



北京邮电大学

计算机网络

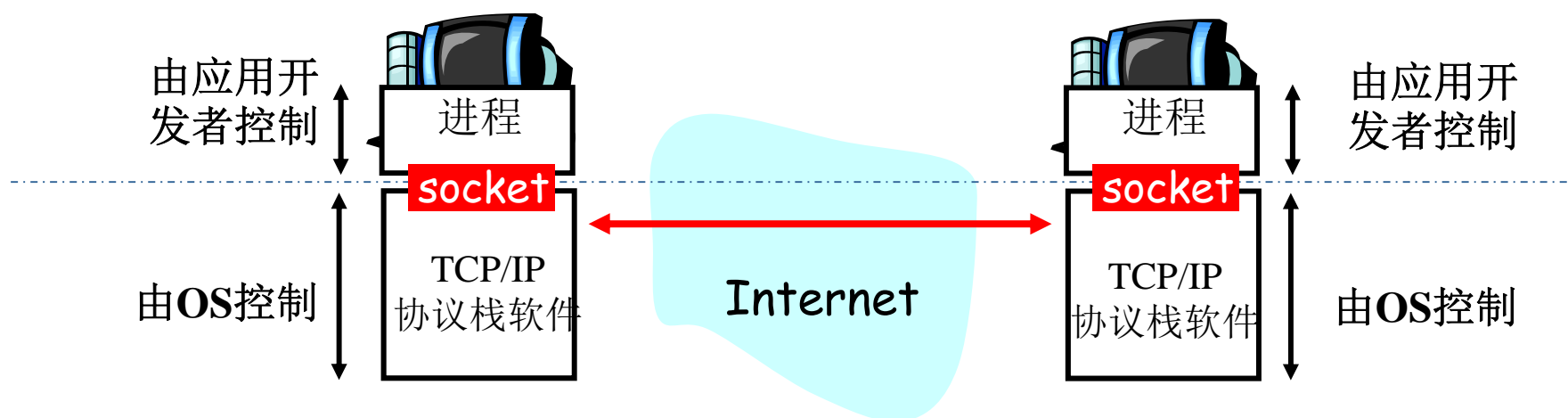
Socket程序设计简介

网络空间安全学院

2016年11月

什么是Socket（1）？

- ◆ 套接字
- ◆ 提供了用户访问网络的通信接口（API）
- ◆ 起源于80年代初期，UNIX BSD的一部分
- ◆ 是所有因特网应用的基础



什么是Socket (2) ?

- ◆ 对于OS内核而言，Socket是网络通信的一个端点（Endpoint）
 - Socket地址：IP地址+端口号



什么是Socket (3) ?

- ◆ 对于网络应用程序而言，Socket是进程间（远程）通信的一种机制
 - ◆ Socket基于客户-服务器（C/S）通信模式
 - I/O的扩展
 - 客户/服务器通过Socket描述符（类似文件描述符）对网络进行读/写（发送/接收）操作
 - 增强了一些特殊的系统调用
 - 建立Socket
- ```
int s = socket (PF_INET, Socket type, Protocol);
```

# Unix/Linux的文件模型

---

◆ 在UNIX中，所有I/O都被看作文件

➤ 由文件描述符标识

➤ 文件操作

- Open
- Close
- Read
- Write
- Lseek
- ....

进程的文件描述符表  
示例

|    |        |
|----|--------|
| 0  | stdin  |
| 1  | stdout |
| 2  | stderr |
| 3  | file   |
| 4  | device |
| 5  | socket |
| .. | ...    |

# Socket的类型

---

## ◆ 流式Socket(Stream Socket)

- 基于TCP，提供可靠的字节流传输
- `int s = socket (PF_INET, SOCK_STREAM, 0);`

## ◆ 数据报Socket(Datagram Socket)

- 基于UDP，提供不可靠的报文传输
- `int s = socket (PF_INET, SOCK_DGRAM, 0);`

## ◆ Raw Socket

- 基于IP，允许用户直接对IP操作
- `int s = socket (PF_INET, SOCKET_RAW, protocol);`

## ◆ 可靠交付报文Socket

- 数据报+确认
- `int s = socket (PF_NS, SOCK_RDM, 0);`

## ◆ 有序包流Socket

- 字节流+固定数据包长度
- `int s = socket (PF_NS, SOCKET_SEQPACKET, 0);`

# Socket地址（1）：因特网特定地址

- ◆ Internet-specific socket address, 在bits/socket.h中定义

```
struct sockaddr_in {
 unsigned short sin_family; /* address family (always AF_INET) */
 unsigned short sin_port; /* port num in network byte order */
 struct in_addr sin_addr; /* IP addr in network byte order */
 unsigned char sin_zero[8]; /* pad to sizeof(struct sockaddr) */
};
```

- ◆ Address family（地址族）：

- AF\_UNIX: for communication between processes on one system;
- AF\_INET (IPv4): for communication between processes on the same or different systems using the DARPA standard protocols (IP/UDP/TCP)
- AF\_INET6 (IPv6)
- AF\_LOCAL (Unix domain)
- AF\_UNSPEC (the importance will be explained later)
- ...

# Socket地址（2）：通用地址

---

## ◆ Generic socket address (<sys/socket.h>)

```
struct sockaddr {
 unsigned short sa_family; /* protocol family */
 char sa_data[14]; /* protocol-specific address,
 up to 14 bytes. */
};
```

## ◆ Protocol family（协议族）


- PF\_LOCAL: Local to host, pipes and file-domain
- PF\_UNIX: Old BSD name for PF\_LOCAL
- **PF\_INET: IP protocol family**
- PF\_AX25: Amateur radio AX.25
- PF\_IPX: Novell internet protocol
- PF\_INET6: IP version 6
- PF\_ATMSVC: ATM SVCs
- PF\_APPLETALK: Appletalk DDP
- ...



# 两种地址之间的关系

```
struct sockaddr {
 unsigned short sa_family; /* PF_INET for IPv4 */
 char sa_data[14]; /* protocol-specific address,
 up to 14 bytes. */
};
```

**Generic socket address**



```
struct sockaddr_in {
 unsigned short sin_family; /* AF_INET */
 unsigned short sin_port; /* 16-bit port number */
 struct in_addr sin_addr; /* 32-bit IP Address */
 char sin_zero[8]; /* unused */
};
```

**Internet-specific socket address**

# 两种地址之间的转换

---

- ◆ 许多Socket系统调用要求使用Generic socket address,
  - 例如: bind( ), connect(), accept()...
  - 此时要进行强制转换, 将Internet-specific socket address (sockaddr\_in \*) 转换成generic socket address (sockaddr \*)

```
struct sockaddr_in serv;
/* fill in serv{ } */
bind (sockfd, (struct sockaddr *)&serv, sizeof(serv));
```

# Socket程序设计：与打电话类比

---

## ◆ 一次电话呼叫过程：

- 通话双方都有一个电话机
- 每个电话机有一个电话号码
- 被叫用户需要打开铃声，以便接听
- 主叫用户摘机、拨号
- 被叫用户的电话振铃，摘机接听
- 双方通话、交换数据
- 通话结束，双方挂机

# 与网络通信类比

---

## ◆ 一个网络应用的工作过程：

- 通信双方各有一个通信端点`endpoint` (电话机)
- 每个端点有一个地址，作为通信的唯一标识 (电话号码)
- 一个端点(主叫用户) 发起向另一个端点的通信 (建立连接)
- 另一个端点(被叫用户) 等待建立连接
- 连接建立后，双方开始交换数据 (通话)
- 数据交换结束后，端点关闭 (挂机)

# 对应的Socket系统调用

---

- ◆ Socket() – 建立连接的端点
- ◆ Bind() - 分配电话号码
- ◆ Listen() – 等待有电话呼入
- ◆ Connect() - 拨号
- ◆ Accept() – 接收电话呼叫
- ◆ Send(), Recv() – 通话、交换数据
- ◆ Close() – 挂机

# Socket系统调用的分类

---

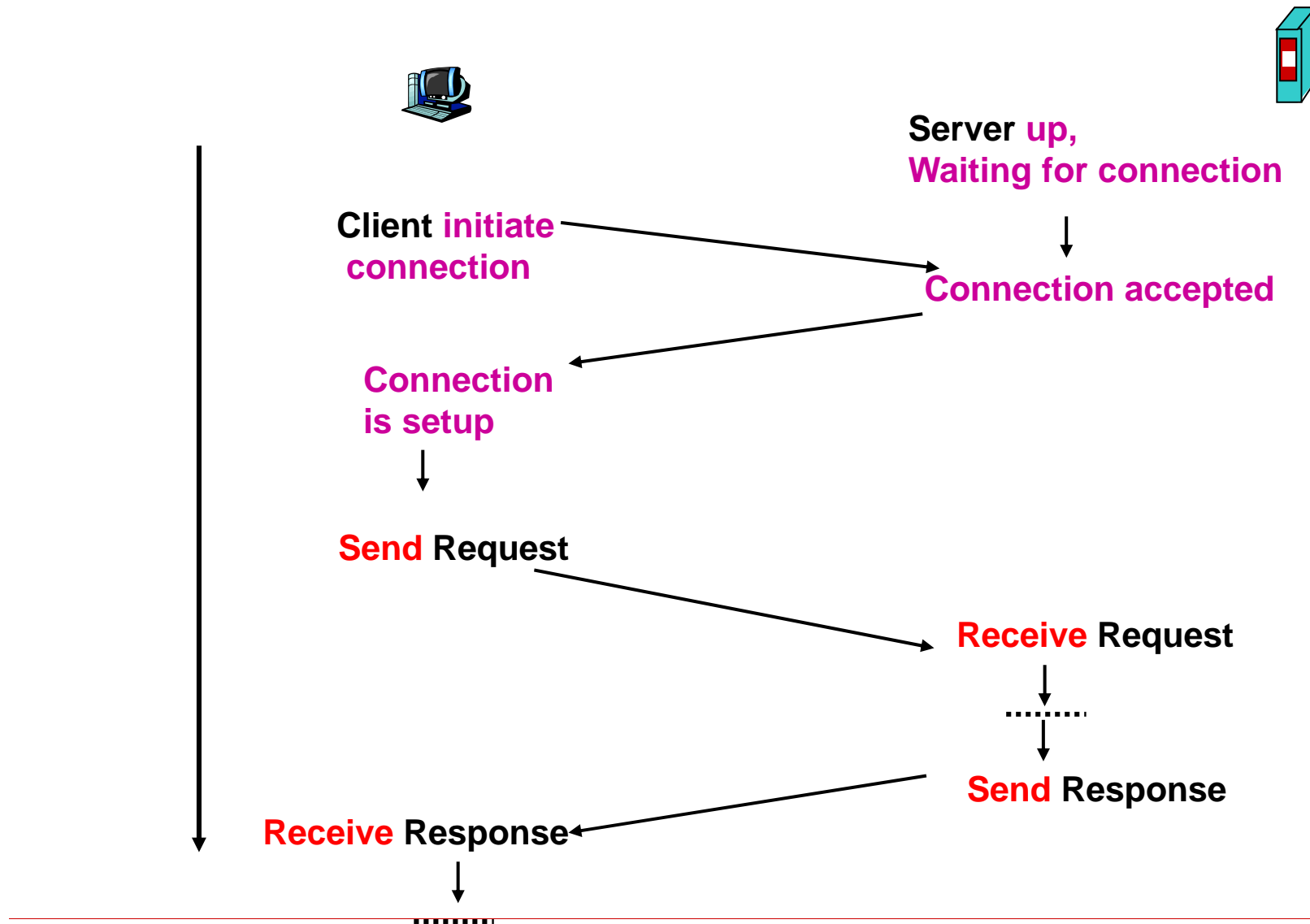
- ◆ Socket操作
- ◆ 字节顺序操作
- ◆ 地址格式操作
- ◆ Socket选项
- ◆ 名字和地址操作

# Socket操作的相关系统调用

---

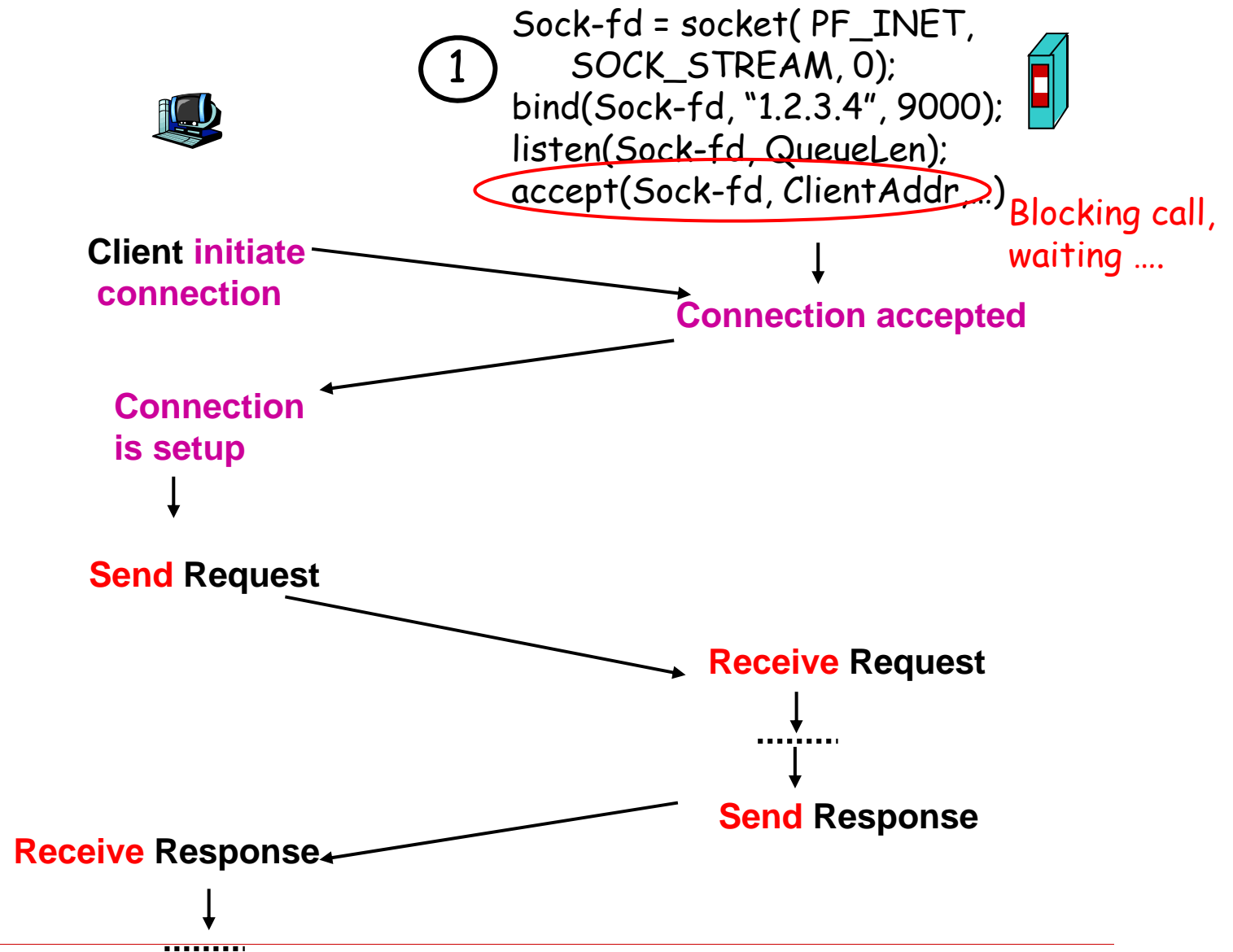
- ◆ `socket()` : 建立Socket端点, 获得Socket描述符
- ◆ `bind()` : Server绑定Socket地址 (IP地址+端口号)
- ◆ `listen()` : Server等待Client连接
- ◆ `connect()` : Client连接到Server
- ◆ `accept()` : Server获得连接请求的Client的Socket地址
- ◆ `send()` and `recv()`
  - 在已建立的连接上发送/接收数据 (TCP方式)
- ◆ `sendto()` and `recvfrom()`
  - 无需连接, 直接发送/接收数据 (UDP方式)
- ◆ `close()` and `shutdown()`
  - 关闭Socket (双向/单向)
- ◆ `readn()`, `writen()`, `readline()`
  - 读/写

# 基于TCP的Socket操作示例

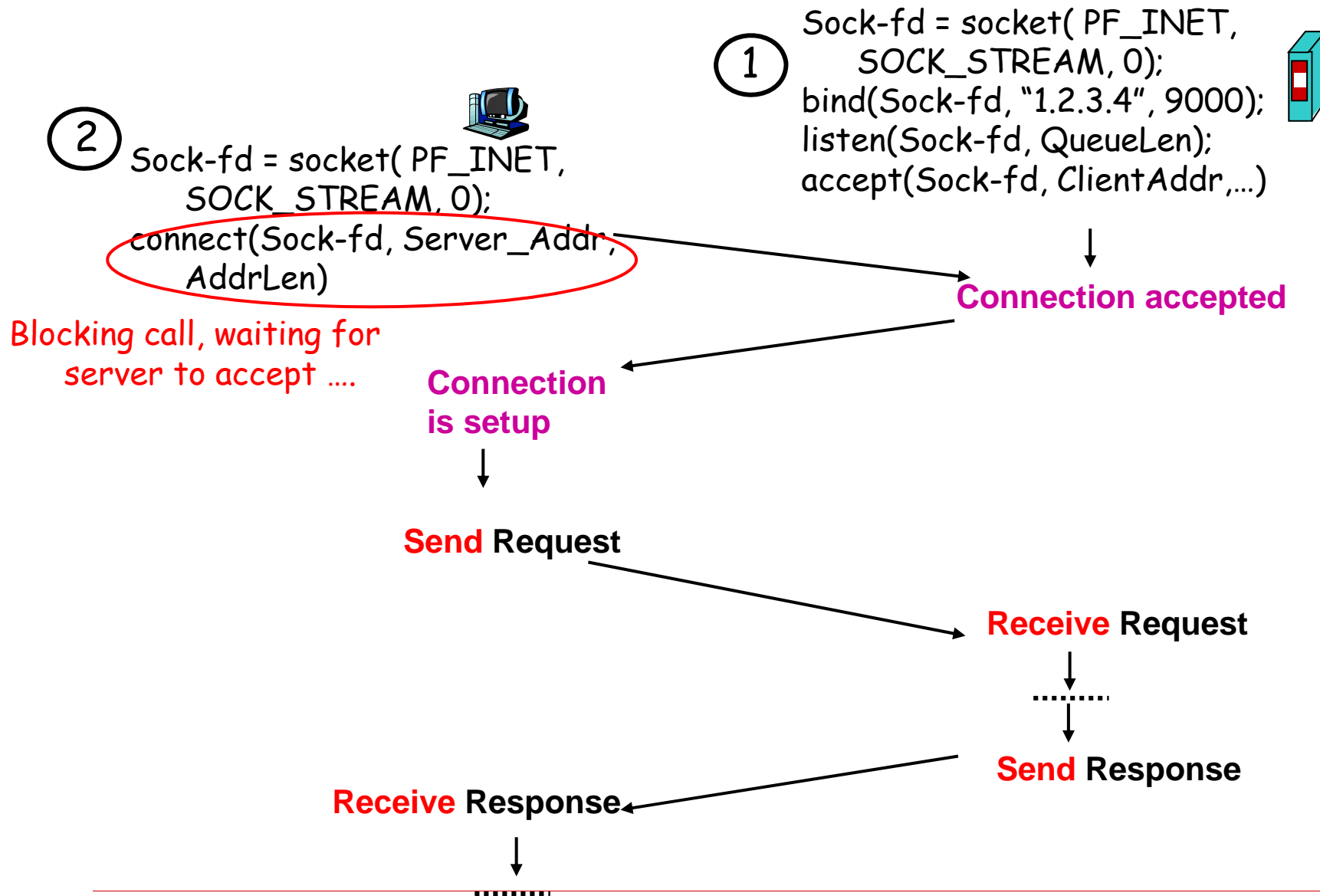




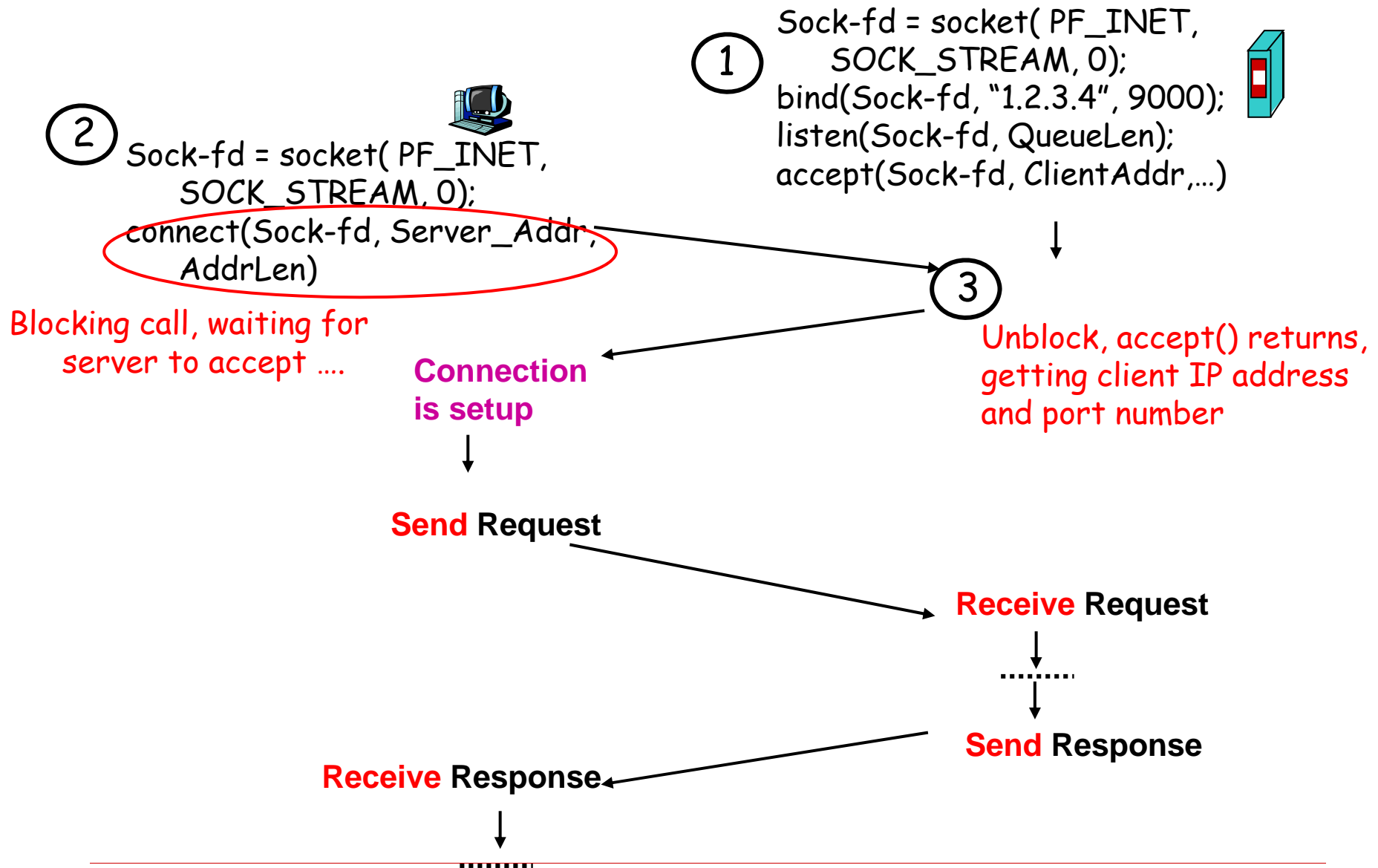
# 基于TCP的Socket操作示例



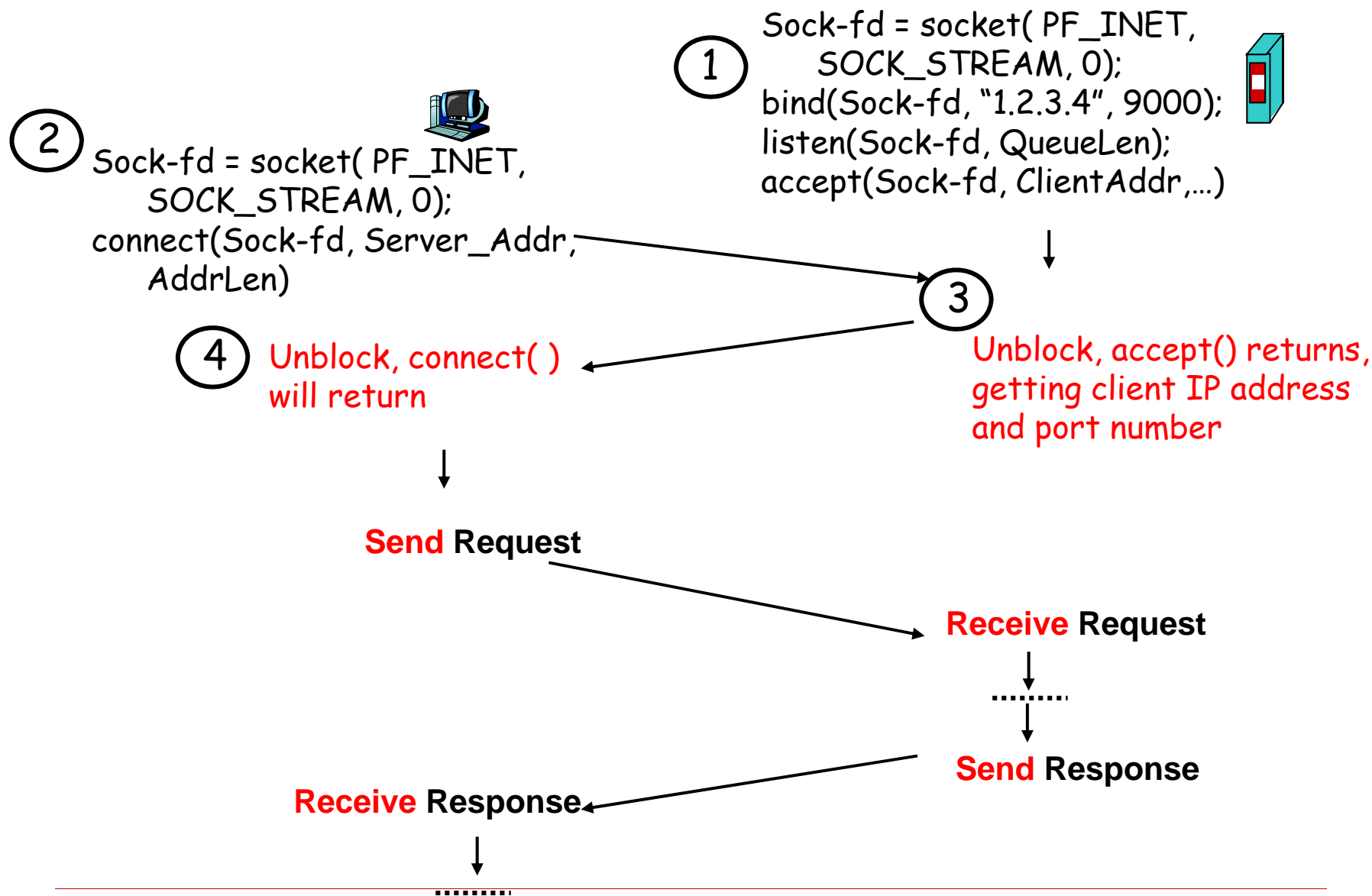
# 基于TCP的Socket操作示例



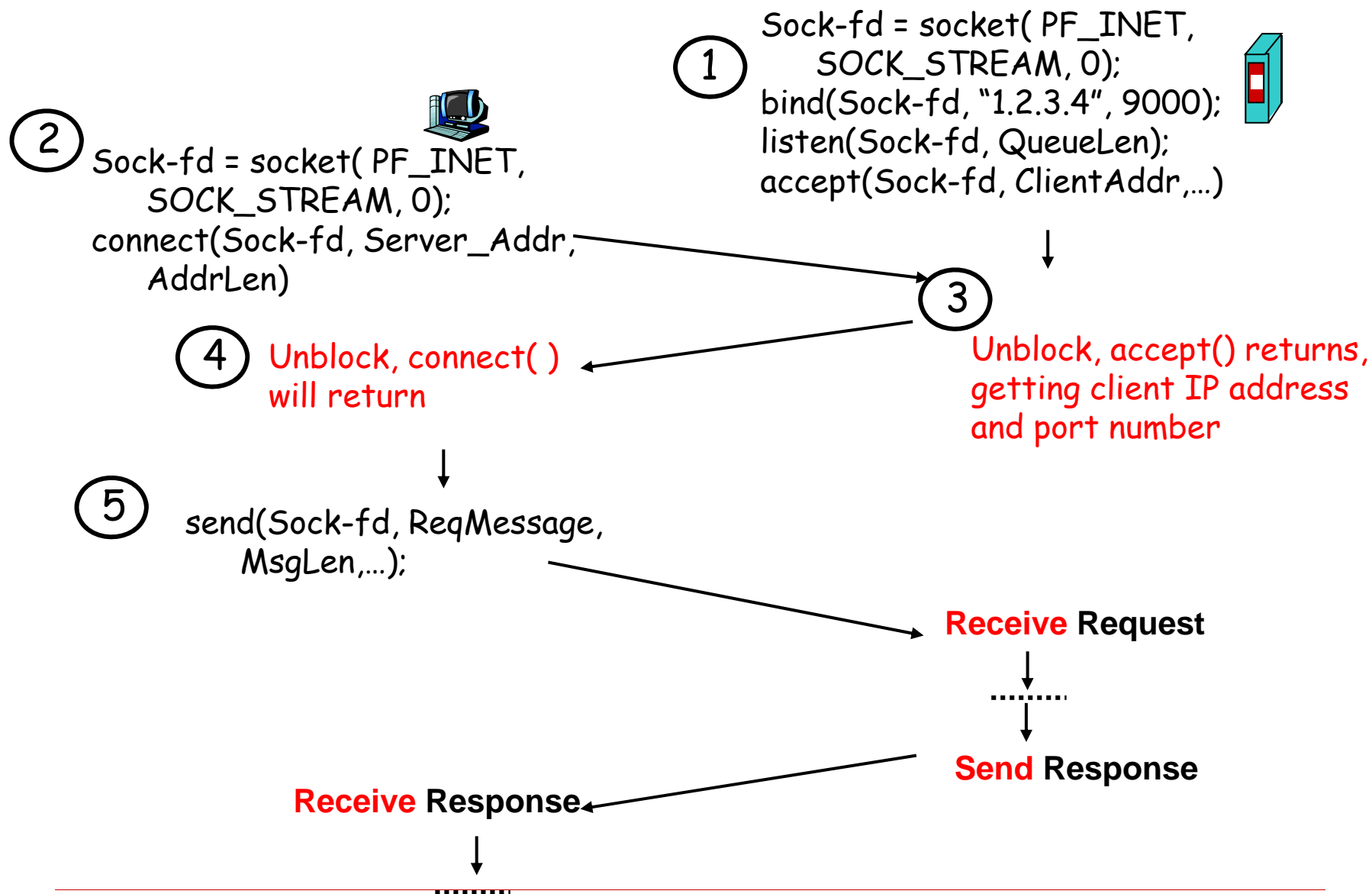
# 基于TCP的Socket操作示例



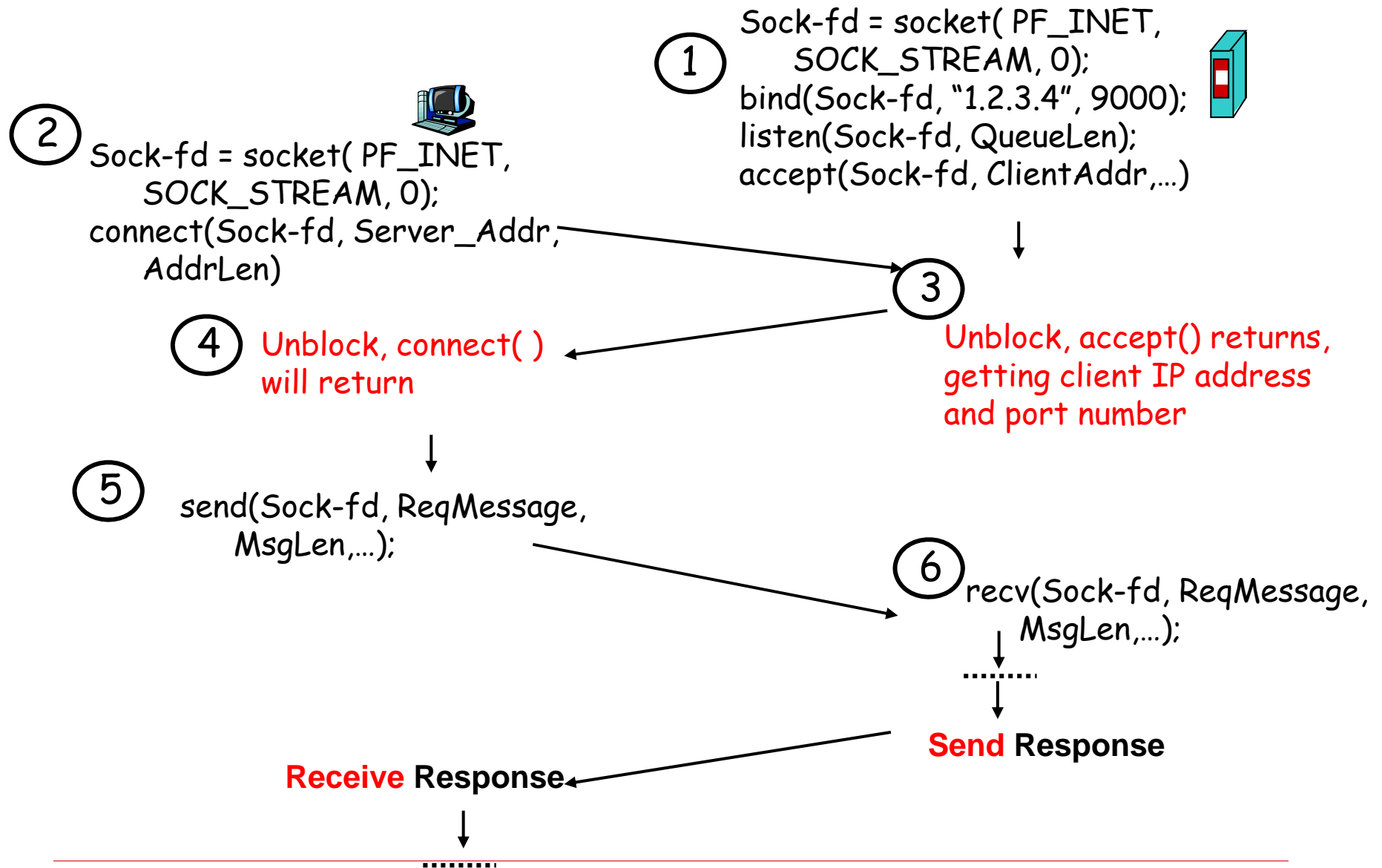
# 基于TCP的Socket操作示例



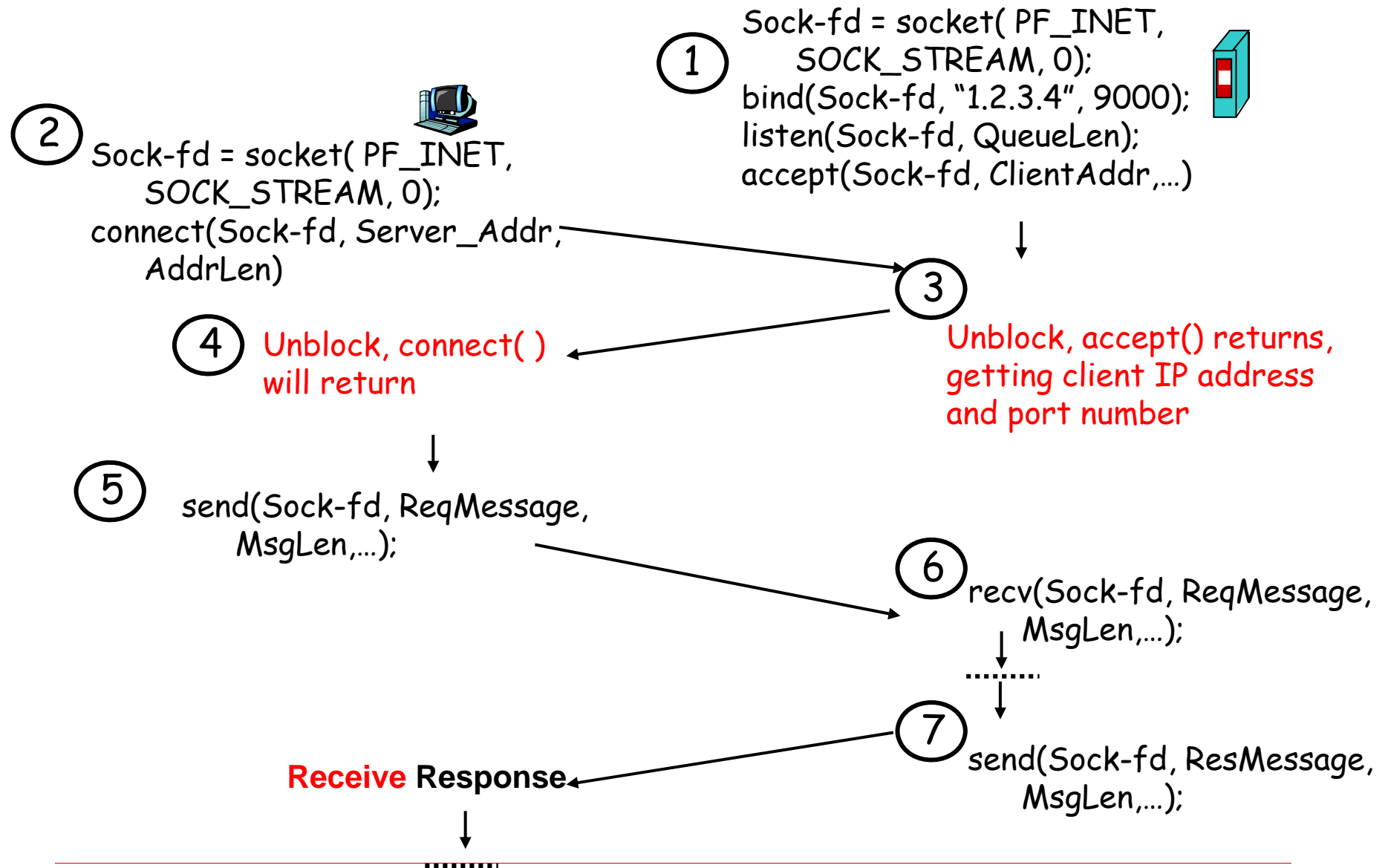
# 基于TCP的Socket操作示例



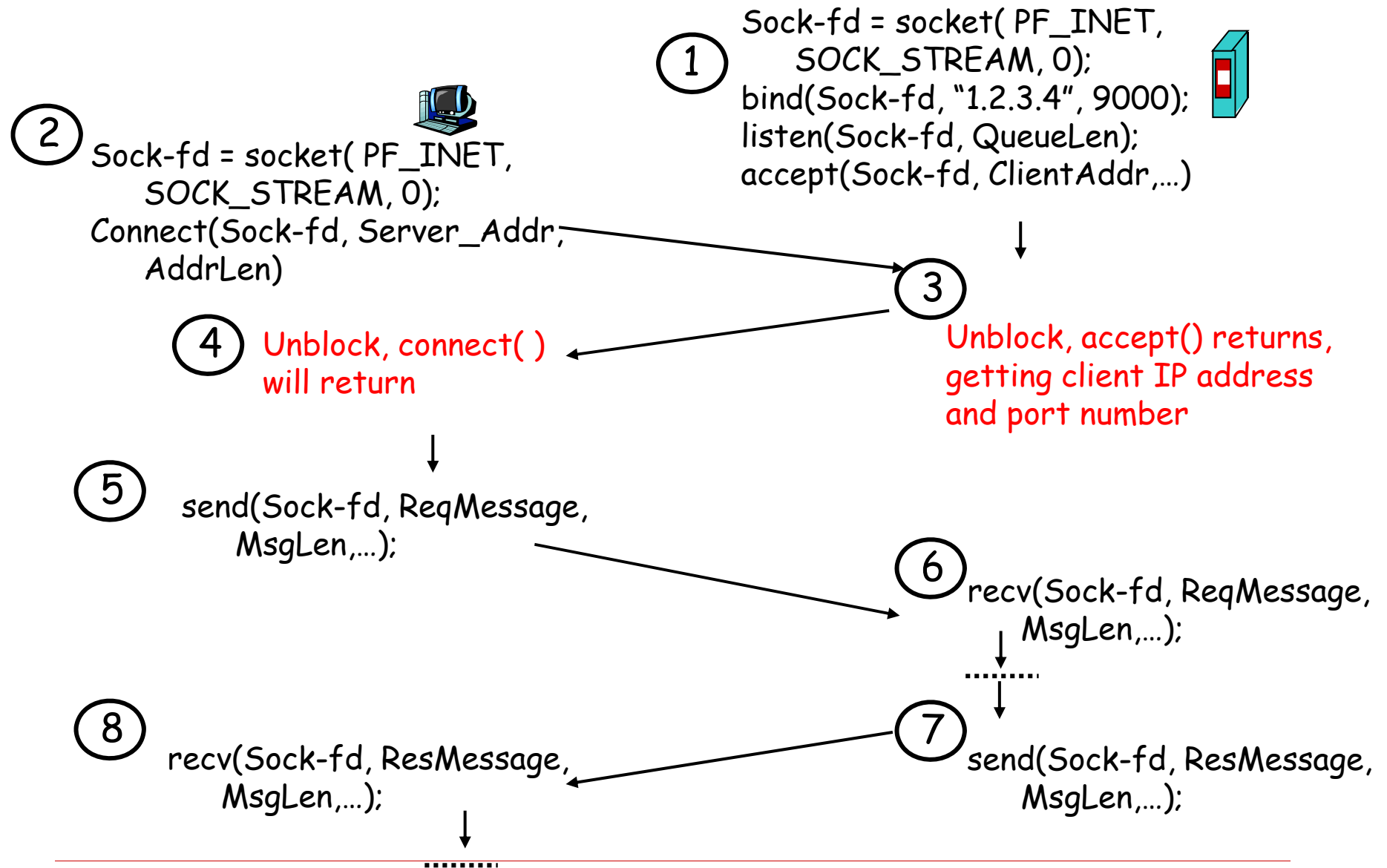
# 基于TCP的Socket操作示例



# 基于TCP的Socket操作示例



# 基于TCP的Socket操作示例





# 字节顺序 (Byte Order)问题

---

- ◆ 对于多字节数据（如IP地址、端口号）的存储和发送有不同的顺序
- ◆ 网络字节顺序（NBO）：Big-endian，首先传输低阶字节，高阶字节最后传输
- ◆ 主机字节顺序（HBO）：主机特定
  - 可能是Little-endian，首先存储高阶字节
  - 也可以是Big-endian，首先存储低阶字节
- ◆ 主机发送数据之前、接收数据之后，必须进行HBO与NBO之间的转换

# 字节顺序转换相关系统调用

---

## ◆ htonl()

- 将一个长整形数据从HBO转换给NBO

## ◆ htons()

- 将一个短整形数据从HBO转换给NBO

## ◆ ntohl()

- 将一个长整形数据从NBO转换给HBO

## ◆ ntohs()

- 将一个短整形数据从NBO转换给HBO

# 地址格式转换的系统调用

---

## ◆ 实现IP地址的ASCII字符串格式与NBO格式之间的转换

### ◆ `inet_aton()`

- 将IP地址从点分十进制数（ASCII字符串）格式转换成NBO的无符号长整型格式

### ◆ `inet_addr()`

- 将IP地址从点分十进制数（ASCII字符串）格式转换成NBO的无符号长整型格式

### ◆ `inet_ntoa()`

- 将IP地址从NBO的无符号长整型格式转换成点分十进制数（ASCII字符串）格式

### ◆ `inet_pton()`

- 类似`inet_aton()`，IPv4和IPv6通用

### ◆ `inet_ntop()`

- 类似`inet_ntoa()`，IPv4和IPv6通用

# Socket选项系统调用

---

## ◆ getsockopt()

- 查询Socket的相关信息

## ◆ setsockopt()

- 设置Socket选项

## ◆ 示例：

- 查询/设置 一个Socket的发送/接收缓存大小

# 名字和地址操作相关的系统调用

---

## ◆ `gethostbyname()`

- 使用域名从DNS数据库查询主机的信息（别名、IP地址）

## ◆ `gethostbyaddr()`

- 使用IP地址从DNS数据库查询主机的信息（域名）

## ◆ `gethostname()`

- 查询主机的规范名(Canonical Name)

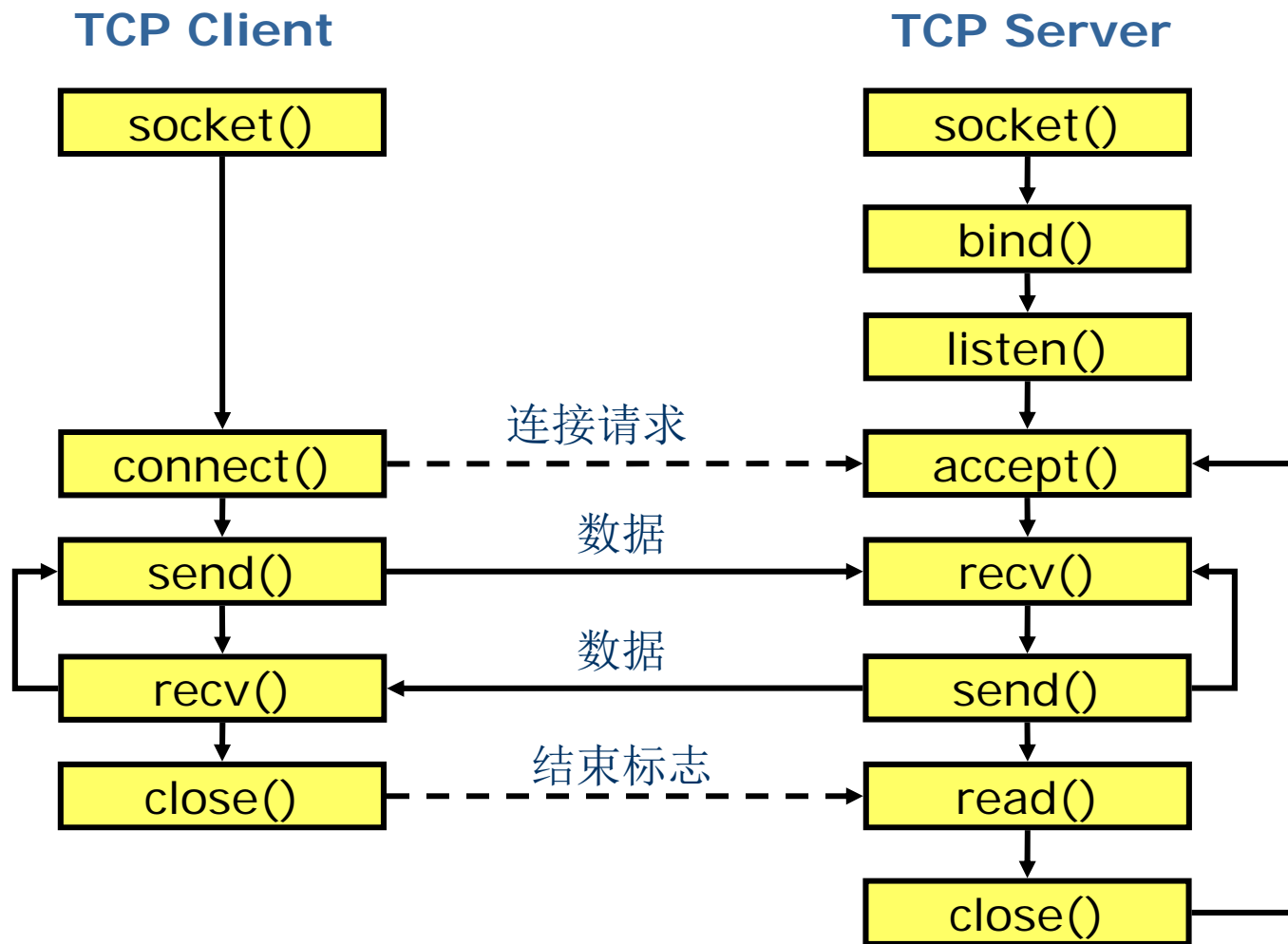
## ◆ `getservbyname()`

- 查询指定服务的端口号

## ◆ `getservbyport()`

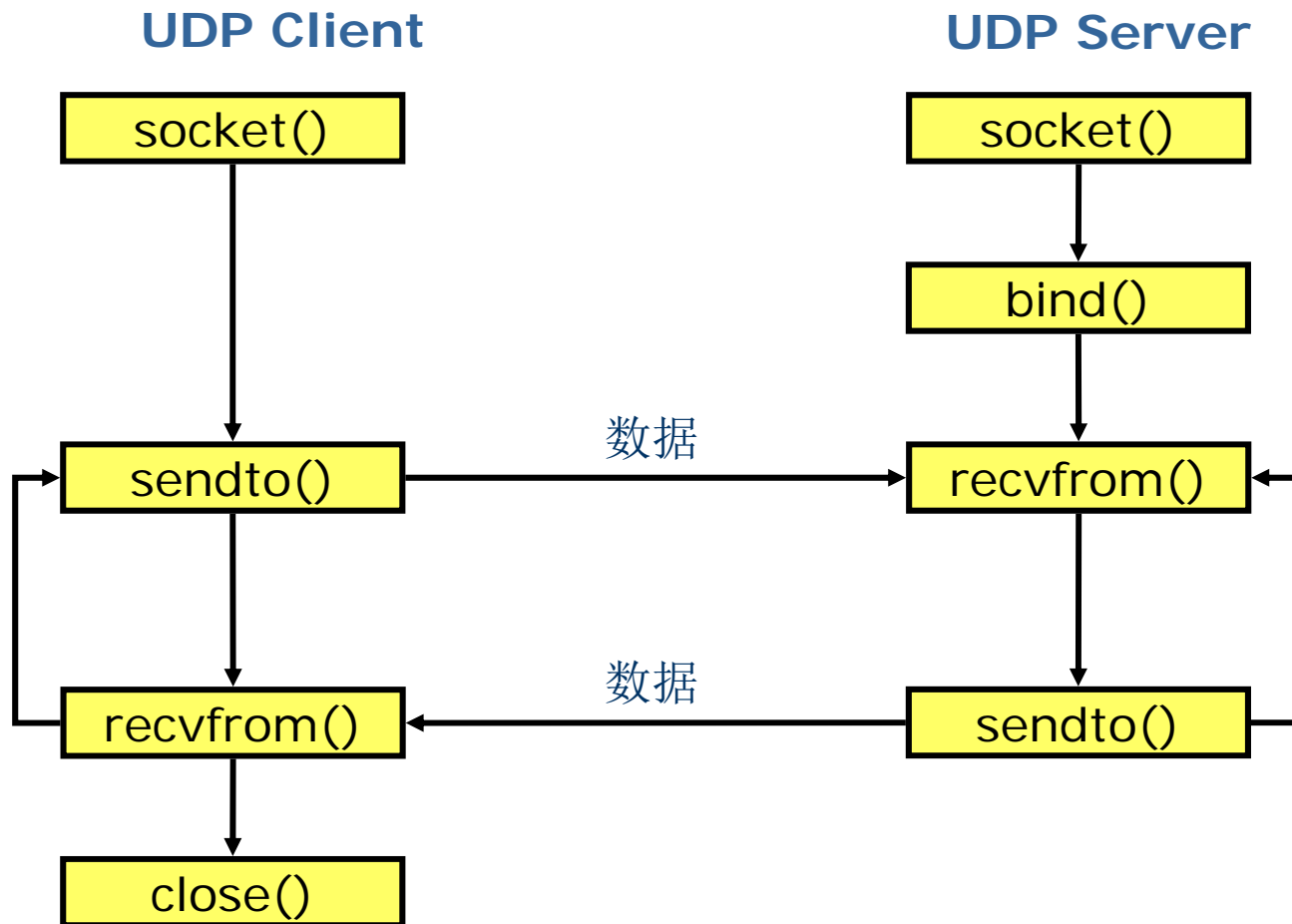
- 查询指定端口号对应的服务名

# 基于TCP的Socket程序流程

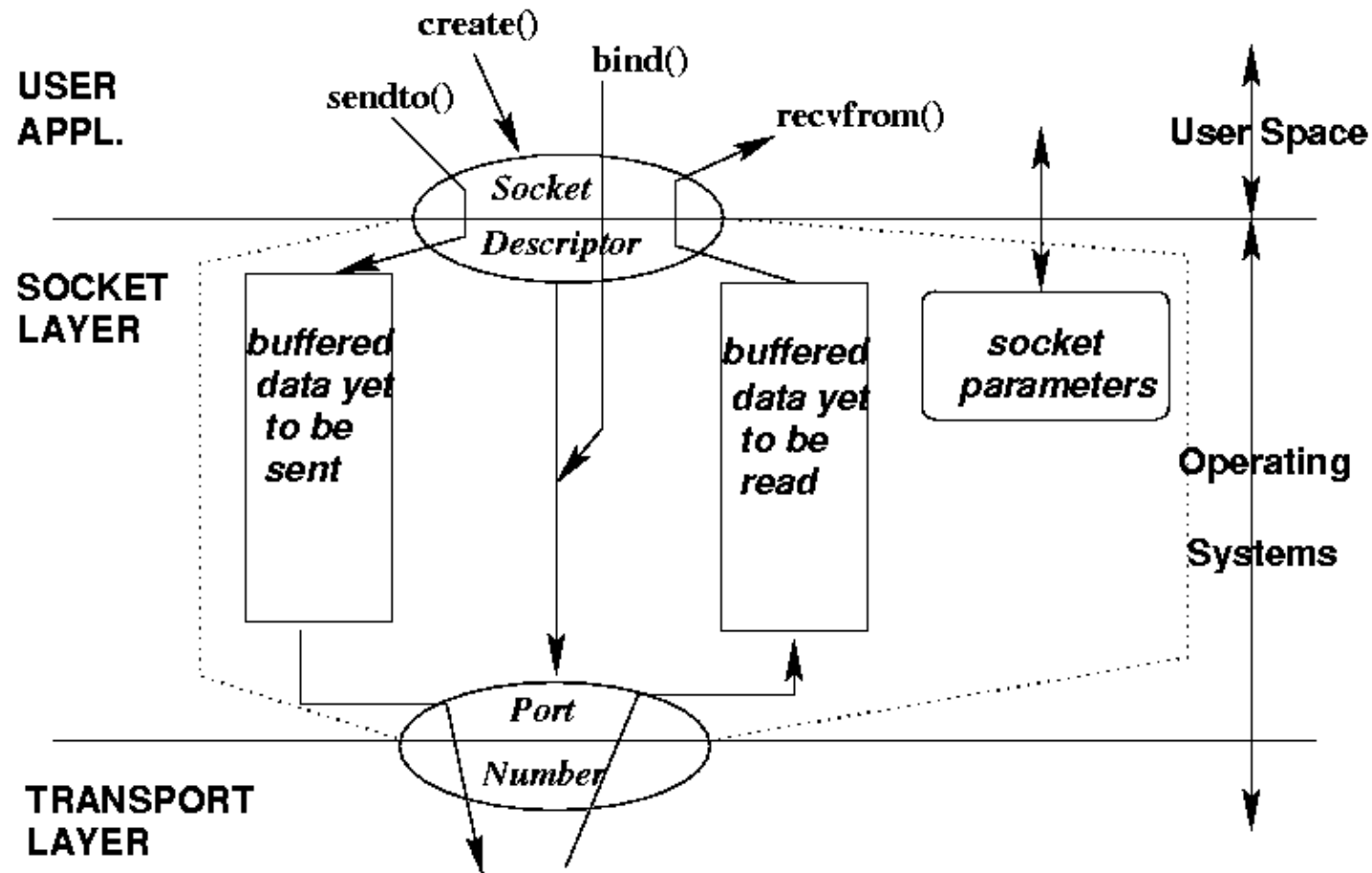


# 基于UDP的Socket程序流程

---



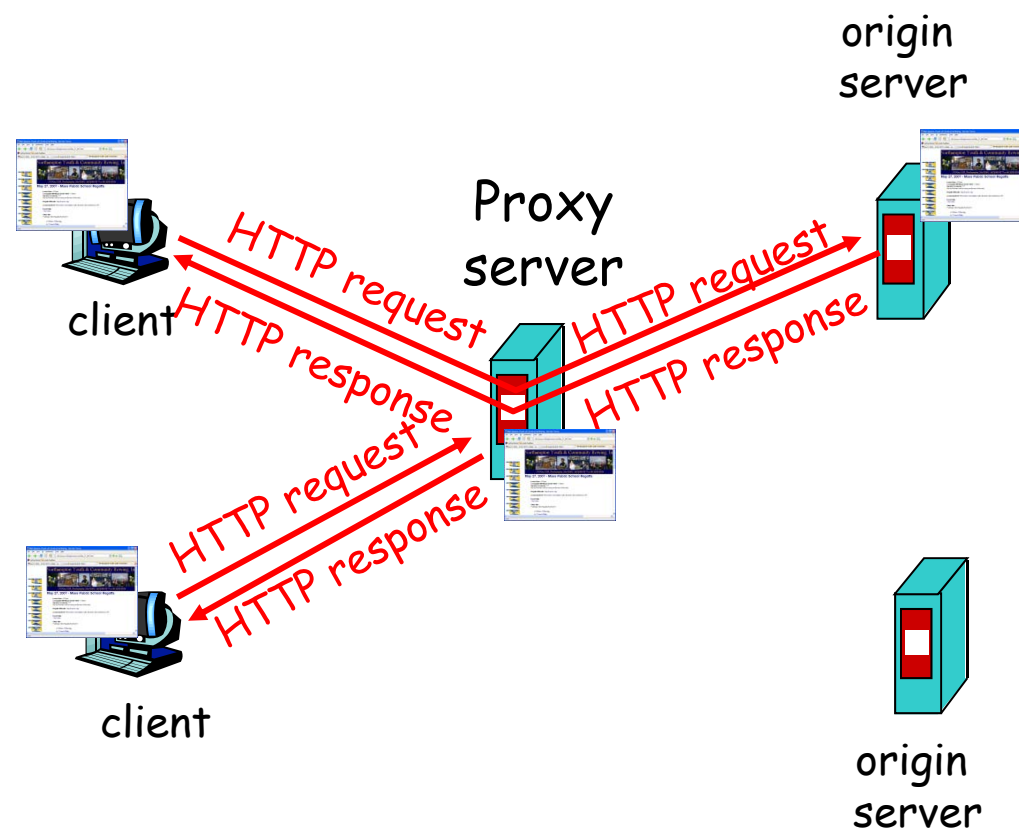
# Socket通信的概念示图





# HTTP代理服务器

- ◆ 在浏览器程序中设置通过代理访问
- ◆ 浏览器与Web服务器之间的中转
- ◆ 浏览器将所有HTTP请求均发送给代理
  - 如果在代理中找到请求的对象，则由代理返回响应
  - 否则代理将请求转发给要访问的Web Server，由该Server 响应



# 代理服务器的基本功能

---

- ◆ 将来自任意网络浏览器（如IE）的HTTP请求转发给Web服务器，将来自Web服务器的响应转发给请求的浏览器；
- ◆ 记录并显示浏览器、服务器的主要信息（IP地址、域名、操作系统版本、显示语言、HTTP版本等）；
- ◆ 缓存访问的网页，在浏览器再次访问时，直接返回响应，不必再去访问源服务器；
- ◆ 设计黑名单，如果浏览器要访问黑名单中的Web服务器，则返回告警页面（告警页面的内容自行设计）。

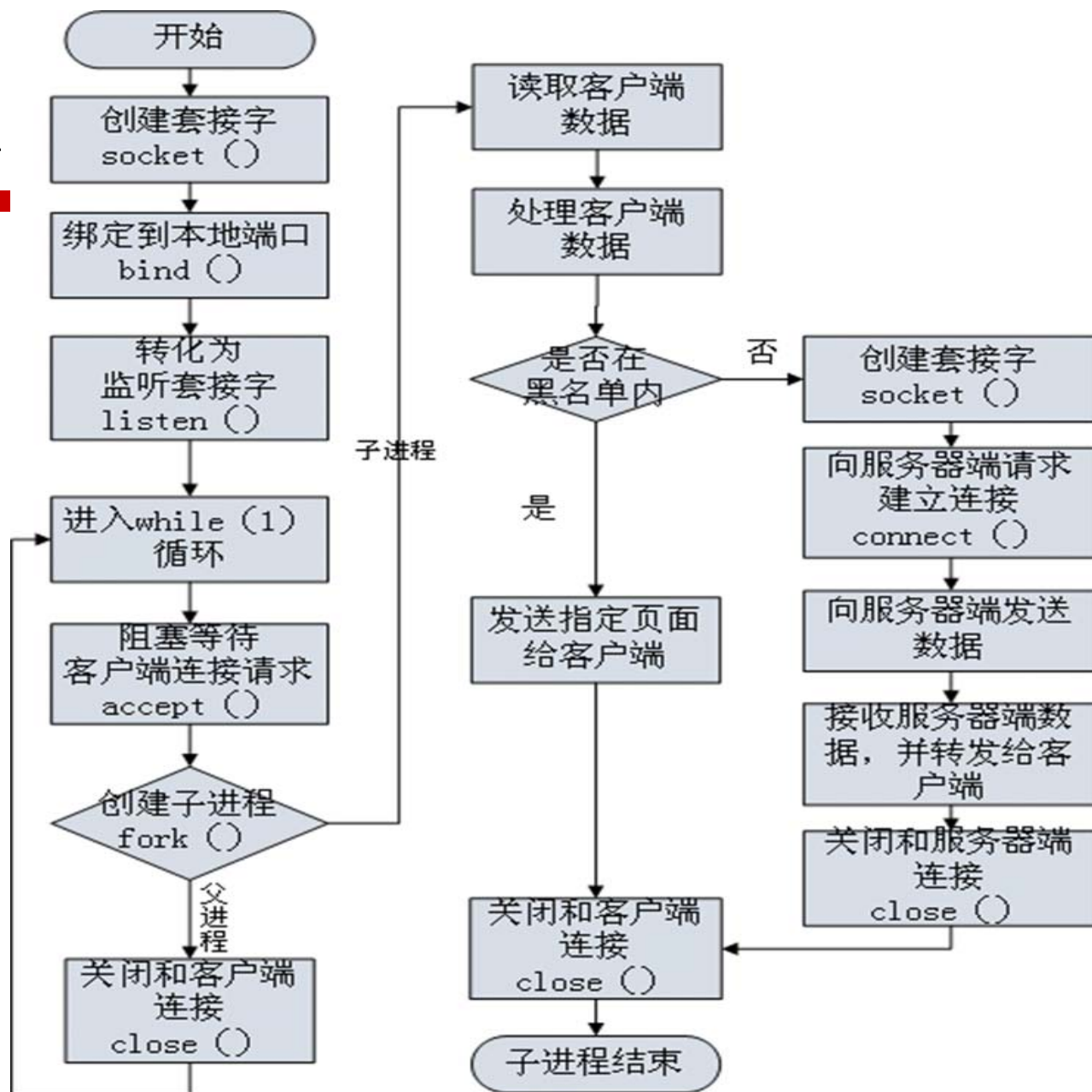
# 代理服务器的扩展功能

---

- ◆ 允许多个浏览器同时访问；
- ◆ 对于访问用户进行身份认证；
- ◆ 在转发服务器的响应页面上增加显示：“本网页由代理服务器转发”。

# 整体流程

- ◆ 以Linux为例
- ◆ C语言
- ◆ 基本功能



# 其他要求

---

- ◆ 完成方式：每两人一组
- ◆ 验收截止日期：12月10日左右
- ◆ 报告截止日期：12月17日
- ◆ 文档内容：
  - 实验报告，程序源码
  - 压缩为rar格式文件
- ◆ 文件命名
  - 班级\_学号1\_学号2.rar
  - 例如：16班\_2013211123\_2013211234.rar