第2章 应用层

TCP、UDP套接字编程

任课老师：周军海

Email:rj_zjh@hnu.edu.cn

2.7 TCP套接字编程

- 网络应用程序的核心：客户机程序和服务器程序。

  运行时，分别创建一个客户机进程和一个服务器进程，相互之间通过套接字读写数据进行通信。

- 网络应用程序类型：

✓ *通用应用程序*：通过RFC文档所定义的标准协议来实现

程序必须满足该RFC所规定的规则；使用与协议相关的端口号。如Web应用

✓ *专用的应用程序*：

程序不必符合RFC规则；开发者根据实际应用设计；不能使用RFC中定义的周知端口号。

说明

研发初期，先选择运输层协议：

- ***TCP*：**

  面向连接的，为两个端系统之间的数据流动提供可靠的字节流通道。

- ***UDP*：**

  无连接的，从一个端系统向另一个端系统发送独立的数据分组，不对交付提供任何保证。

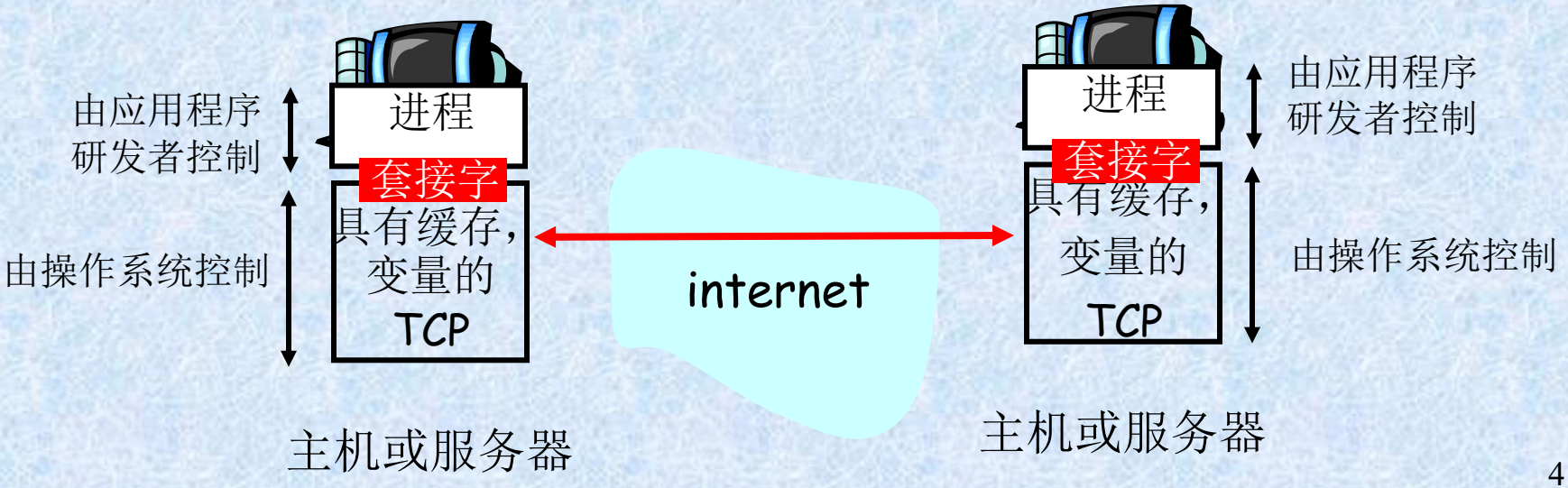
2.7.1 TCP套接字编程

运行在不同机器上的进程彼此通过套接字传递报文来进行通信。

- 进程/套接字：房子/门户，即套接字是应用进程和TCP之间的门户。

程序开发者可以控制应用层端所有东西；不能控制运输层端。

- TCP服务：从一个进程到另一个进程的可靠字节传输



客户机和服务器程序之间的交互

先建立TCP连接，再进行数据传输。

- ✓ 客户机程序是连接的发起方；
- ✓ 服务器必须先准备好，对客户机程序发起的连接做出响应：
  - 服务器程序事先已经在系统中运行；
  - 服务器程序的一个套接字（欢迎套接字）已经打开，准备接收客户机程序发起的连接（敲门）。

具体过程：

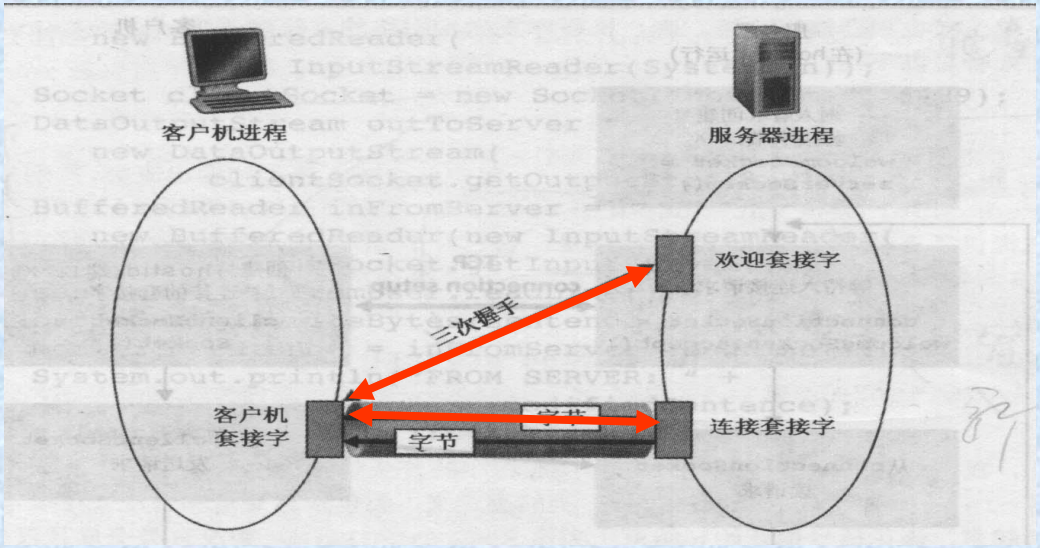
建立TCP连接

- ✓ *客户机进程向服务器发起一个TCP连接：*

  创建一个本地套接字，指定相应服务器进程的地址（IP地址和端口号）。

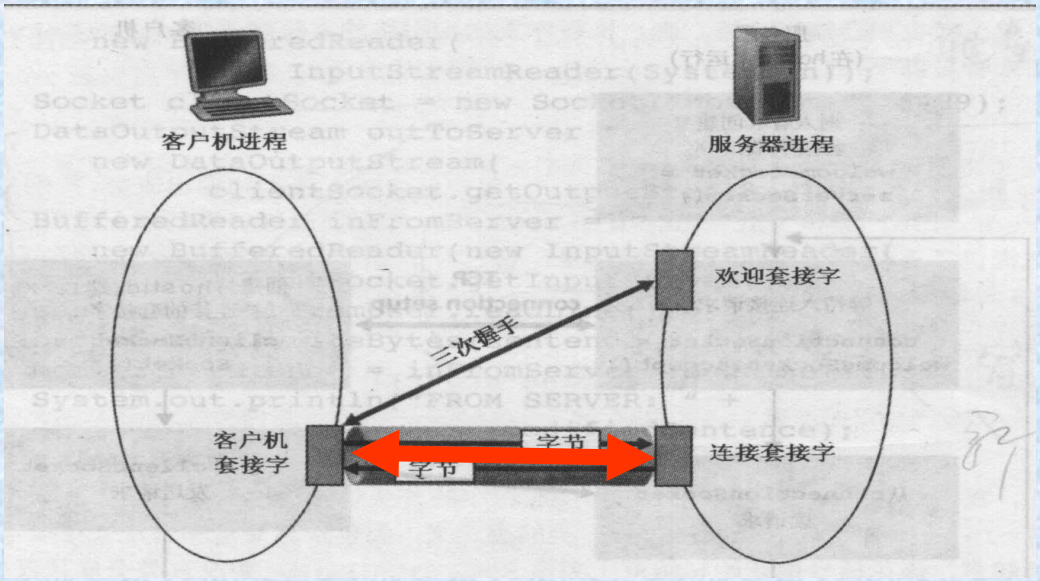
- ✓ *建立一个TCP连接：*

  当服务器听到客户机的连接请求（敲门）时，创建一个新套接字，经过“三次握手”，客户机套接字和服务器套接字之间建立一个TCP连接（直接的虚拟管道）。



&传送数据

- TCP连接为客户机和服务器提供了一个直接的传输管道。
- 可靠的, 顺序的,字节流的传输



术语

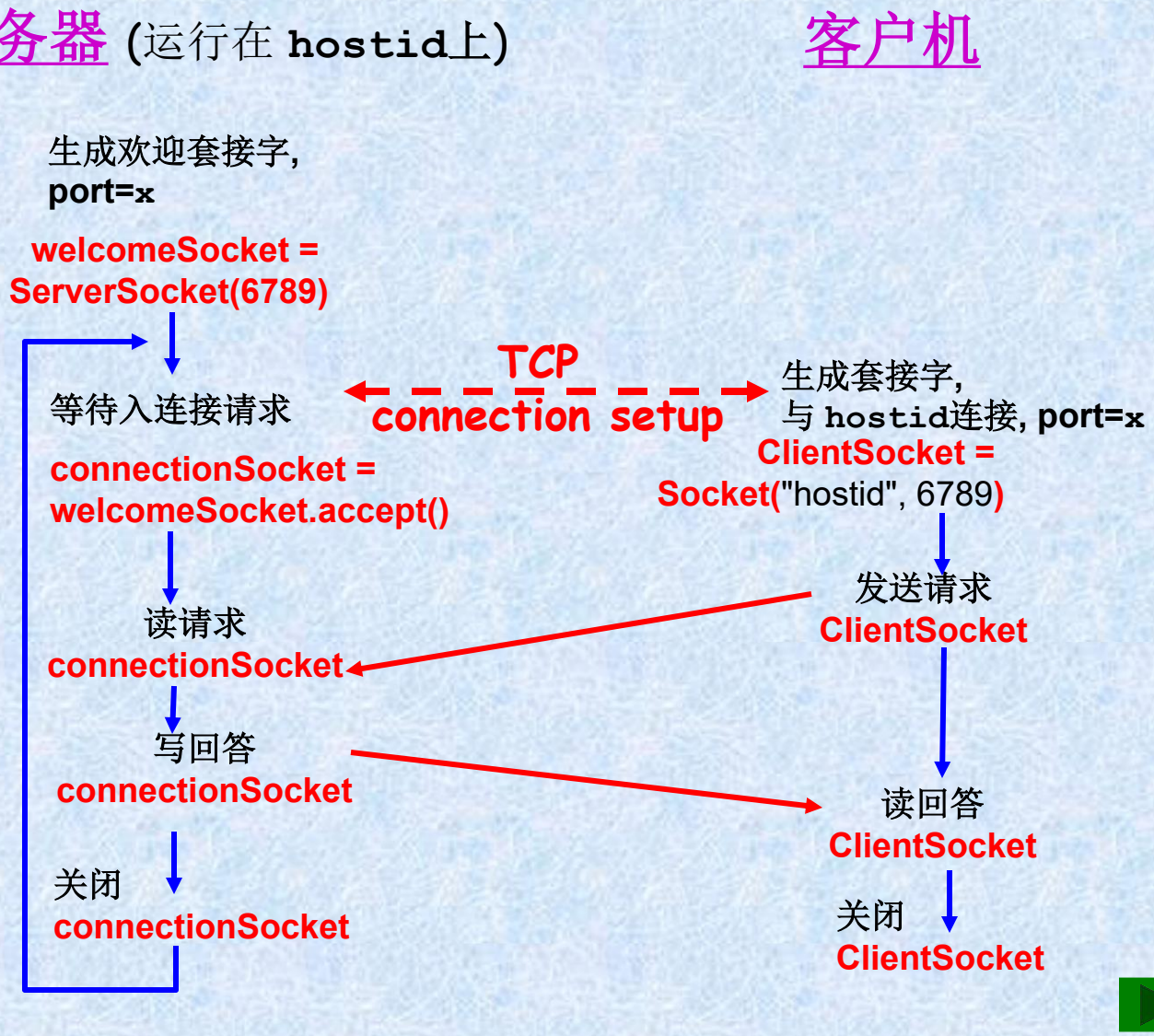
- ✓ **流：**流入或流出某进程的一串字符序列。
- ✓ **输入流：**来自某个输入源（如键盘）、或某个套接字（因特网的数据流入套接字）。
- ✓ **输出流：**到某个输出源（如显示器）、或某个套接字（数据通过套接字流向因特网）。

2.7.2 Java应用程序示例

客户机和服务器经TCP连接进行通信。

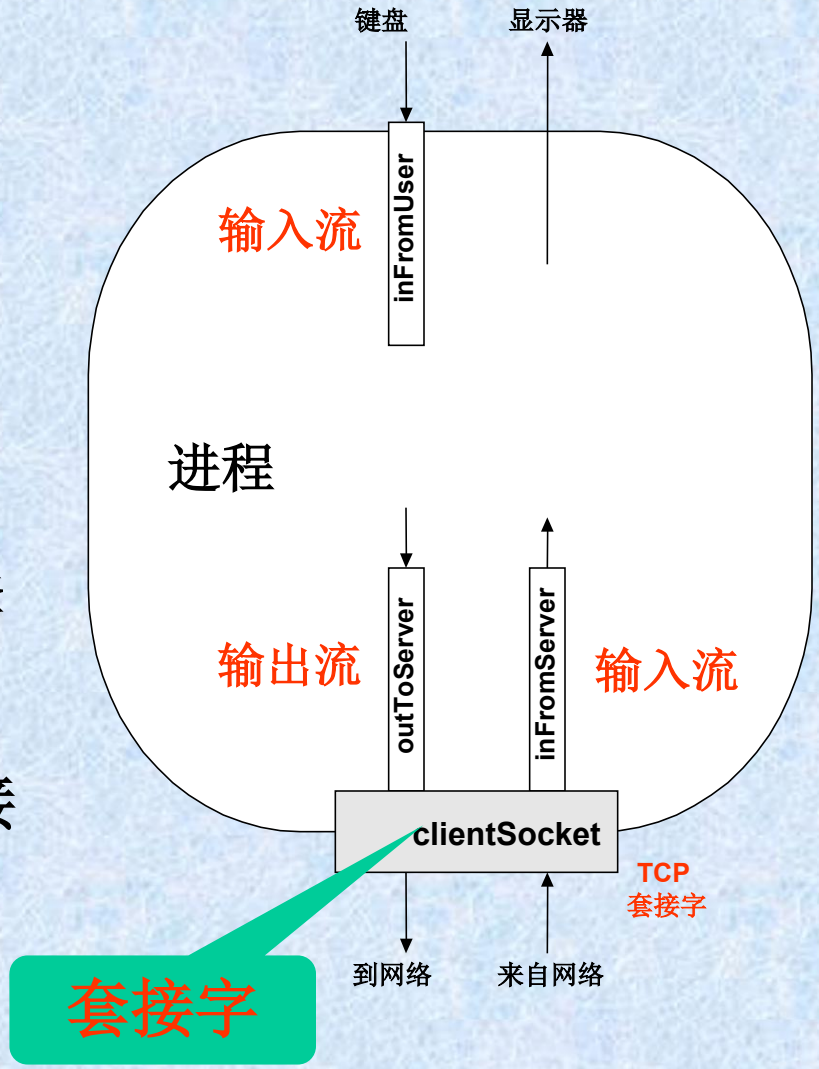
- ✓ 客户机从键盘读一行字符，通过套接字向服务器发送
- ✓ 服务器从套接字读取数据；
- ✓ 将该行字符转换成大写；
- ✓ 将修改的行通过其连接套接字再回发给客户机。
- ✓ 客户机从其套接字中读取修改的行，并将该行在显示器上显示。

客户机/服务器程序交互


服务器 (运行在 hostid上)
客户机
生成欢迎套接字,
port=x
welcomeSocket =
ServerSocket(6789)
TCP
connection setup
等待入连接请求
生成套接字,
与 hostid连接, port=x
connectionSocket =
welcomeSocket.accept()
ClientSocket =
Socket("hostid", 6789)
读请求
connectionSocket
发送请求
ClientSocket
写回答
connectionSocket
读回答
ClientSocket
关闭
connectionSocket
关闭
ClientSocket

客户机端

- 创建了三个流和一个套接字，如图所示。
  - ✓ *套接字：* clientSocket ；
  - ✓ *InFromUser输入流：* 连接到键盘；
  - ✓ *InFromServers输入流：* 与套接字连接。从网络来的字符
  - ✓ *outToServers输出流：* 与套接字连接，客户机发送到网络的字符。



客户机：(*TCPClient.java*)

产生输入流、输出流、套接字→输入→发送→接收→显示

```
import java.io.*;
import java.net.*;
class TCP Client {

    public static void main(String argv[ ]) throws Exception
    {
        String sentence;
        String modifiedSentence;

        BufferedReader inFromUser =
          new BufferedReader(new InputStreamReader(System.in));

        Socket ClientSocket = new Socket("hostname", 6789);

        DataOutputStream outToServer =
          new DataOutputStream(ClientSocket.getOutputStream());
```

产生输入流

产生客户机套接字，与服务器连接

生成输出流与套接字联系

产生与套接字联系的输入流

```
BufferedReader inFromServer =
  new BufferedReader(new
  InputStreamReader(ClientSocket.getInputStream()));

sentence = inFromUser.readLine();
```

键盘输入

向服务器发送行

```
outToServer.writeBytes(sentence + '\n');
```

从服务器读行

```
modifiedSentence = inFromServer.readLine();

System.out.println("FROM Server: " + modifiedSentence);
```

显示内容

```
ClientSocket.close();

    }
}
```

服务器: (*TCPServer.java*)

产生输入流、输出流、套接字→接收→发送

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

class TCP Server {

  public static void main(String argv[]) throws Exception
   {
     String ClientSentence;
     String capitalizedSentence;

     ServerSocket welcomeSocket = new ServerSocket(6789);

     while(true) {

        Socket connectionSocket = welcomeSocket.accept();

        BufferedReader inFromClient =
          new BufferedReader(new
          InputStreamReader(connectionSocket.getInputStream()));
```

在端口6789 生成欢迎套接字，监听

创建一个连接套接字

生成输入流，与套接字联系

生成输出流，
与套接字联系

```
DataOutputStream  outToClient =
  new DataOutputStream(connectionSocket.getOutputStream());
```

从套接字读入客
户机来的数据

```
ClientSentence = inFromClient.readLine();

capitalizedSentence = ClientSentence.toUpperCase() + '\n';
```

向套接字输出
数据到客户机

```
outToClient.writeBytes(capitalizedSentence);
            }
        }
    }
```

循环结束，返回并等待另
一个客户机连接

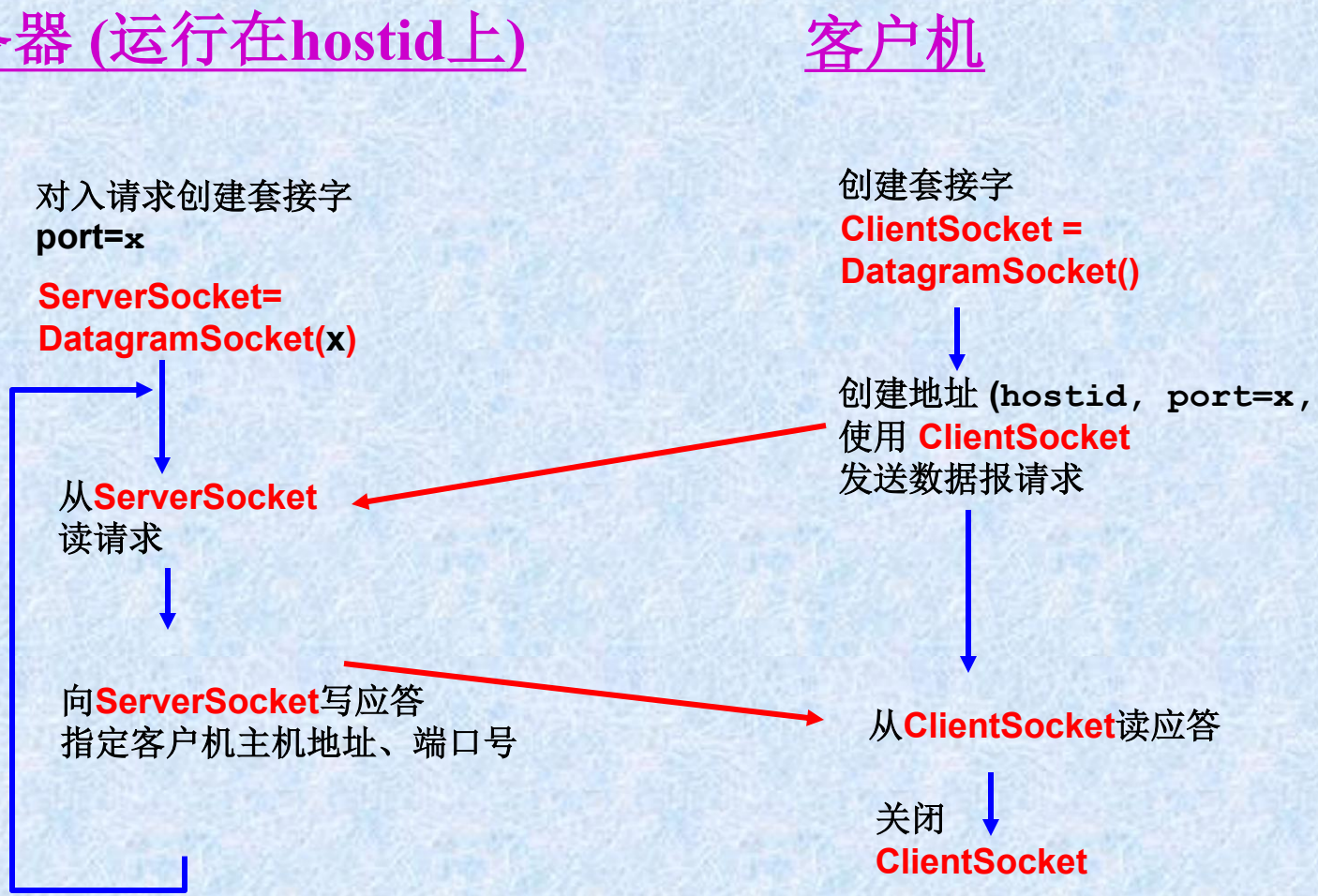
2.8 UDP套接字编程

- ✓ UDP是一种无连接的服务，即在两个进程之间没有创建管道时所需的初始握手阶段。
- ✓ 进程之间的数据传递以分组为单位进行。
- ✓ 分组中含目的进程地址（主机IP地址和端口号）。
- ✓ 提供不可靠的传输服务。

编程说明：

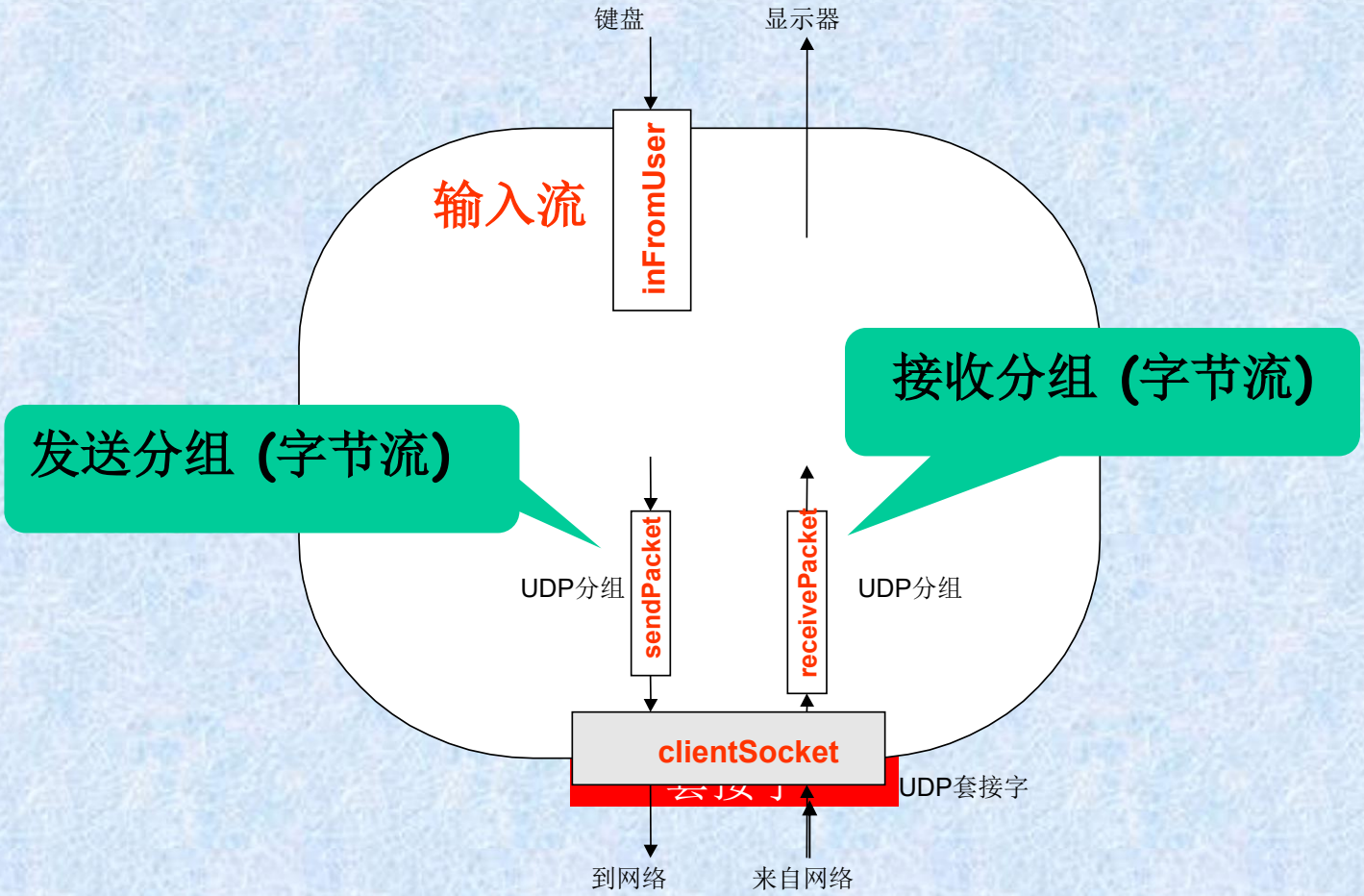
- ✓ 通信进程之间没有初始握手，不需要欢迎套接字；
- ✓ 没有流与套接字相联系；
- ✓ 发送主机将信息字节封装生成分组，再发送；
- ✓ 接收进程解封收到的分组，获得信息字节。

客户机/服务器程序交互


服务器 (运行在hostid上)
客户机
对入请求创建套接字
port=x
ServerSocket=
DatagramSocket(x)
从ServerSocket
读请求
向ServerSocket写应答
指定客户机主机地址、端口号
创建套接字
ClientSocket =
DatagramSocket()
创建地址 (hostid, port=x,
使用 ClientSocket
发送数据报请求
从ClientSocket读应答
关闭
ClientSocket

客户机：

- ✓ 一个套接字ClientSocket：发送和接收分组。
- ✓ 一个输入流InfromUser：与标准输入联系（键盘）。



客户机 (*UDPClient.java*)

产生输入流、套接字→输入→封装→发送

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

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

      BufferedReader inFromUser =
        new BufferedReader(new InputStreamReader(System.in));

      DatagramSocket ClientSocket = new DatagramSocket();

      InetAddress IPAddress = InetAddress.getByName("hostname");

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

      String sentence = inFromUser.readLine();

      sendData = sentence.getBytes();
```

生成输入流

生成客户机套接字

使用DNS将主机名转换为IP地址

输入

使用要发送的数据、长度、IP地址、端口生成数据报

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

向服务器发送数据报

```
ClientSocket.send(sendPacket);
```

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

从服务器读数据报

```
ClientSocket.receive(receivePacket);
```

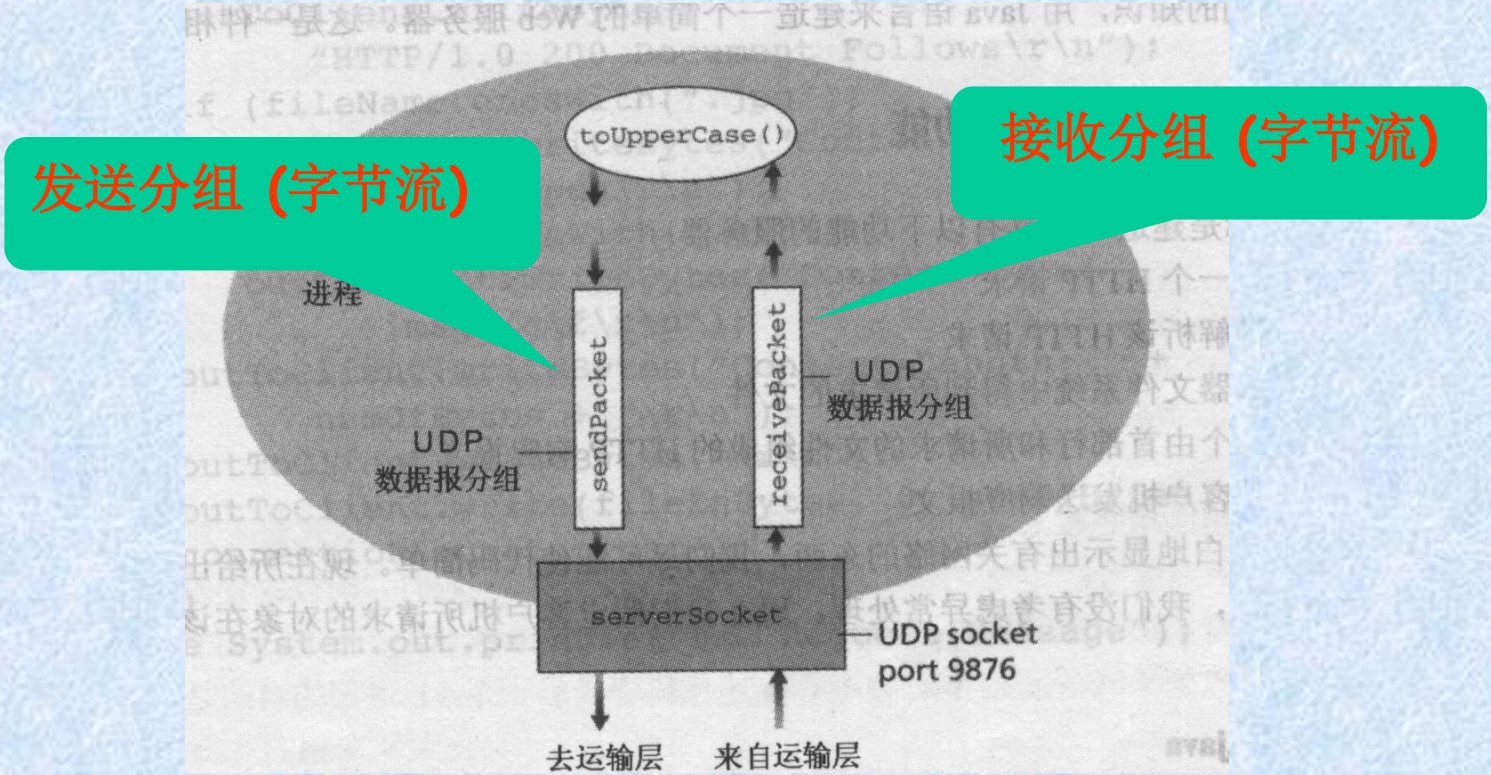
```
String modifiedSentence =
    new String(receivePacket.getData());
```

显示

```
System.out.println("FROM Server:" + modifiedSentence);
ClientSocket.close();
}
}
```

服务器：

- 一个套接字serverSocket：发送和接收分组。



服务器(*UDPServer.java*)

产生套接字→接收→解封→发送

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

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

在端口9876生成数据报套接字 →

```
    DatagramSocket ServerSocket = new DatagramSocket(9876);

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

    while(true)
     {
```

为接收的数据报生成空间 →

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

接收数据报 →

```
ServerSocket.receive(receivePacket);
```

```
String sentence = new String(receivePacket.getData());

InetAddress IPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

String capitalizedSentence = sentence.toUpperCase();

sendData = capitalizedSentence.getBytes();

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

ServerSocket.send(sendPacket);
        }
    }
}
```

获得发送方的 IP 地址，端口#

产生数据报发送给客户机

向套接字写出数据报

循环结束，返回等待另一个数据报

2.10 小结

- 应用程序体系结构
  - 客户机-服务器
  - P2P
  - 混合
- 应用程序服务要求:
  - 可靠，带宽，时延
- 因特网传输服务模型
  - 面向连接，可靠: TCP
  - 无连接，不可靠: UDP
- 特定协议:
  - HTTP
  - FTP
  - SMTP, POP3, IMAP
  - DNS
- P2P文件共享
- 套接字编程