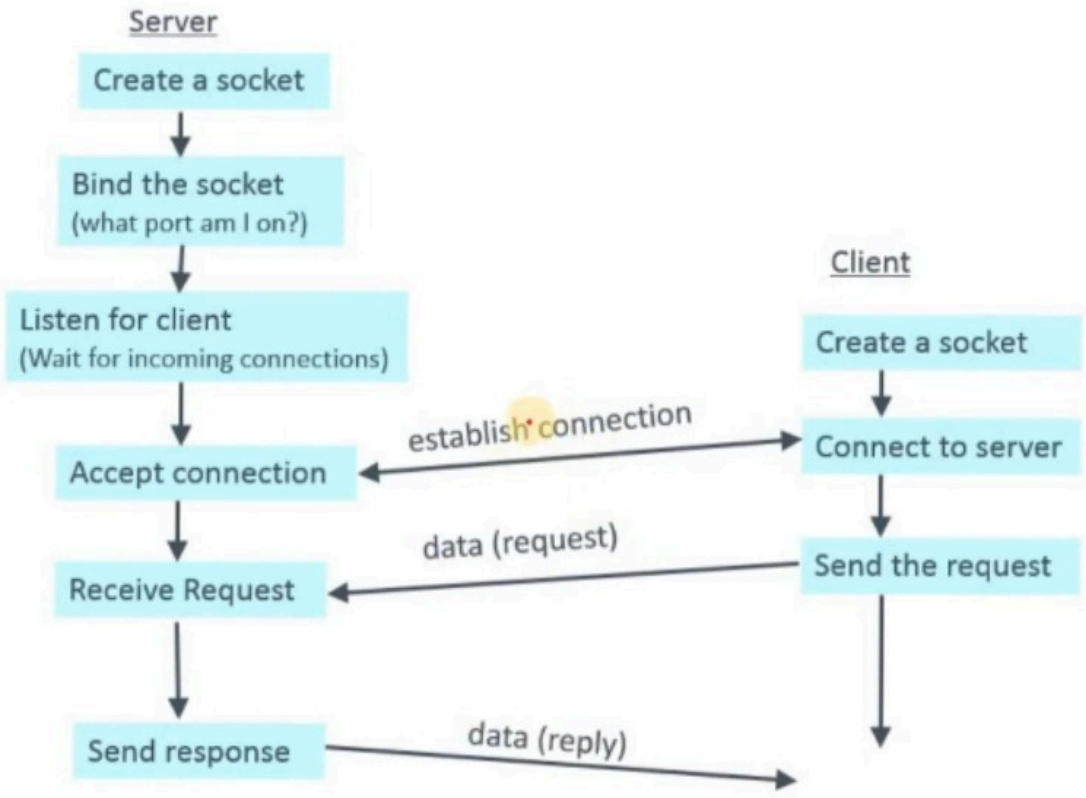
Knowing What Port Number To Use

- Popular applications have well-known ports
  - E.g., port 80 for Web and port 25 for e-mail
  - See http://www.iana.org/assignments/port-numbers
- Well-known vs. ephemeral ports
  - Server has a well-known port (e.g., port 80)
    - Between 0 and 1023 (requires root to use)
  - Client picks an unused ephemeral (i.e., temporary) port
    - Between 1024 and 65535
- Uniquely identifying traffic between the hosts
  - Two IP addresses and two port numbers
  - Underlying transport protocol (e.g., TCP or UDP)

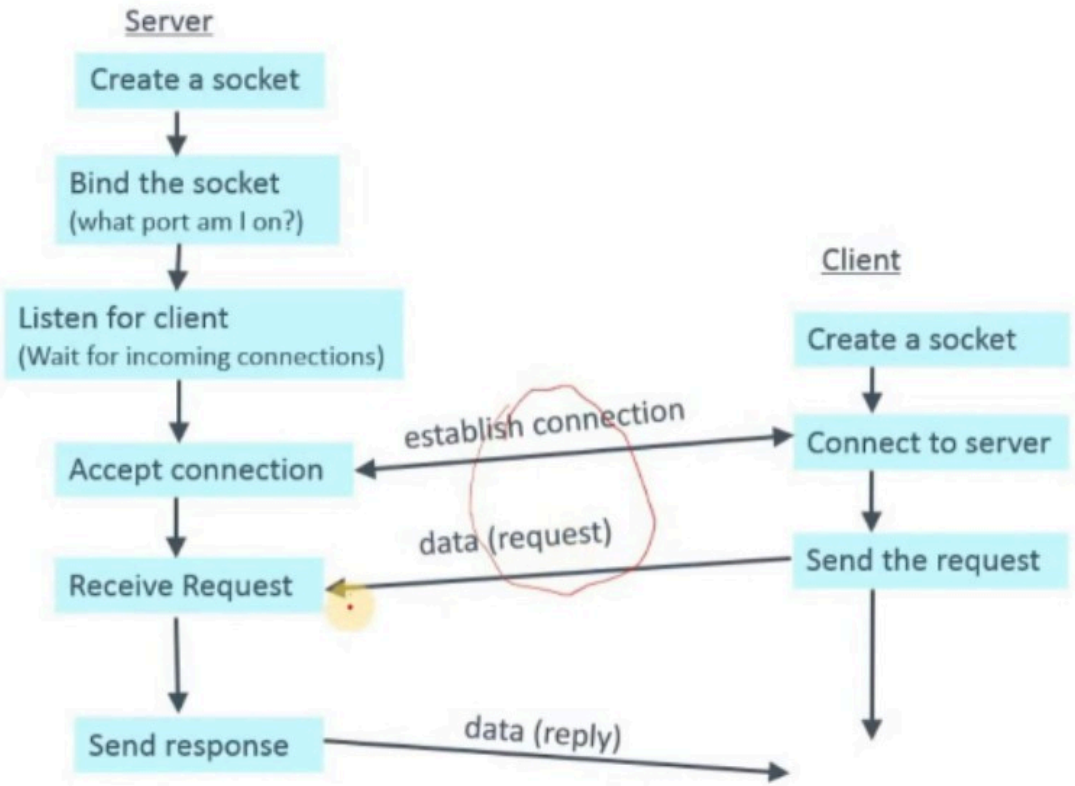
Using Ports to Identify Services



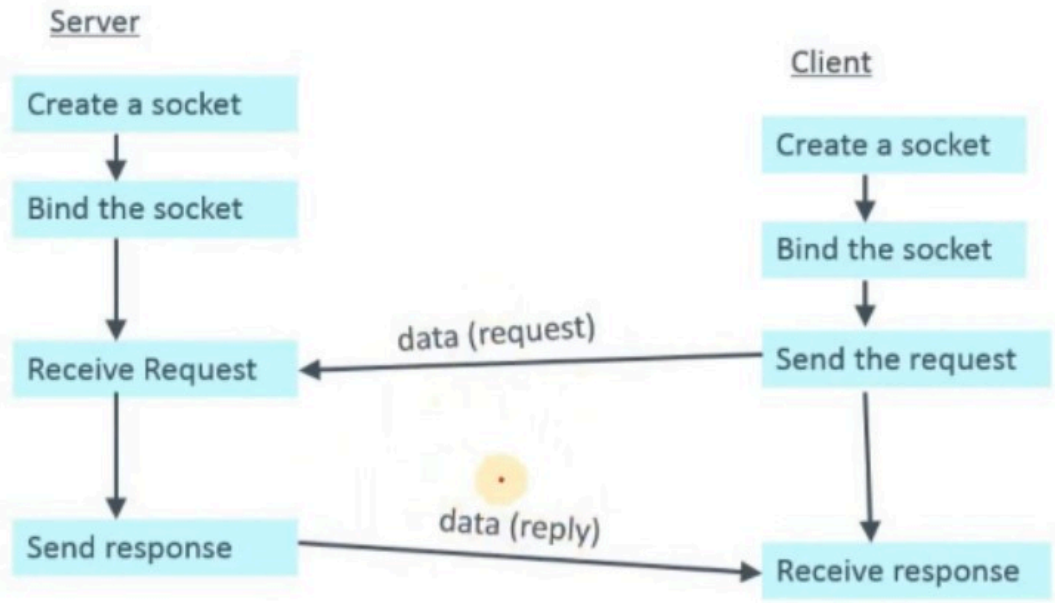
Client-Server Communication
Stream Sockets (TCP): Connection-oriented



Client-Server Communication
Stream Sockets (TCP): Connection-oriented



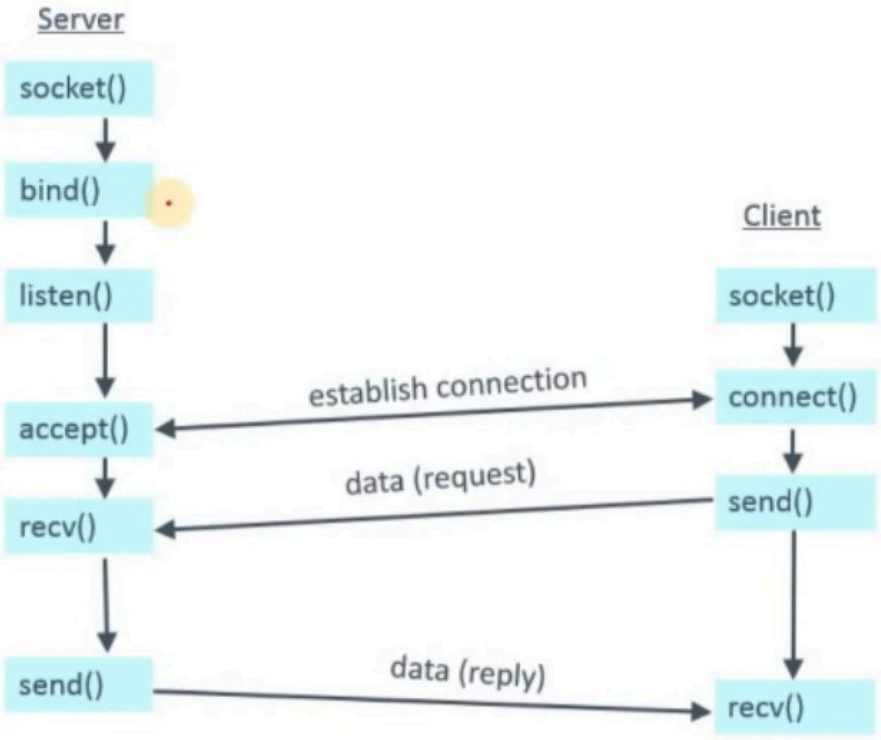
Client-Server Communication
Datagram Sockets (UDP): Connectionless



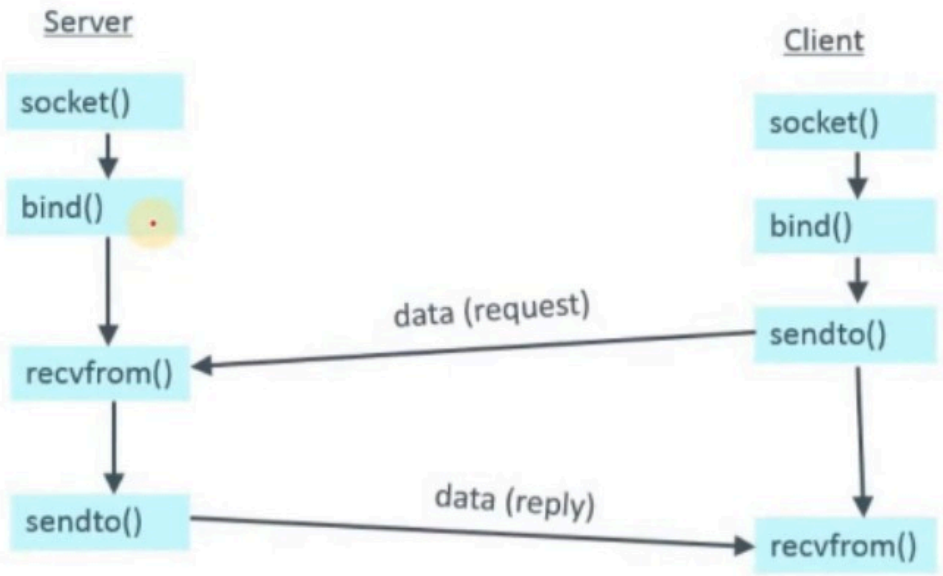
UNIX Socket API

- Socket interface
  - Originally provided in Berkeley UNIX
  - Later adopted by all popular operating systems
  - Simplifies porting applications to different OSes
- In UNIX, everything is like a file
  - All input is like reading a file
  - All output is like writing a file
  - File is represented by an integer file descriptor
- API implemented as system calls
  - E.g., connect, send, recv, close, ...

Connection-oriented Example (Stream Sockets -TCP)



Connectionless Example (Datagram Sockets - UDP)



```
mport java.net.DatagramPacket;
mport java.net.DatagramSocket;
mport java.net.InetAddress;

ublic class UDPServer
{
    public static void main(String a[]) throws Exception
    {
        DatagramSocket ds = new DatagramSocket(9999);

        byte[] b1 = new byte[1024];

        DatagramPacket dp = new DatagramPacket(b1, b1.length);
        ds.receive(dp);
        String str = new String(dp.getData(),0,dp.getLength());
        System.out.println("str " + str);
        int num = Integer.parseInt(str.trim());
        System.out.println("num " + num);
        int result = num*num;
        byte[] b2 = String.valueOf(result).getBytes();
        InetAddress ia = InetAddress.getLocalHost();
        DatagramPacket dp1 = new DatagramPacket(b2, b2.length,ia,dp.getPort());
        ds.send(dp1);


    }
}
```

Writable | Smart Insert | 25 : 22

```
import java.net.DatagramSocket;
import java.net.InetAddress;

public class UDPClient {

    public static void main(String[] args) throws Exception
    {

            DatagramSocket ds = new DatagramSocket();

            int i=8;
            byte[] b = (i+"").getBytes();

            InetAddress ia = InetAddress.getLocalHost();
            DatagramPacket dp = new DatagramPacket(b,b.length,ia,9999);
            ds.send(dp);


            byte[] b1 = new byte[1024];
            DatagramPacket dp1 = new DatagramPacket(b, b.length);
            ds.receive(dp1);

            String str = new String(dp1.getData());
            System.out.println("result is " + str);


    }
```