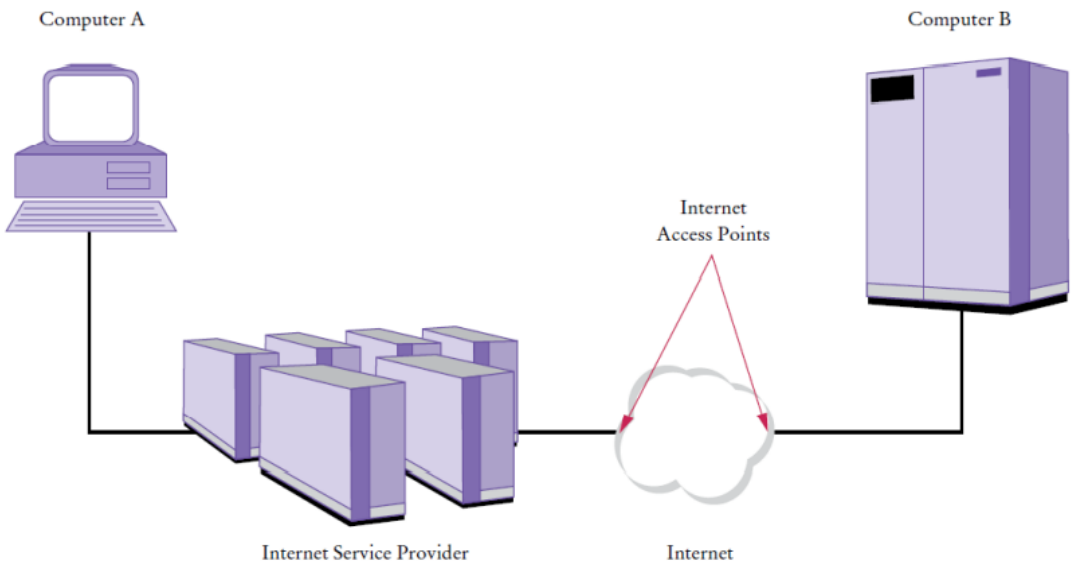
Lecture12-2 网络编程

1. 网络介绍



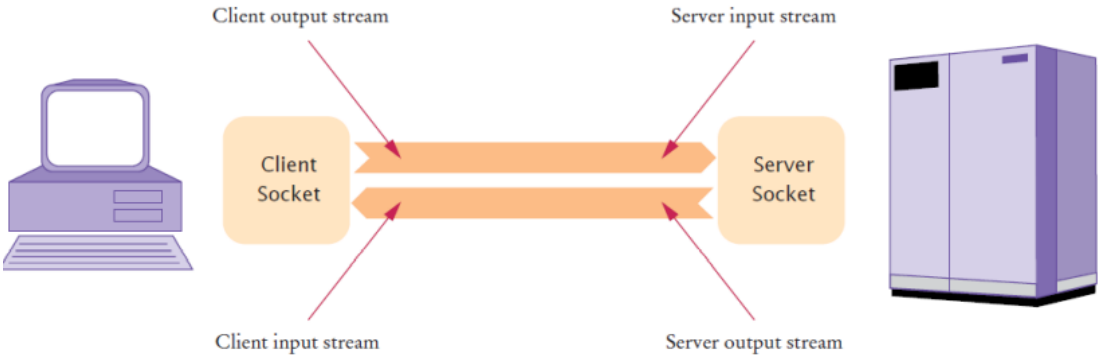
HTTP 请求

Table 1 HTTP Commands

Command	Meaning
GET	Return the requested item
HEAD	Request only the header information of an item
OPTIONS	Request communications options of an item
POST	Supply input to a server-side command and return the result
PUT	Store an item on the server
DELETE	Delete an item on the server
TRACE	Trace server communication

2. 客户端

构建客户端



在 TCP/IP 术语中，在两边都存在一个 Socket 用于通信，客户端通过如下代码建立一个 Socket

```
Socket s = new Socket(hostname, portnumber);
```

例如，如果想连接服务器 horstmann.com 你可以使用

```
final int HTTP_PORT = 80;
Socket s = new Socket("horstmann.com", HTTP_PORT)
```

如果 Socket 不能找到 host，它构造器会抛出 `UnknownHostException`

当你拥有了一个 socket 时，你获得了它的输入和输出流

```
InputStream instream = s.getInputStream;
OutputStream outstream = s.getOutputStream;
```

第一个客户端程序

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

public class WebGet {

    private final static String HOST = "www.baidu.com";
    private final static String RESOURCE = "/";
    private final static int PORT = 80;
    public static void main(String[] args) throws IOException {
        try(Socket s = new Socket(HOST, PORT)){

            InputStream inputStream = s.getInputStream();
```

```
            OutputStream outputStream = s.getOutputStream();

            // 将输入输出流转换为 scanners 和 writers
            Scanner in = new Scanner(inputStream);
            PrintWriter out = new PrintWriter(outputStream);

            // 发送请求
            String cmd = "GET " + RESOURCE + " HTTP/1.1\n" + "Host: " + HOST +
    "\n\n";
            out.println(cmd);
            out.flush();

            while(in.hasNextLine()){
                System.out.println(in.nextLine());
            }
        }// try-with-resources 语句会自动关闭 socket
    }
}
```

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: no-cache
Connection: keep-alive
Content-Length: 9508
Content-Type: text/html
Date: Tue, 30 Nov 2021 11:57:18 GMT
P3p: CP=" OTI DSP COR IVA OUR IND COM "
P3p: CP=" OTI DSP COR IVA OUR IND COM "
Pragma: no-cache
Server: BWS/1.1
Set-Cookie: BAIDUID=633A17297CE4560ED1E272924E33B908:FG=1; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: BIDUPSID=633A17297CE4560ED1E272924E33B908; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: PSTM=1638273438; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: BAIDUID=633A17297CE4560EFD9EB3B1E5593089:FG=1; max-age=31536000; expires=Wed, 30-Nov-22 11:57:18 GMT; domain=.baidu.com; path=/; version=1; comment=bd
Traceid: 163827343802318307941117543067139354934 5
Vary: Accept-Encoding
X-Frame-Options: sameorigin
X-Ua-Compatible: IE=Edge,chrome=1

<!DOCTYPE html><html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"><meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"><meta content="
```

3. 服务器

每当开发一个服务器应用程序时，需要指定一些客户端可以用来与服务器交互的**应用程序级协议**

我们以多线程中的 Bank 部分进一步介绍，服务器与客户端约定了下面的功能

Table 2 A Simple Bank Access Protocol

Client Request	Server Response	Description
BALANCE *n*	*n* and the balance	Get the balance of account *n*
DEPOSIT *n a*	*n* and the new balance	Deposit amount *a* into account *n*
WITHDRAW *n a*	*n* and the new balance	Withdraw amount *a* from account *n*
QUIT	None	Quit the connection

构建服务器

为了构建一个服务器 socket，需要提供端口号

```
ServerSocker server = new ServerSocket(8888);
```

`ServerSocket` 的 `accept` 方法等待一个客户端的连接，当一个客户端连接时，服务器维护一个与客户端连接的 socket

```
Socket s = server.accept();
BankService service = new BankService(s, bank);
```

`BankService` 类负责执行服务，这个类实现了 `Runnable` 接口，它的 `run()` 方法将在每一个客户端的连接时执行，`run()` 方法从 socket 获得的 `Scanner` 和 `PrintWriter` 和在客户端将的方法相同

```
public void doService() throws IOException{
    while(true){
        if(!in.hasNext){return;}
        String command = in.next();
        if(command.equals("QUIT"){return;}
        executeCommand(command);
    }
}
```

`executeCommand` 执行一个单一的命令

- 如果命令是 `DEPOSIT`，它会执行存款

  ```
  int account = in.nextInt();
  double amount = in.nextDouble();
  bank.deposit(account, amount);
  ```

- 如果命令是 `WITHDRAW` 它会执行存款

  ```
  int account = in.nextInt();
  double amount = in.nextDouble();
  bank.withdraw(account, amount);
  ```

在每条命令之后，账户号码和现在的存款额度将会被发送给客户端

```
out.println(account + " " + bank.getBalance(account));
```

`BankService` 类实现了 `Runnable` 接口，因此服务器的程序只需要通过简单的启动线程即可以开始

```
Thread t = new Thread(service);
t.start();
```

当客户端退出或断开连接并且 `run` 方法退出时，线程死亡

同时，`BankServer` 循环回去接受下一个连接

```
while(true){
    try(Socked s = server.accept()){
        BankService service = new BankService(s, bank);
        Thread t = new Thread(service);
        t.start();
    }
}
```

4. URL 连接

`URLConnection` 类使得从 web 服务器获取一个文件变得非常容易，因为 URL 是一个字符串

首先，以熟悉的格式从 URL 构造一个 `URL` 对象，从 http 或 ftp 前缀开始

然后你使用 `URL` 对象的 `openConnection()` 方法来获取 `URLConnection` 对象本身

```
URL u = new URL("http://horstmann.com/index.html");
URLConnection connection = u.openConnection();
```

然后，调用 `getInputStream()` 方法来获得 `InputStream`

```
InputStream instream = connection.getInputStream();
```

可以按照通常的方式将 `InputStream` 转换为 `Scanner`，并从扫描器读取输入

```
import java.io.InputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.HttpURLConnection;
```

```java
import java.net.URL;
import java.net.URLConnection;
import java.util.Scanner;

/**
   This program demonstrates how to use a URL connection
   to communicate with a web server. Supply the URL on the
   command line, for example
      java URLGet http://horstmann.com/index.html
*/
public class URLGet {
   public static void main (String[] args) throws IOException {
      // Get command line arguments
      String sURL = "http://www.sustech.edu.cn/";
      if (args.length >= 1) sURL = args[0];

      System.out.println( "URLGet " + sURL );

      // Open connection
      URL u = new URL( sURL);
      URLConnection connection = u.openConnection();

      // Check if response code is HTTP_OK (200)
      HttpURLConnection httpConnection = (HttpURLConnection) connection;
      int code = httpConnection.getResponseCode();
      String message = httpConnection.getResponseMessage();
      System.out.println( code + " " + message );
      if (code != HttpURLConnection.HTTP_OK) return;

      // Read server response
      InputStream instream = connection.getInputStream();
      Scanner in = new Scanner( instream);

      while (in.hasNextLine()) {
         String input = in.nextLine();
         System.out.println( input);
      }
   }
}
```