Technology Tarena 达内科技

Unix系统高级编程

Network DAY13

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	网络与网络协议
	10:30 ~ 11:20	套接字
	11:30 ~ 12:20	TCP客户机/服务器
下午	14:00 ~ 14:50	UDP客户机/服务器
	15:00 ~ 15:50	基于并发进程的网络银行
	16:00 ~ 16:50	
	17:00 ~ 17:30	总结和答疑

网络与网络协议


网络与网络协议
网络协议模型
消息与地址
ISO/OSI网络协议模型
TCP/IP协议族
TCP/IP与ISO/OSI模型
消息包与消息流
IP地址

Technology Tarena 达内科技

网络协议模型

ISO/OSI网络协议模型

- 什么是计算机网络?
  - 计算机网络，是指将地理位置不同的具有独立功能的多台计算机及其外部设备，通过通信线路连接起来，在网络操作系统、网络管理软件及网络通信协议的管理和协调下，实现资源共享和信息传递的计算机系统

知识讲解

ISO/OSI网络协议模型（续1）

Technology Tarena 达内科技

- 什么是网络协议？
  - 网络协议是一种特殊的软件，是计算机网络实现其功能的最基本的机制。网络协议的本质就是规则，即各种硬件和软件必须遵循的共同守则。网络协议并不是一套单独的软件，它融合于其它所有的软件甚至硬件系统中，因此可以说协议在网络中无所不在

知识讲解

ISO/OSI网络协议模型（续2）

Technology Tarena 达内科技

- 什么是协议栈?
  - 为了减少网络设计的复杂性，绝大多数网络采用分层设计的方法。所谓分层设计，就是按照信息的流动过程将网络的整体功能分解为一个个的功能层，不同机器上的同等功能层之间采用相同的协议，同一机器上的相邻功能层之间通过接口进行信息传递。各层的协议和接口统称为协议栈

知识讲解

ISO/OSI网络协议模型（续3）

Technology Tarena 达内科技

- 描述计算机网络各协议层的一般方法是采用国际标准化组织(International Standardization Organization, ISO)的计算机通信开放系统互连(Open System Interconnection, OSI)模型，简称ISO/OSI网络协议模型

知识讲解

上层协议定义网络数据的格式及各种网络应用	应用层	为用户的应用程序提供各种网络服务	http/ftp/telnet
	表示层	将不同数据格式转换为一种通用格式	ascii/jpeg/mpeg
	会话层	建立、管理和终止通信主机间的对话	安排访问次序
底层协议定义数据如何传输到目的地	传输层	提供端到端的流量控制维持可靠传输	TCP/UDP
	网络层	路径选择、路由、IP寻址、建立连接	IP/SPX、路由器
	数据链路层	物理寻址、网络拓扑结构、错误检测	MAC、网桥
	物理层	高低电平、速率与距离、物理连接器	HUB/中继器/线缆

TCP/IP协议族

- TCP/IP不是个单一的网络协议，而是由一组具有层次关系的网络协议组成的协议家族，简称TCP/IP协议族
  - TCP：传输控制协议，面向连接，可靠的全双工的字节流
  - UDP：用户数据报协议，无连接，不如TCP可靠但速度快
  - ICMP：网际控制消息协议，处理路由器和主机间的错误和控制消息
  - IGMP：网际组管理协议，用于多播
  - IPv4：网际协议版本4，使用32位地址，为TCP、UDP、ICMP和IGMP提供递送分组服务
  - IPv6：网际协议版本6，使用128位地址，为TCP、UDP和ICMPv6提供递送分组服务

知识讲解

TCP/IP协议族（续1）

Technology Tarena 达内科技

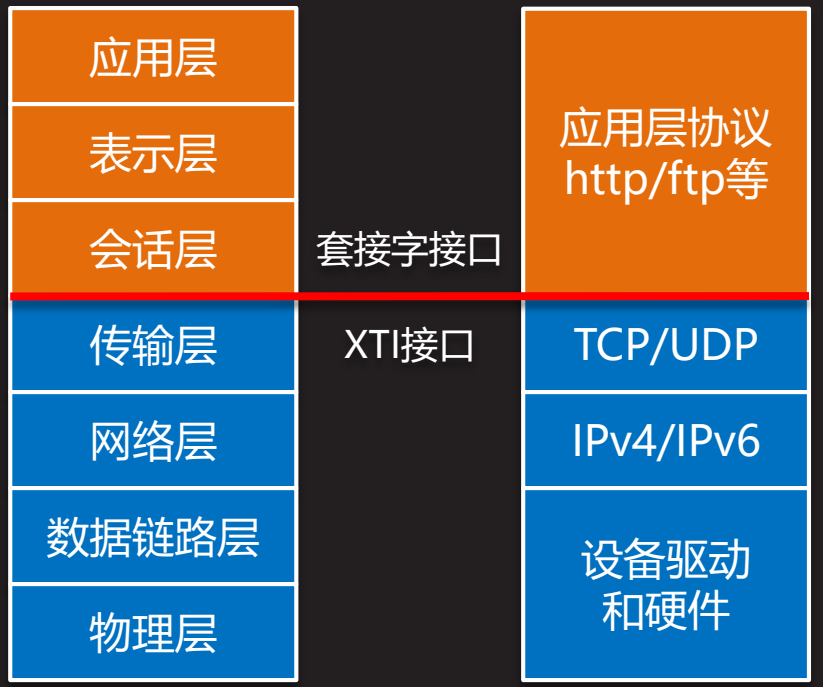
- TCP/IP不是个单一的网络协议，而是由一组具有层次关系的网络协议组成的协议家族，简称TCP/IP协议族
  - ARP：地址解析协议，把IPv4地址映射到硬件地址
  - RARP：逆地址解析协议，把硬件地址映射到IPv4地址
  - ICMPv6：网际控制消息协议版本6，综合了ICMP、IGMP和ARP的功能
  - BPF：BSD分组过滤器，为应用程序提供访问数据链路层的接口，由源自BSD的系统内核提供
  - DLPI：数据链路提供者接口，为应用程序提供访问数据链路层的接口，由源自SVR4的系统内核提供
- 通常所说的TCP、UDP的ICMP等协议都是工作在IP协议之上的，IP协议作为它们的基础协议为其提供服务支撑

TCP/IP与ISO/OSI模型

Technology Tarena 达内科技

知识讲解

- 在ISO/OSI网络协议模型的基础上，TCP/IP协议做了部分合并和简化，同时将网络编程的接口设定在传输层与会话层之间，这样做的理由有二
  - 上三层与应用程序的业务逻辑(如数据包的组织与解析、收发的时机与次序等)密切相关，而与具体的通信细节(如收发分组、等待确认、分组排序、计算验证校验和、丢包重传等)关系不大；下四层主要处理通信细节而与具体应用的业务逻辑无关



  - 上三层通常构成用户进程，而下四层通常是系统内核的一部分

Technology Tarena 达内科技

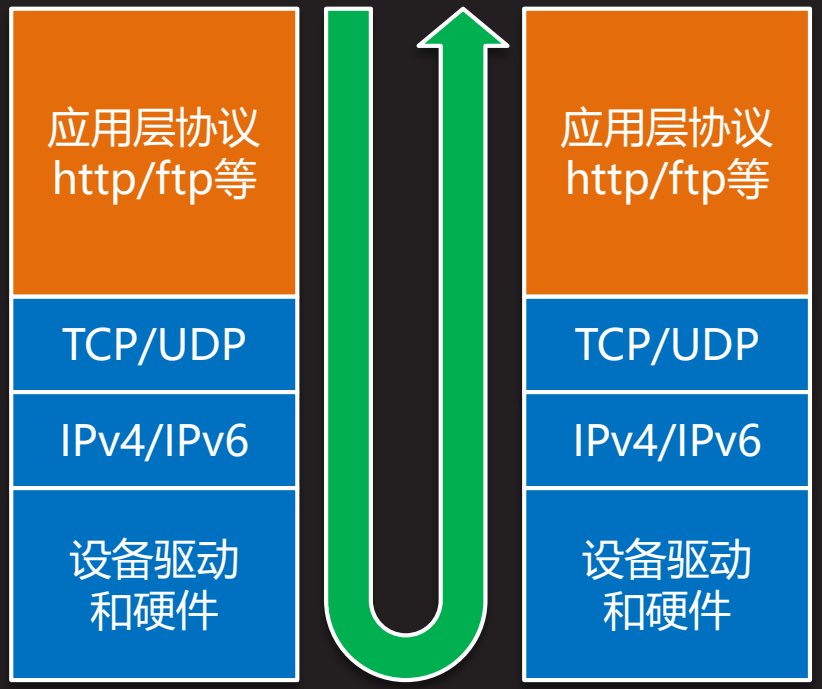
消息与地址

消息包与消息流

Technology Tarena 达内科技

知识讲解

- 应用程序负责组织的通常都是与业务相关的数据内容，而要想把这些数据内容通过网络发送出去，就要将其自上向下地压入协议栈，每经历一个协议层，就会对数据做一层封包，每一层输出的封包都是下一层输入的内容，消息包沿着协议栈的运动形成了消息流
- 当从网络上接收数据时，过程刚好相反，消息包自下向上地流经协议栈，每经历一个协议层，就会对输入的数据解一层封包，经过层层解包以后，应用程序最终得到的将只是与业务相关的数据内容



IP地址

Technology Tarena 达内科技

- 什么是IP地址?
  - IP地址，全称网际协议地址(Internet Protocol Address)，是IP协议提供的一种统一的地址格式，为互联网上的每个网络和每台主机分配一个逻辑地址，借以消除物理地址差异性所带来的影响
- IP地址如何表示?
  - 在计算机内部，IP地址用一个32位无符号整数表示，如：0x01020304。如无特别说明本课程只讨论IPv4的情况
  - 人们更习惯使用点分十进制字符串表示，如：1.2.3.4。字符串形式的从左到右，对应整数形式的从高字节到低字节。注意这里所说的高低指的是数位高低而非地址高低

知识讲解

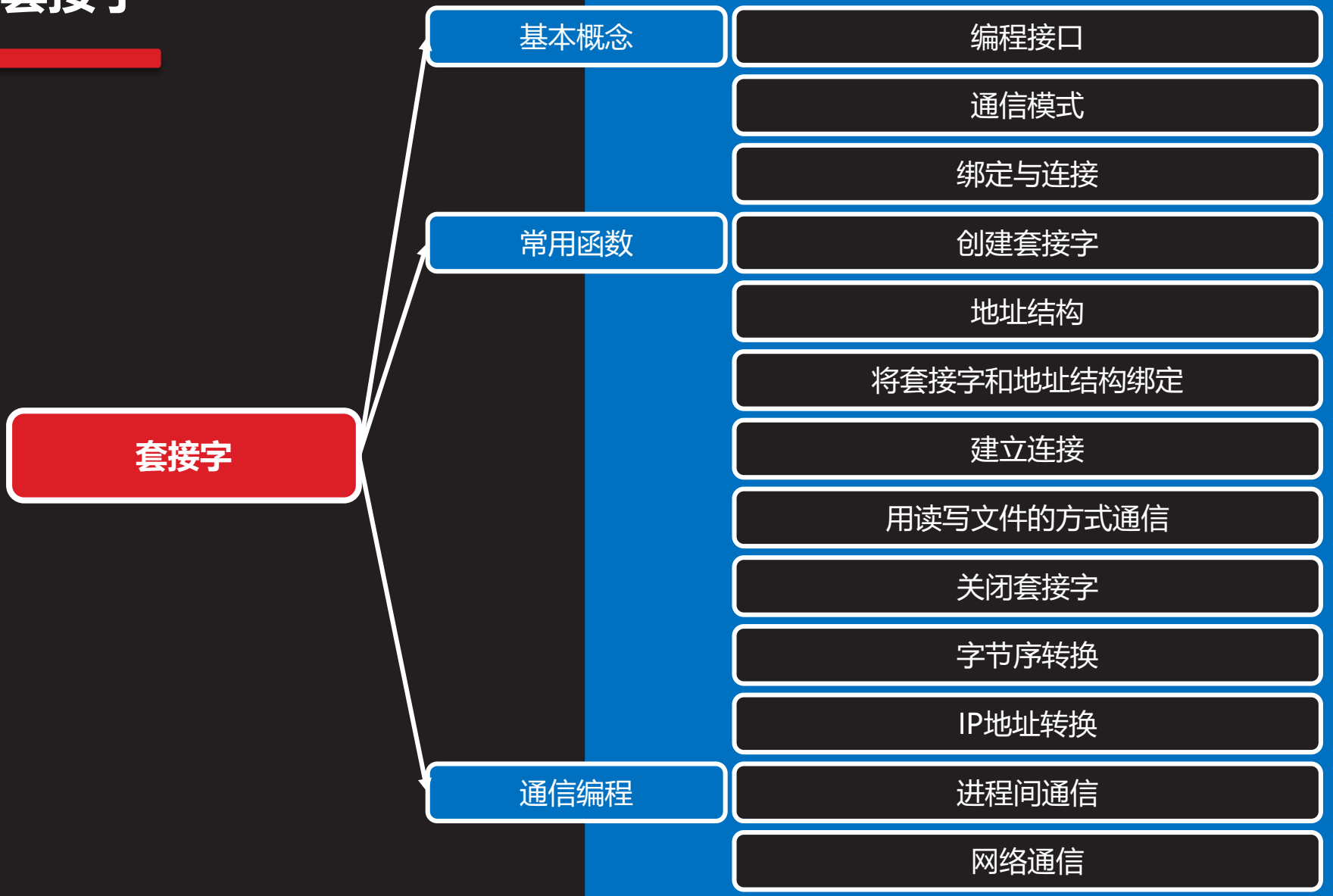
IP地址（续1）

Technology Tarena 达内科技

- 什么是IP地址分级？
  - A级地址：以0为首的8位网络地址+24位本地地址
  - B级地址：以10为首的16位网络地址+16位本地地址
  - C级地址：以110为首的24位网络地址+8为本地地址
  - D级地址：以1110为首的32位多播地址
  - 例如：某台计算机的IP地址：192.168.182.48，写成整数形式：11000000　10101000　10110110　00110000，C级地址，网络地址：192.168.182.0，本地地址：48
- 什么是子网掩码？
  - IP地址 & 子网掩码 = 网络地址
    192.168.182.48 & 255.255.255.0 = 192.168.182.0

知识讲解

套接字


套接字
基本概念
编程接口
通信模式
绑定与连接
常用函数
创建套接字
地址结构
将套接字和地址结构绑定
建立连接
用读写文件的方式通信
关闭套接字
字节序转换
IP地址转换
通信编程
进程间通信
网络通信

Technology Tarena 达内科技

基本概念

编程接口

Technology Tarena 达内科技

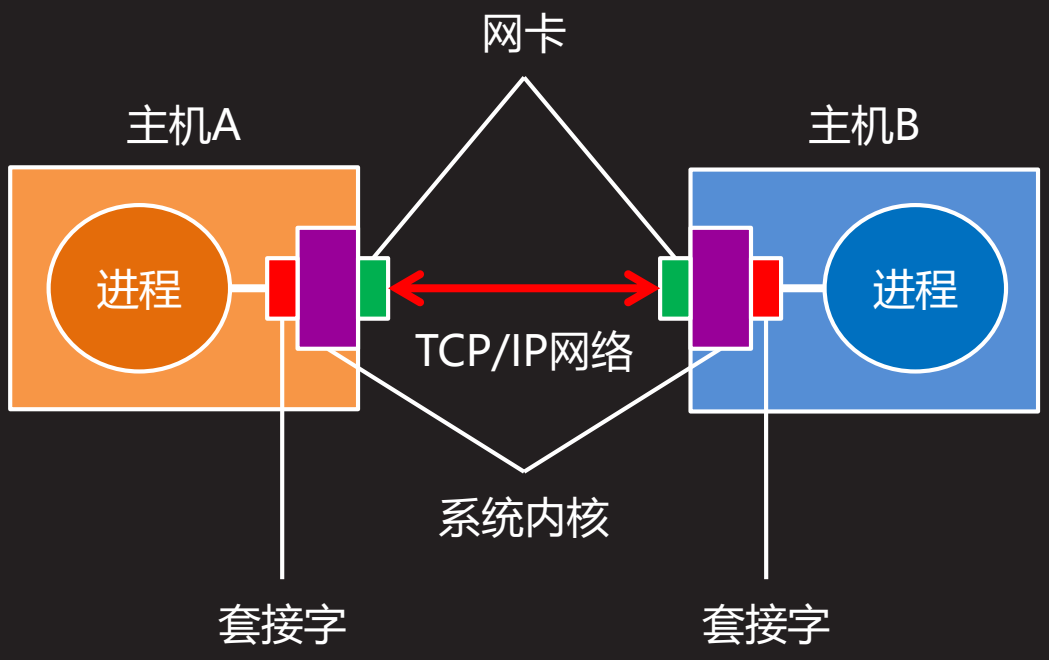
- 什么是伯克利套接字(Berkeley Socket)?
  - 美国加利福尼亚大学伯克利分校(University of California-Berkeley, UC Berkeley)于1983年发布4.2 BSD Unix系统。其中包含一套用C语言编写的应用程序开发库。该库既可用于在同一台计算机上实现进程间通信，也可用于在不同计算机上实现网络通信。当时的Unix还受AT&T的专利保护，因此直到1989年，伯克利大学才能自由发布他们的操作系统和网络库，而后者即被称为伯克利套接字应用编程接口(Berkeley Socket APIs)
  - 伯克利套接字接口的实现完全基于TCP/IP协议，因此它是构建一切互联网应用的基石。几乎所有现代操作系统都或多或少有一些源自伯克利套接字接口的实现。它已成为应用程序连接互联网的标准接口

知识讲解

编程接口（续1）

Technology Tarena 达内科技

- 什么是套接字?
  - 套接字(socket)的本意是指电源插座，这里将其引申为一个基于TCP/IP协议可实现基本网络通信功能的逻辑对象
  - 机器与机器的通信，或者进程与进程的通信，在这里都可以被抽象地看作是套接字与套接字的通信
  - 应用程序编写者无需了解网络协议的任何细节，更无需知晓系统内核和网络设备的运作机制，只要把想发送的数据写入套接字，或从套接字中读取想接收的数据即可

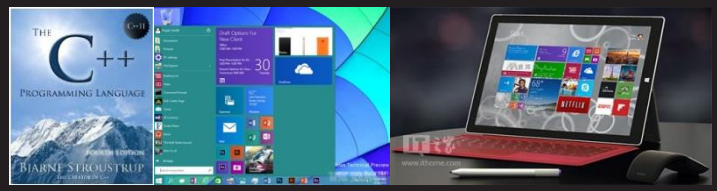


编程接口（续2）

Technology Tarena 达内科技

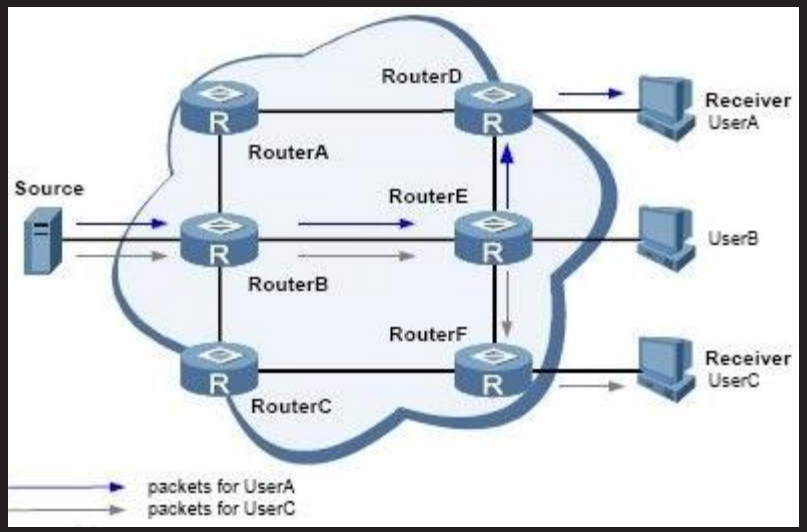
知识讲解

- 什么是套接字?
  - 从这个意义上讲，套接字就相当于一个文件描述符，而网络就是一种特殊的文件，面向网络的编程与面向文件的编程已没有分别，而这恰恰是Unix系统一切皆文件思想的又一例证
- 什么是套接字的异构性?
  - 如前所述，套接字是对ISO/OSI网络协议模型中传输层及其以下诸层的逻辑抽象，是对TCP/IP网络通信协议的高级封装，因此无论所依赖的是什么硬件，所运行的什么操作系统，所使用的是什么编程语言，只要是基于套接字构建的应用程序，只要是在互联网环境中通信，就不会存在任何障碍



通信模式

- 单播模式
  - 每个数据包发往单个目的主机，目的地址指明单个接收者
  - 服务器可以及时响应客户机的请求
  - 服务器可以针对不同客户的不同请求提供个性化的服务
  - 网络中传输的信息量与请求该信息的用户量成正比，当请求该信息的用户量较大时，网络中将出现多份内容相同的信息流，此时带宽就成了限制传输质量的瓶颈



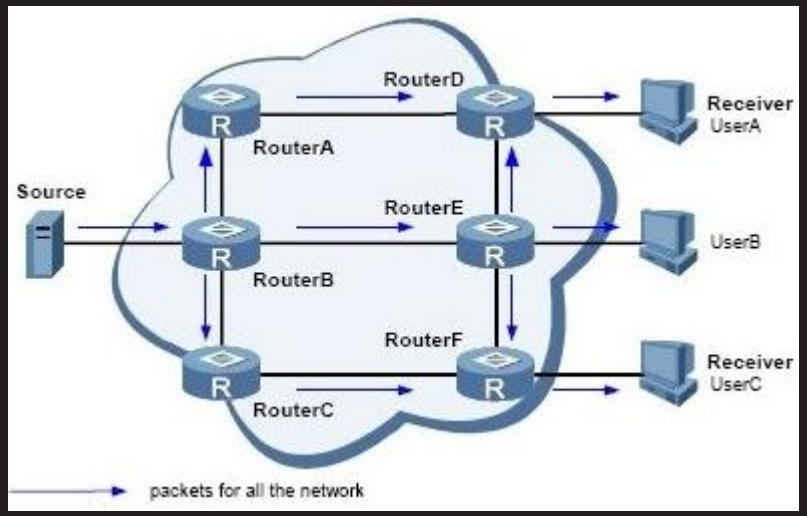
知识讲解

通信模式（续1）

Technology Tarena 达内科技

知识讲解

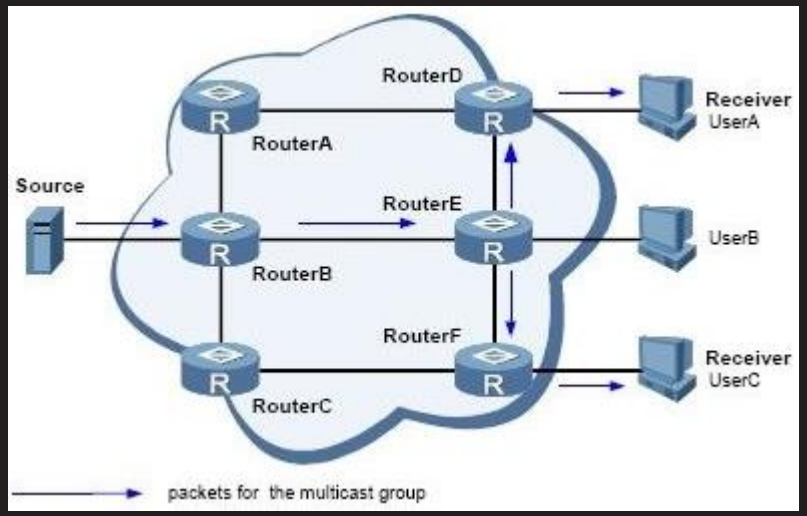
- 广播模式
  - 一台主机向网上的所有其它主机发送数据
  - 无需路径选择，设备简单，维护方便，成本低廉
  - 服务器不用向每个客户机单独发送数据，流量负载极低
  - 无法针对具体客户的具体要求，及时提供个性化的服务
  - 网络无条件地复制和转发每一台主机产生的信息，所有的主机可以收到所有的信息，而不管是否需要，网络资源利用率低，带宽浪费严重
  - 禁止广播包穿越路由器，防止在更大范围内泛滥



通信模式（续2）

Technology Tarena 达内科技

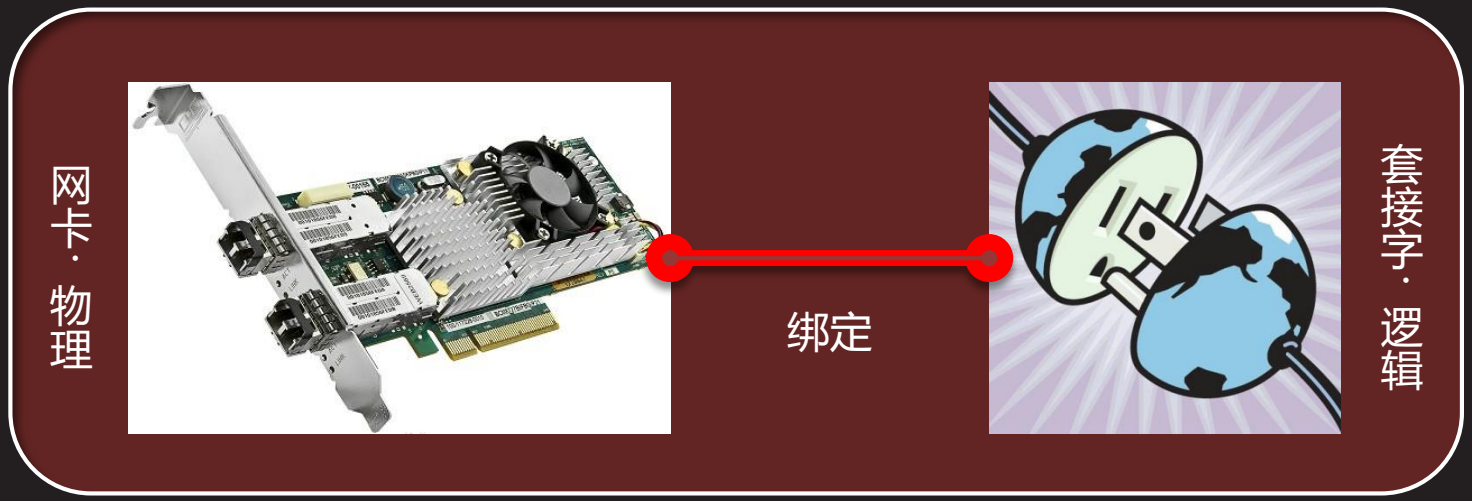
- 多播模式
  - 网络中的主机可以向路由器请求加入或退出某个组，路由器和交换机有选择地复制和转发数据，只将组内数据转发给那些加入组的主机
  - 需要相同信息的客户机只要加入同一个组即可共享同一份数据，降低了服务器和网络的流量负载
  - 既能一次将数据传输给多个有需要的主机，又能保证不影响其它不需要的主机
  - 多播包可以穿越路由器，并在穿越中逐渐衰减
  - 缺乏纠错机制，丢包错包在所难免



绑定与连接

- 如前所述，套接字是一个提供给程序员使用的逻辑对象，它表示对ISO/OSI网络协议模型中传输层及其以下诸层的抽象。但真正发送和接收数据的毕竟是那些实实在在的物理设备。这就需要在物理设备和逻辑对象之间建立一种关联，使后续所有针对这个逻辑对象的操作，最终都能够反映到实际的物理设备上。建立这种关联关系的过程就叫做绑定

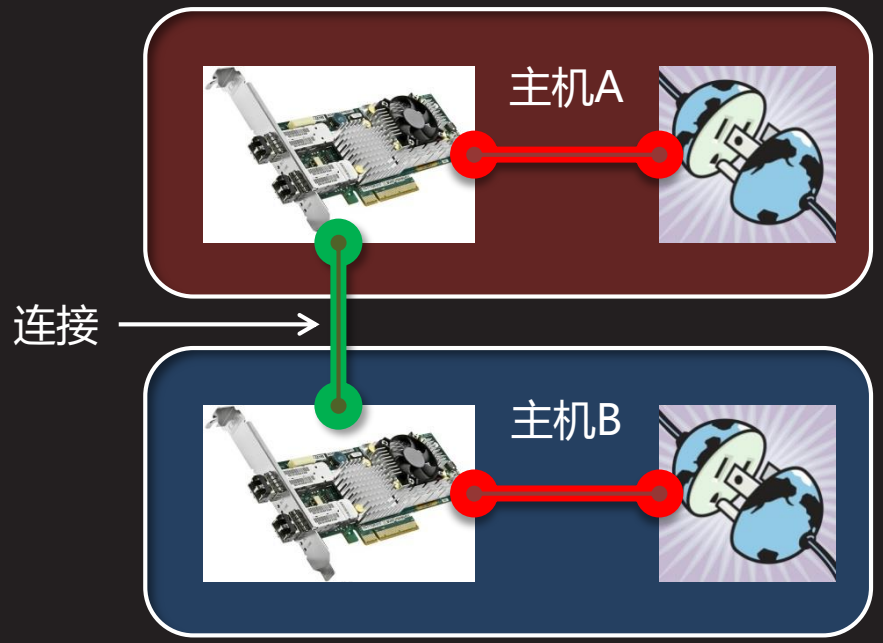
知识讲解



绑定与连接（续1）

Technology Tarena 达内科技

- 绑定只是把套接字对象和一个代表自己的物理设备关联起来。但为了实现通信还需要把自己的物理设备与对方的物理设备关联起来。只有这样才能建立起一种以物理设备为媒介的，跨越不同进程甚至机器的，多个套接字对象之间的联系。建立这种联系的过程就叫做连接



Technology Tarena 达内科技

常用函数

创建套接字

- 创建套接字

```
#include <sys/socket.h>

int socket (int domain, int type, int protocol);
```

成功返回套接字描述符，失败返回-1

– *domain*：通信域，即协议族，可取以下值

`AF_LOCAL/AF_UNIX`	- 本地通信，即进程间通信
`AF_INET`	- 基于IPv4的网络通信
`AF_INET6`	- 基于IPv6的网络通信
`AF_PACKET`	- 基于底层包接口的网络通信

知识讲解

创建套接字（续1）

Technology Tarena 达内科技

- 创建套接字
  - ***type***：套接字类型，可取以下值
    SOCK_STREAM - 流式套接字，即使用TCP协议的套接字
    SOCK_DGRAM - 数据报套接字，即使用UDP协议的套接字
    SOCK_RAW - 原始套接字，即使用IP协议的套接字
  - ***protocol***：特殊协议，通常不用，取0即可
- socket函数所返回的套接字描述符类似于文件描述符，Unix系统把网络也看成是文件，发送数据即写文件，接收数据即读文件，一切皆文件

知识讲解

创建套接字（续2）

Technology Tarena 达内科技

- 例如
  - int sockfd = socket (AF_LOCAL, SOCK_DGRAM, 0);
    if (sockfd == -1) {
        perror ("socket"); exit (EXIT_FAILURE);
    }
  - int sockfd = socket (AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        perror ("socket"); exit (EXIT_FAILURE);
    }
  - int sockfd = socket (AF_INET, SOCK_DGRAM, 0);
    if (sockfd == -1) {
        perror ("socket"); exit (EXIT_FAILURE);
    }

知识讲解

地址结构

Technology Tarena 达内科技

- 套接字接口库通过地址结构定位一个通信主体，可以是一个文件，可以是一台远程主机，也可以是执行者自己
- 基本地址结构，本身没有实际意义，仅用于泛型化参数
  -
```
struct sockaddr {
    sa_family_t sa_family;   // 地址族
    char        sa_data[14]; // 地址值
};
```
- 本地地址结构，用于AF_LOCAL/AF_UNIX域的本地通信
  -
```
#include <sys/un.h>
struct sockaddr_un {
    sa_family_t sun_family; // 地址族(AF_LOCAL)
    char        sun_path[]; // 套接字文件路径
};
```

知识讲解

地址结构（续1）

Technology Tarena 达内科技

- 网络地址结构，用于AF_INET域的IPv4网络通信
  -
```
#include <netinet/in.h>
struct sockaddr_in {
    sa_family_t    sin_family; // 地址族(AF_INET)
    in_port_t      sin_port;   // 端口号
    struct in_addr sin_addr;   // IP地址
};
```
  -
```
struct in_addr {
    in_addr_t s_addr;
};
```
  -
```
typedef uint16_t in_port_t; // 无符号短整型
typedef uint32_t in_addr_t; // 无符号长整型
```

知识讲解

地址结构（续2）

Technology Tarena 达内科技

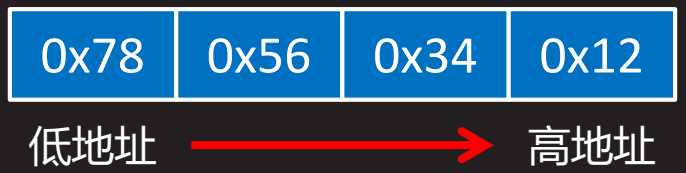
知识讲解

- 如前所述，通过IP地址可以定位网络上的一台主机，但一台主机上可能同时有多个网络应用在运行，究竟想跟哪个网络应用通信呢？这就需要靠所谓的端口号来区分，因为不同的网络应用会使用不同的端口号。用IP地址定位主机，再用端口号定位运行在这台主机上一个具体的网络应用，这样一种对通信主体的描述才是唯一确定的
- 套接字接口库中的端口号被定义为一个16位的无符号整数，其值介于0到65535，其中0到1024已被系统和一些网络服务占据，比如21端口用于ftp服务、23端口用于telnet服务、80端口用于www服务等，因此一般应用程序最好选择1024以上的端口号，以避免和这些服务冲突

地址结构（续3）

Technology Tarena 达内科技

- 网络应用与单机应用不同，经常需要在具有不同硬件架构和操作系统的计算机之间交换数据，因此编程语言里一些多字节数据类型的字节序问题就需要特别予以关注
  - 假设一台小端机器里有一个32位整数：0x12345678，它在内存中按照小端字节序低位低地址的规则存放：



  - 现在，这个整数通过网络被传送到一台大端机器里，内存中的形态不会有丝毫差别，但在大端机器看来地址越低的字节数位应该越高，因此它会把这4个字节解读为：0x78563412，而这显然有悖于发送端的初衷

知识讲解

地址结构（续4）

Technology Tarena 达内科技

- 为了避免字节序带来的麻烦，套接字接口库规定凡是在网络中交换的多字节整数(short、int、long、long long和它们的unsigned版本)一律采用网络字节序传输。而所谓网络字节序，其实就是大端字节序。也就是说，发数据时，先从主机字节序转成网络字节序，然后发送；收数据时，先从网络字节序转成主机字节序，然后使用
  - 小端机A，0x12345678，主机序[0x78,0x56,0x34,0x12]转成网络序[0x12,0x34,0x56,0x78]，发送给B和C
  - 大端机B，接收网络序[0x12,0x34,0x56,0x78]，转成主机序[0x12,0x34,0x56,0x78]，0x12345678
  - 小端机C，接收网络序[0x12,0x34,0x56,0x78]，转成主机序[0x78,0x56,0x34,0x12]，0x12345678

知识讲解

地址结构（续5）

Technology Tarena 达内科技

- 网络地址结构sockaddr_in中表示端口号的sin_port成员和表示IP地址的sin_addr.s_addr成员，分别为2字节和4字节的无符号整数，同样需要用网络字节序来表示
  - struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons (8888);
    addr.sin_addr.s_addr = inet_addr ("192.168.182.48");
  - 其中htons函数将一个16位整数形式的端口号从主机序转成网络序，而inet_addr函数则将一个点分十进制字符串形式的IP地址转成网络序的32位整数形式

知识讲解

将套接字和地址结构绑定

Technology Tarena 达内科技

- 将套接字对象和自己的地址结构绑定在一起

```
#include <sys/socket.h>

int bind (int sockfd, const struct sockaddr* addr,
    socklen_t addrlen);
```

成功返回0，失败返回-1

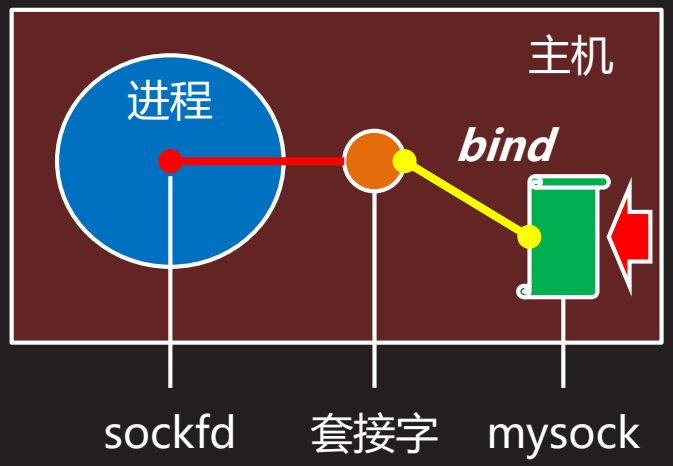
  - ***sockfd***：套接字描述符
  - ***addr***：自己的地址结构
  - ***addrlen***：地址结构长度(以字节为单位)

- 套接字接口库中的很多函数都用到地址结构，但为了同时支持不同的地址结构类型，其参数往往都会选择更一般化的sockaddr类型的指针，使用时需要强制类型转换

将套接字和地址结构绑定（续1）

Technology Tarena 达内科技

- 例如
  - struct sockaddr_un addr;
    addr.sun_family = AF_LOCAL;
    strcpy (addr.sun_path, "mysock");
    if (bind (sockfd, (struct sockaddr*)&addr,
    sizeof (addr)) == -1) {
    perror ("bind"); exit (EXIT_FAILURE); }
- 在上面例子中，将套接字绑定到一个本地文件上，此后所有针对这个本地文件的操作都可以反映到这个套接字之上

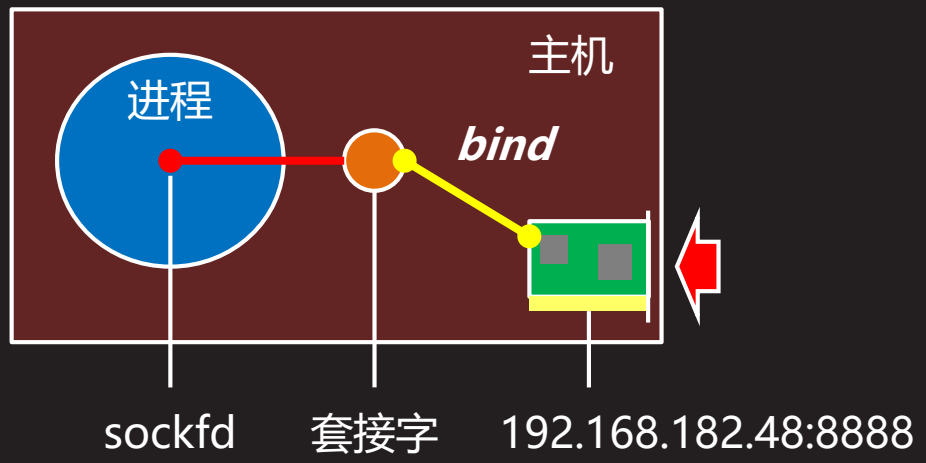


将套接字和地址结构绑定（续2）

Technology Tarena 达内科技

知识讲解

- 例如
  - struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons (8888);
    addr.sin_addr.s_addr = inet_addr ("192.168.182.48");
    if (bind (sockfd, (struct sockaddr*)&addr,
        sizeof (addr)) == -1) {
        perror ("bind"); exit (EXIT_FAILURE); }
- 在上面例子中，将套接字绑定到一个IP和端口上，此后所有针对这个IP和端口的操作都可以反映到这个套接字之上



将套接字和地址结构绑定（续3）

Technology Tarena 达内科技

- 例如
  - struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons (8888);
    addr.sin_addr.s_addr = INADDR_ANY;
    if (bind (sockfd, (struct sockaddr*)&addr,
        sizeof (addr)) == -1) {
        perror ("bind"); exit (EXIT_FAILURE); }
- 在上面的例子中，IP地址使用了INADDR_ANY宏，该宏的值被定义为0，表示任意IP。这样的绑定操作主要用于服务器端，假设服务器主机配置了多个IP地址，无论客户机用哪个IP地址发起通信，服务器套接字都能感觉到

知识讲解

建立连接

Technology Tarena 达内科技

- 将套接字对象和对方的地址结构连接在一起

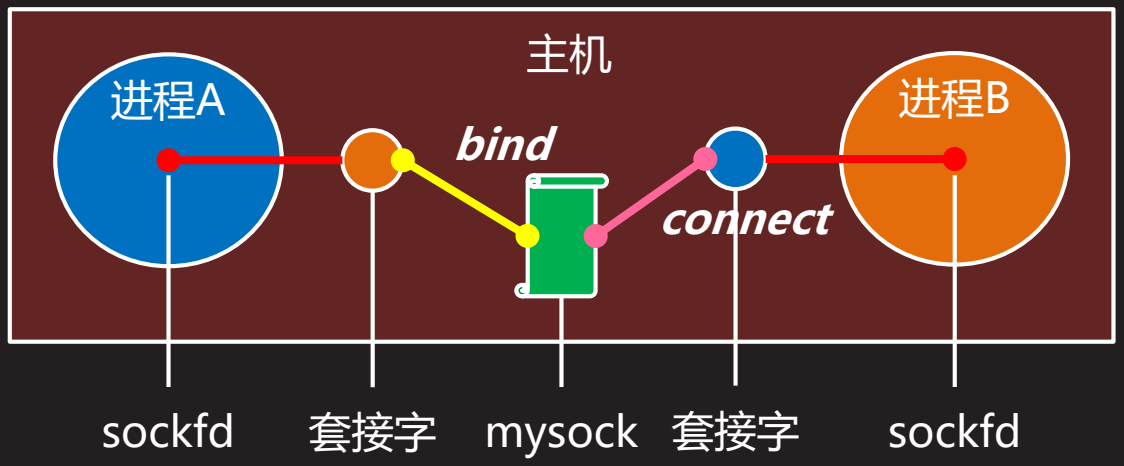
```
#include <sys/socket.h>

int connect (int sockfd, const struct sockaddr* addr,
    socklen_t addrlen);
```

成功返回0，失败返回-1

  - ***sockfd***：套接字描述符
  - ***addr***：对方的地址结构
  - ***addrlen***：地址结构长度(以字节为单位)

建立连接（续1）

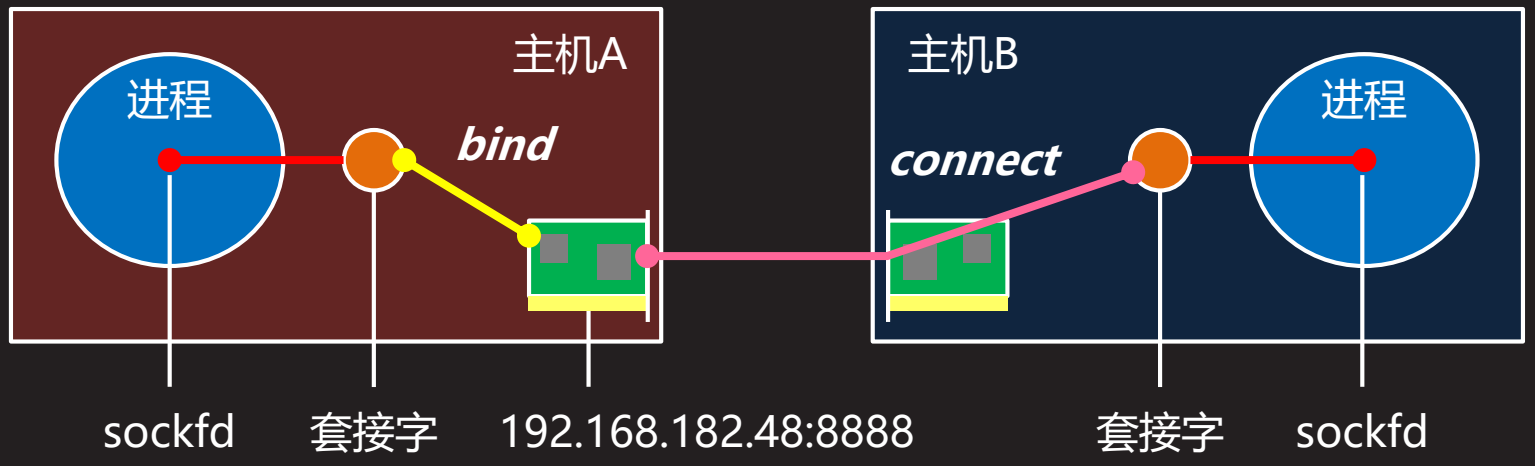
- 例如
  - struct sockaddr_un addr;
    addr.sun_family = AF_LOCAL;
    strcpy (addr.sun_path, "mysock");
    if (connect (sockfd, (struct sockaddr*)&addr,
    sizeof (addr)) == -1) {
    perror ("connect"); exit (EXIT_FAILURE); }



知识讲解

建立连接（续2）

- 例如
  - struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons (8888);
    addr.sin_addr.s_addr = inet_addr ("192.168.182.48");
    if (connect (sockfd, (struct sockaddr*)&addr,
    sizeof (addr)) == -1) {
    perror ("connect"); exit (EXIT_FAILURE); }



知识讲解

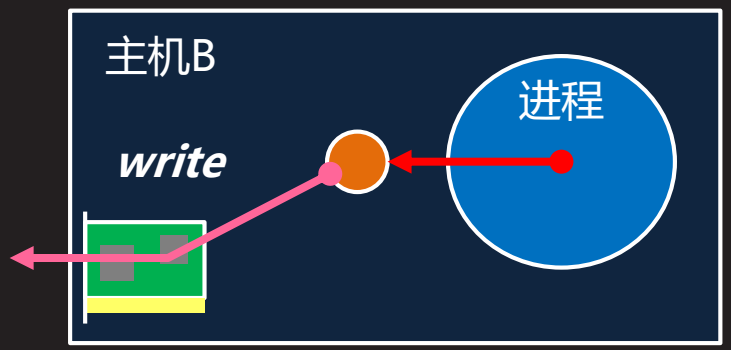
用读写文件的方式通信

Technology Tarena 达内科技

- 通过套接字发送字节流

```
#include <unistd.h>

ssize_t write (int sockfd, const void* buf, size_t count);

        成功返回实际发送的字节数(0表示未发送)，失败返回-1
```

  - *sockfd*：套接字描述符
  - *buf*：内存缓冲区
  - *count*：期望发送的字节数



- 例如
  - ssize_t written = write (sockfd, text, towrite);
    if (written == -1) { perror ("write"); exit (EXIT_FAILURE); }

知识讲解

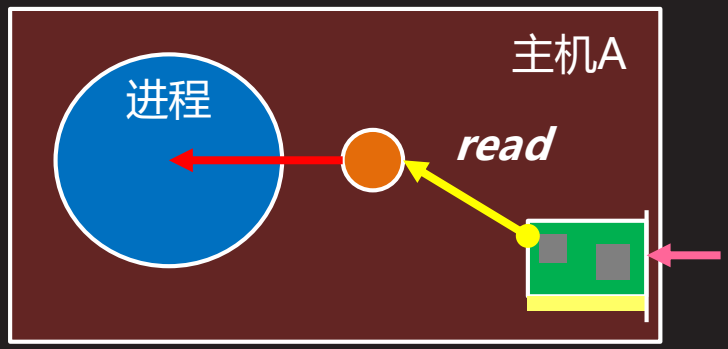
用读写文件的方式通信（续1）

Technology Tarena 达内科技

- 通过套接字接收字节流

```
#include <unistd.h>

ssize_t read (int sockfd, void* buf, size_t count);

    成功返回实际接收的字节数(0表示连接已关闭)，失败返回-1
```

  - ***sockfd***：套接字描述符
  - ***buf***：内存缓冲区
  - ***count***：期望读取的字节数



- 例如
  - ssize_t readed = read (sockfd, text, toread);
    if (readed == -1) { perror ("read"); exit (EXIT_FAILURE); }

知识讲解

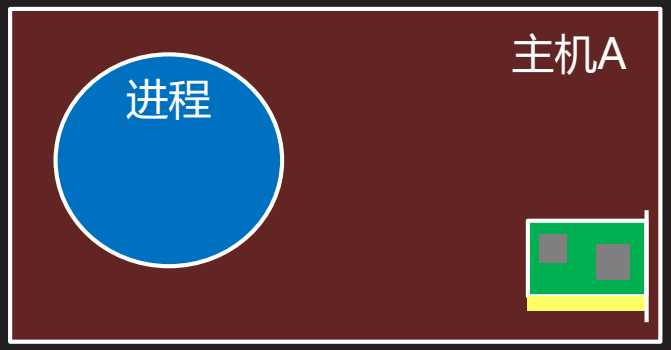
关闭套接字

- 关闭处于打开状态的套接字描述符

```
#include <unistd.h>

int close (int sockfd);
```

成功返回0，失败返回-1

  - *sockfd*：套接字描述符



- 例如
  - if (close (sockfd) == -1) {
    perror ("close");
    exit (EXIT_FAILURE);
    }

知识讲解

字节序转换

- 将主机或网络字节序的长短整数转换为网络或主机字节序

```
#include <arpa/inet.h>

uint32_t htonl (uint32_t hostlong);
uint16_t htons (uint16_t hostshort);
uint32_t ntohl (uint32_t netlong);
uint16_t ntohs (uint16_t netshort);
```

h：主机字节序
n：网络字节序
l：长整数(32位)
s：短整数(16位)

返回网络或主机字节序的长短整数

- ***hostlong***：主机字节序长整数
- ***hostshort***：主机字节序短整数
- ***netlong***：网络字节序长整数
- ***netshort***：网络字节序短整数

知识讲解

IP地址转换

Technology Tarena 达内科技

- 点分十进制字符串->网络字节序32位无符号整数

```
#include <arpa/inet.h>

in_addr_t inet_addr (const char* cp);

                    返回网络字节序32位无符号整数形式的IP地址
```

  - *cp*：点分十进制字符串形式的IP地址
- 例如
  - struct sockaddr_in addr;
    addr.sin_addr.s_addr = inet_addr ("192.168.182.48");

知识讲解

IP地址转换（续1）

Technology Tarena 达内科技

- 点分十进制字符串->网络字节序32位无符号整数

```
#include <arpa/inet.h>

int inet_aton (const char* cp, struct in_addr* inp);

                                成功返回0，失败返回-1
```

  - *cp*：点分十进制字符串形式的IP地址
  - *inp*：输出包含网络字节序32位无符号整数形式IP地址的in_addr结构

- 例如
  - struct sockaddr_in addr;
    if (inet_aton ("192.168.182.48", &addr.sin_addr) == -1) {
        perror ("inet_aton"); exit (EXIT_FAILURE); }

知识讲解

IP地址转换（续2）

Technology Tarena 达内科技

- 网络字节序32位无符号整数->点分十进制字符串

```
#include <arpa/inet.h>

char* inet_ntoa (struct in_addr in);

                        返回点分十进制字符串形式的IP地址
```

  - *in*：包含网络字节序32位无符号整数形式IP地址的in_addr结构
- 例如
  - struct sockaddr_in addr;
    inet_aton ("192.168.182.48", &addr.sin_addr);
    printf ("%s\n", inet_ntoa (addr.sin_addr));

知识讲解

Technology Tarena 达内科技

通信编程

进程间通信

Technology Tarena 达内科技

- 基于套接字实现进程间通信的编程模型

步骤	服务器		客户机		步骤
1	创建套接字	socket	socket	创建套接字	1
2	准备地址结构	sockaddr_un	sockaddr_un	准备地址结构	2
3	绑定地址	bind	connect	建立连接	3
4	接收请求	read	write	发送请求	4
5	发送响应	write	read	接收响应	5
6	关闭套接字	close	close	关闭套接字	6

知识讲解

- 创建套接字时使用AF_LOCAL域
  - int sockfd = socket (AF_LOCAL, ...);

进程间通信（续1）

Technology Tarena 达内科技

- 准备地址结构时使用sockaddr_un结构体类型
  - struct sockaddr_un addr;
    addr.sun_family = AF_LOCAL;
    strcpy (addr.sun_path, "mysock");
- 套接字文件如不再用需要显式删除
  - unlink ("mysock")

知识讲解

基于套接字的进程间通信

【参见：locsvr.c、loccli.c】

课堂练习

- 基于套接字的进程间通信

网络通信

Technology Tarena 达内科技

- 基于套接字实现网络通信的编程模型

步骤	服务器		客户机		步骤
1	创建套接字	socket	socket	创建套接字	1
2	准备地址结构	sockaddr_in	sockaddr_in	准备地址结构	2
3	绑定地址	bind	connect	建立连接	3
4	接收请求	read	write	发送请求	4
5	发送响应	write	read	接收响应	5
6	关闭套接字	close	close	关闭套接字	6

- 创建套接字时使用AF_INET域
  - int sockfd = socket (AF_INET, ...);

知识讲解

网络通信（续1）

Technology Tarena 达内科技

- 准备地址结构时使用sockaddr_in结构体类型
  - 服务器

```
struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_port = htons (8888);
addr.sin_addr.s_addr = INADDR_ANY;
```

  - 客户机

```
struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_port = htons (8888);
addr.sin_addr.s_addr = inet_addr ("192.168.182.48");
```

  - 客户机连本机服务器可以使用本地环回地址

```
addr.sin_addr.s_addr = inet_addr ("127.0.0.1");
```

知识讲解

Technology Tarena 达内科技

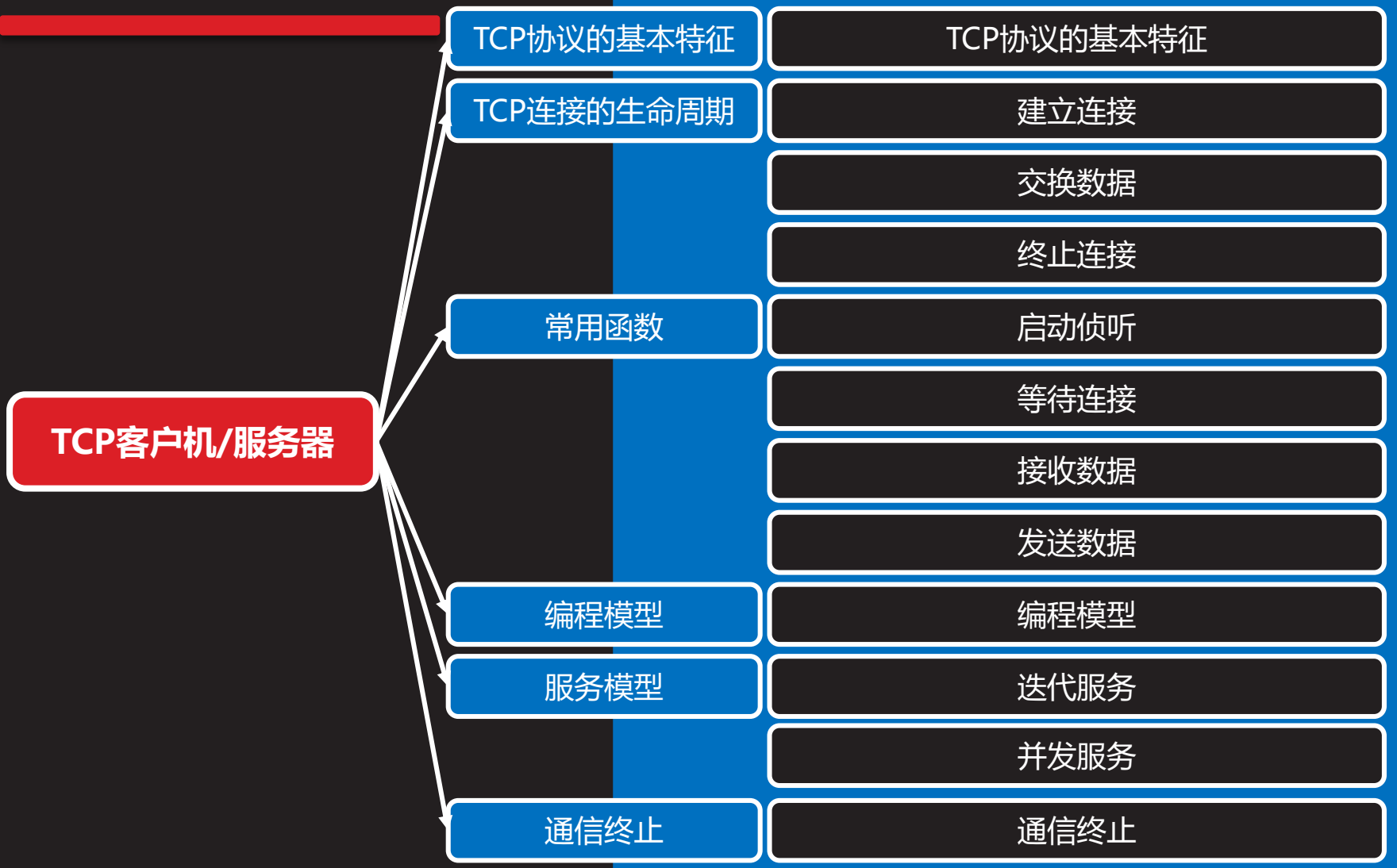
基于套接字的网络通信

【参见：netsvr.c、netcli.c】

课堂练习

- 基于套接字的网络通信

TCP客户机/服务器


TCP客户机/服务器
TCP协议的基本特征
TCP连接的生命周期
常用函数
编程模型
服务模型
通信终止
TCP协议的基本特征
建立连接
交换数据
终止连接
启动侦听
等待连接
接收数据
发送数据
编程模型
迭代服务
并发服务
通信终止

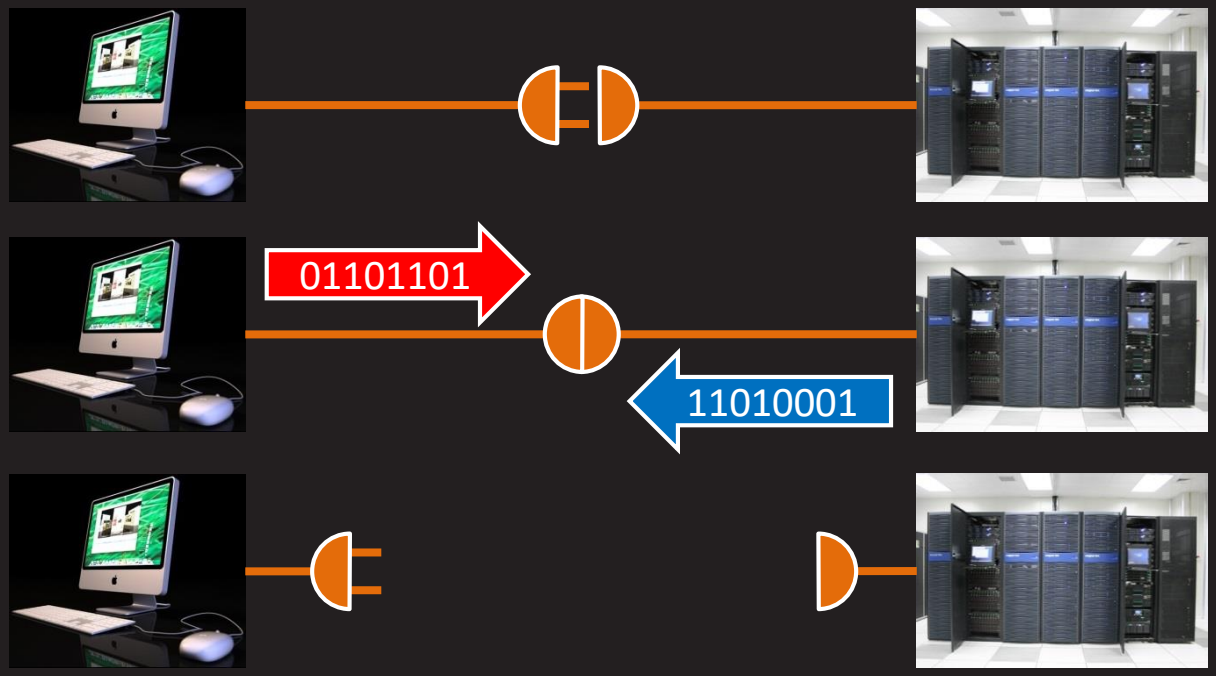
Technology
Tarena
达内科技

TCP协议的基本特征

TCP协议的基本特征

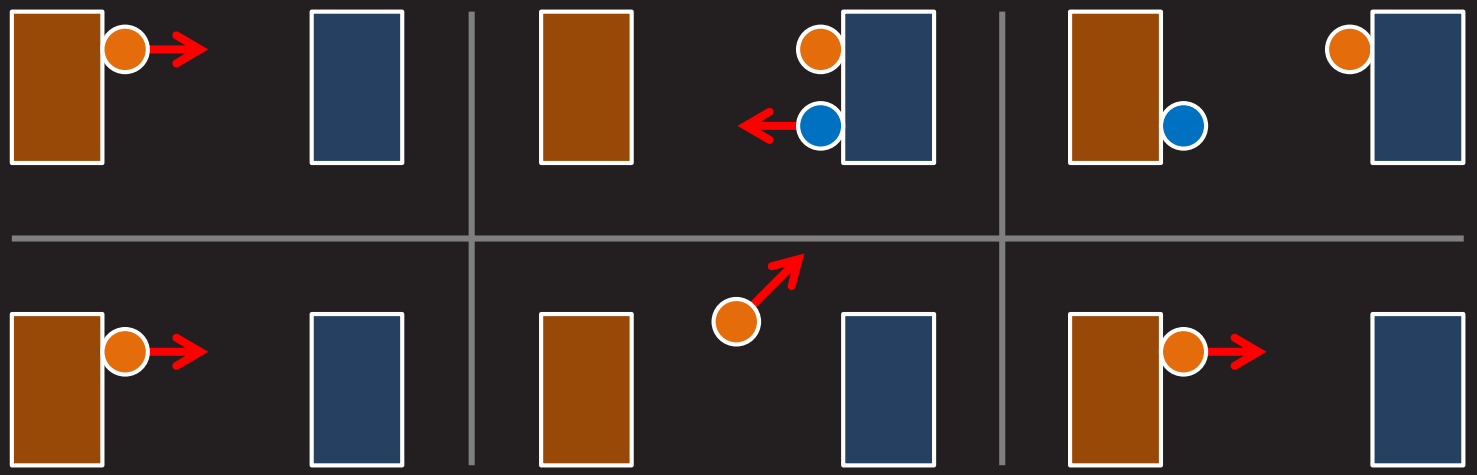
- TCP提供客户机与服务器的连接
  - 一个完整TCP通信过程需要依次经历三个阶段
    - 首先，客户机必须建立与服务器的连接，所谓虚电路
    - 然后，凭借已建立好的连接，通信双方相互交换数据
    - 最后，客户机与服务器双双终止连接，结束通信过程

知识讲解



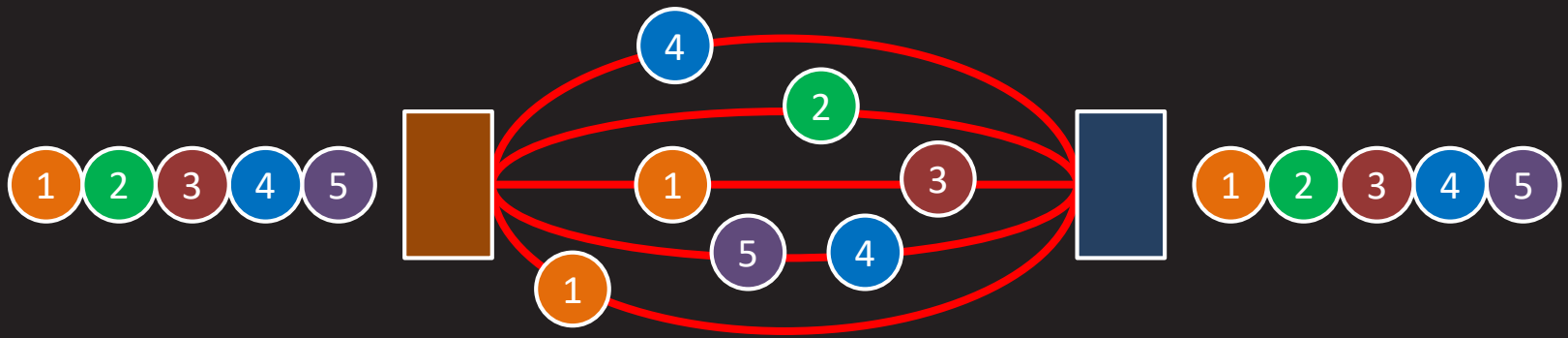
TCP协议的基本特征（续1）

- TCP保证数据传输的可靠性
  - TCP的协议栈底层在向另一端发送数据时，会要求对方在一个给定的时间窗口内返回确认。如果超过了这个时间窗口仍没有收到确认，则TCP会重传数据并等待更长的时间。只有在数次重传均告失败以后，TCP才会最终放弃。TCP含有用于动态估算数据往返时间(Round-Trip Time, RTT)的算法，因此它知道等待一个确认需要多长时间



TCP协议的基本特征（续2）

- TCP保证数据传输的有序性
  - TCP的协议栈底层在向另一端发送数据时，会为所发送数据的每个字节指定一个序列号。即使这些数据字节没有能够按照发送时的顺序到达接收方，接收方的TCP也可以根据它们的序列号重新排序，再把最后的结果交给应用程序
  - 如果TCP收到重复的数据(比如发送方认为数据已丢失并重传，但它可能并没有真的丢失，而只是由于网络拥塞而被延误)，它也可以根据序列号做出判断，丢弃重复的数据



知识讲解

TCP协议的基本特征（续3）

- TCP提供流量控制
  - TCP的协议栈底层在从另一端接收数据时，会不断告知对方它能够接收多少字节的数据，即所谓通告窗口。任何时候，这个窗口都反映了接收缓冲区可用空间的大小，从而确保不会因为发送方发送数据过快而导致接收缓冲区溢出

知识讲解

TCP协议的基本特征（续4）

Technology Tarena 达内科技

知识讲解

- TCP是流式传输协议
  - TCP是一个字节流协议，无记录边界

  - 应用程序如果需要确定记录边界，必须自己实现
    - 定长记录

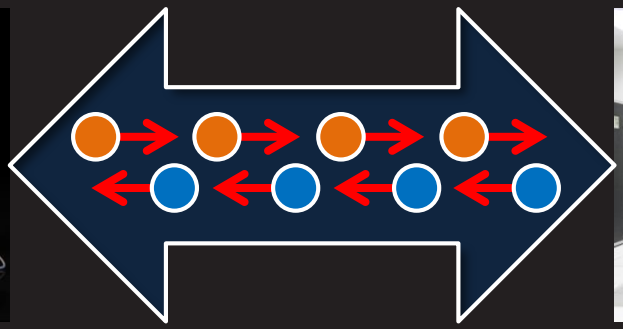
    - 不定长记录+分隔符

    - 长度+不定长记录

TCP协议的基本特征（续5）

- TCP是全双工的
  - 在给定的连接上，应用程序在任何时候都既可以发送数据也可以接收数据。因此，TCP必须跟踪每个方向上数据流的状态信息，如序列号和通告窗口的大小

知识讲解



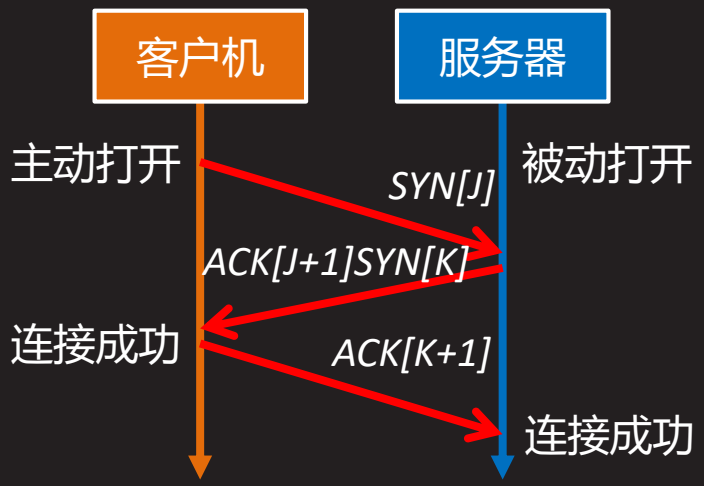
Technology Tarena 达内科技

TCP连接的生命周期

建立连接

Technology Tarena 达内科技

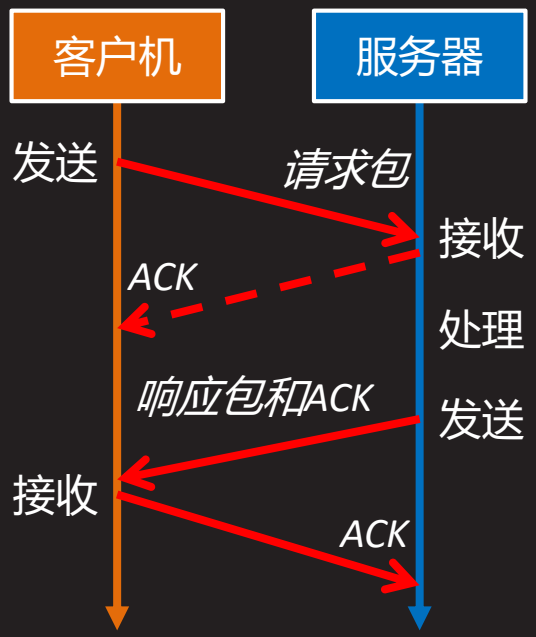
- 被动打开
  - 服务器必须首先做好准备随时接受来自客户机的连接请求
- 三路握手
  - 客户机的TCP协议栈向服务器发送一个SYN分节，告知对方自己将在连接中发送数据的初始序列号，谓之主动打开
  - 服务器的TCP协议栈向客户机发送一个单个分节，其中不仅包括对客户机SYN分节的ACK应答，还包含服务器自己的SYN分节，以告知对方自己在同一连接中发送数据的初始序列号
  - 客户机的TCP协议栈向服务返回ACK应答，以表示对服务器所发SYN的确认



知识讲解

交换数据

- 一旦连接建立，客户机即可构造请求包并发往服务器
- 服务器接收并处理来自客户机的请求包，构造响应包
- 服务器向客户机发送响应包，同时捎带对客户机请求包的ACK应答。但如果服务器处理请求和构造响应的时间长于200毫秒，则应答也可能先于响应发出
- 客户机接收来自服务器的响应包，同时向对方发送ACK应答

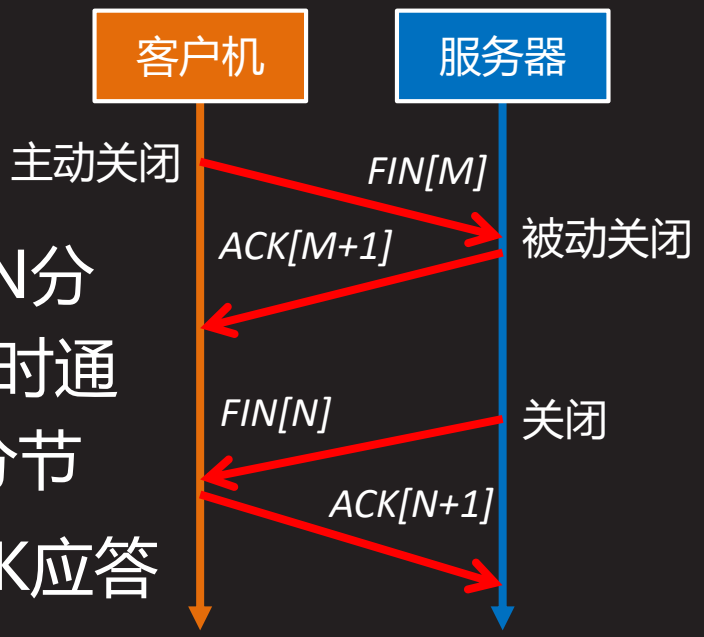


知识讲解

终止连接

Technology Tarena 达内科技

- 客户机或者服务器主动关闭连接，TCP协议栈向对方发送FIN分节，表示数据通信结束。如果此时尚有数据滞留于发送缓冲区中，则FIN分节跟在所有未发送数据之后
- 接收到FIN分节的另一端执行被动关闭，一方面通过TCP协议栈向对方发送ACK应答，另一方面向应用程序传递文件结束符。如果此时接收缓冲区不空，则将所接收到的FIN分节追加到接收缓冲区的末尾
- 一段时间以后，方才接收到FIN分节的进程关闭自己的连接，同时通过TCP协议栈向对方发送FIN分节
- 对方在收到FIN分节后发送ACK应答



Technology Tarena 达内科技

常用函数

启动侦听

- 在指定套接字上启动对连接请求的侦听

```
#include <sys/socket.h>

int listen (int sockfd, int backlog);

                                成功返回0，失败返回-1
```

  - ***sockfd***：套接字描述符
  - ***backlog***：未决连接请求的最大值
- socket函数所创建的套接字一律被初始化为主动套接字，即可以通过后续connect函数调用向服务器发起连接请求的客户机套接字。listen函数可以将一个这样的主动套接字转换为被动套接字，即可以等待并接受来自客户机的连接请求的服务器套接字

知识讲解

启动侦听（续1）

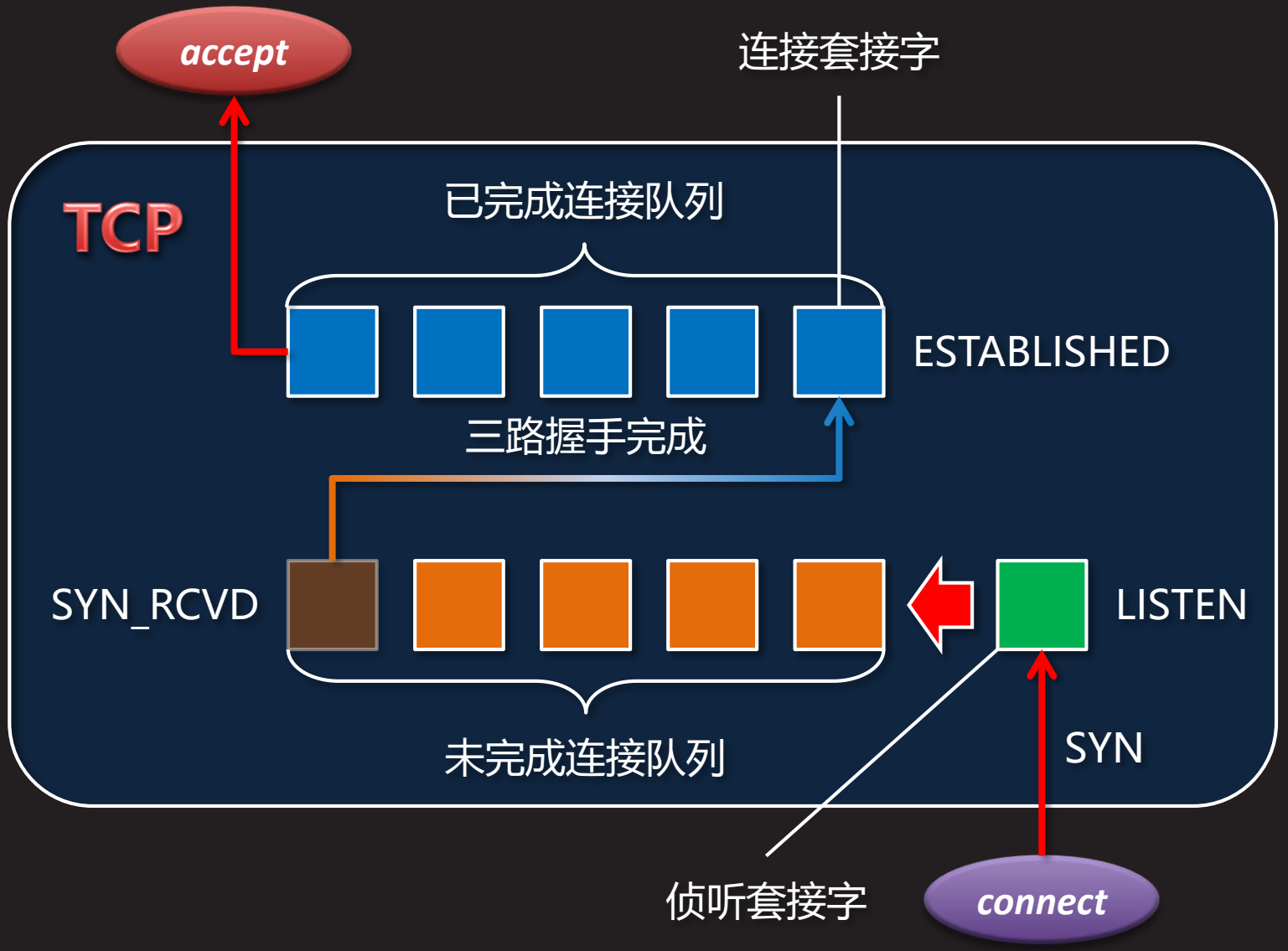
Technology Tarena 达内科技

- 被listen函数启动侦听的套接字将由CLOSED状态转入LISTEN状态
- 客户机调用connect函数即开启了TCP连接建立的第一路握手：通过协议栈向服务器发送SYN分节。服务器的LISTEN套接字一旦收到该分节，即创建一个新的处于SYN_RCVD状态的套接字，并将其排入未完成连接队列
- 服务器的TCP协议栈不断监视未完成连接队列的状态，并在适当的时机依次处理其中等待连接的套接字。一旦某个套接字上的第二、三路握手完成，由SYN_RCVD状态转入ESTABLISHED状态，即被移送到已完成连接队列
- 两个队列中的套接字个数之和不能超过***backlog***参数值

知识讲解

启动侦听（续2）

Technology Tarena 达内科技



启动侦听（续3）

Technology Tarena 达内科技

- 若未完成连接队列和已完成连接队列中的套接字个数之和已经达到***backlog***，此时又有客户机通过connect函数发起连接请求，则该请求所产生的SYN分节将被服务器的TCP协议栈直接忽略。客户机的TCP协议栈会因第一路握手应答超时而重发SYN分节，期望不久能在未决队列中找到空闲位置。若多次重发均告失败，则客户机放弃，connect函数返回失败
- 客户机对connect函数的调用在第二路握手完成时即返回，而此时服务器连接套接字可能还在未完成连接队列(第三路握手尚未完成)或已完成连接对列(套接字尚未返回给用户进程)中。这种情况下客户机发送的数据，会被服务器的TCP协议栈排队缓存，直到接收缓冲区满为止

知识讲解

启动侦听（续4）

Technology Tarena 达内科技

- 例如
  - int listenfd = socket (AF_INET, SOCK_STREAM, 0);
    if (listenfd == -1) {
        perror ("socket"); exit (EXIT_FAILURE); }
    struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons (8888);
    addr.sin_addr.s_addr = INADDR_ANY;
    if (bind (listenfd, (struct sockaddr*)&addr,
        sizeof (addr)) == -1) {
        perror ("bind"); exit (EXIT_FAILURE); }
    if (listen (listenfd, 1024) == -1) {
        perror ("listen"); exit (EXIT_FAILURE); }

等待连接

Technology Tarena 达内科技

- 在指定套接字上等待并接受连接请求

```
#include <sys/socket.h>

int accept (int sockfd, struct sockaddr* addr,
    socklen_t* addrlen);
```

成功返回连接套接字描述符，失败返回-1

  - ***sockfd***：侦听套接字描述符
  - ***addr***：输出连接请求发起者地址结构
  - ***addrlen***：输入/输出，连接请求发起者地址结构长度(以字节为单位)

- accept函数由TCP服务器调用，返回排在已完成连接队列首部的连接套接字对象的描述符，若队列为空则阻塞

等待连接（续1）

Technology Tarena 达内科技

知识讲解

- 若accept函数执行成功，则通过***addr***和***addrlen***向调用者输出发起连接请求的客户机的协议地址及其字节长度。注意***addrlen***既是输入参数也是输出参数。调用accept函数时，指针***addrlen***所指向的变量被初始化为***addr***结构体的字节大小；等到该函数返回时，该指针的目标则被更新为系统内核保存在***addr***结构体内的实际字节数
- accept函数成功返回的是一个有别于其参数套接字，由系统内核自动生成的全新套接字描述符。它代表与客户机的TCP连接，因此被称为连接套接字，而该函数的第一个参数则被称为侦听套接字。通常一个服务器只有一个侦听套接字，且一直存在直到服务器关闭，而连接套接字则是一个客户机一个，专门负责与该客户机的通信

等待连接（续2）

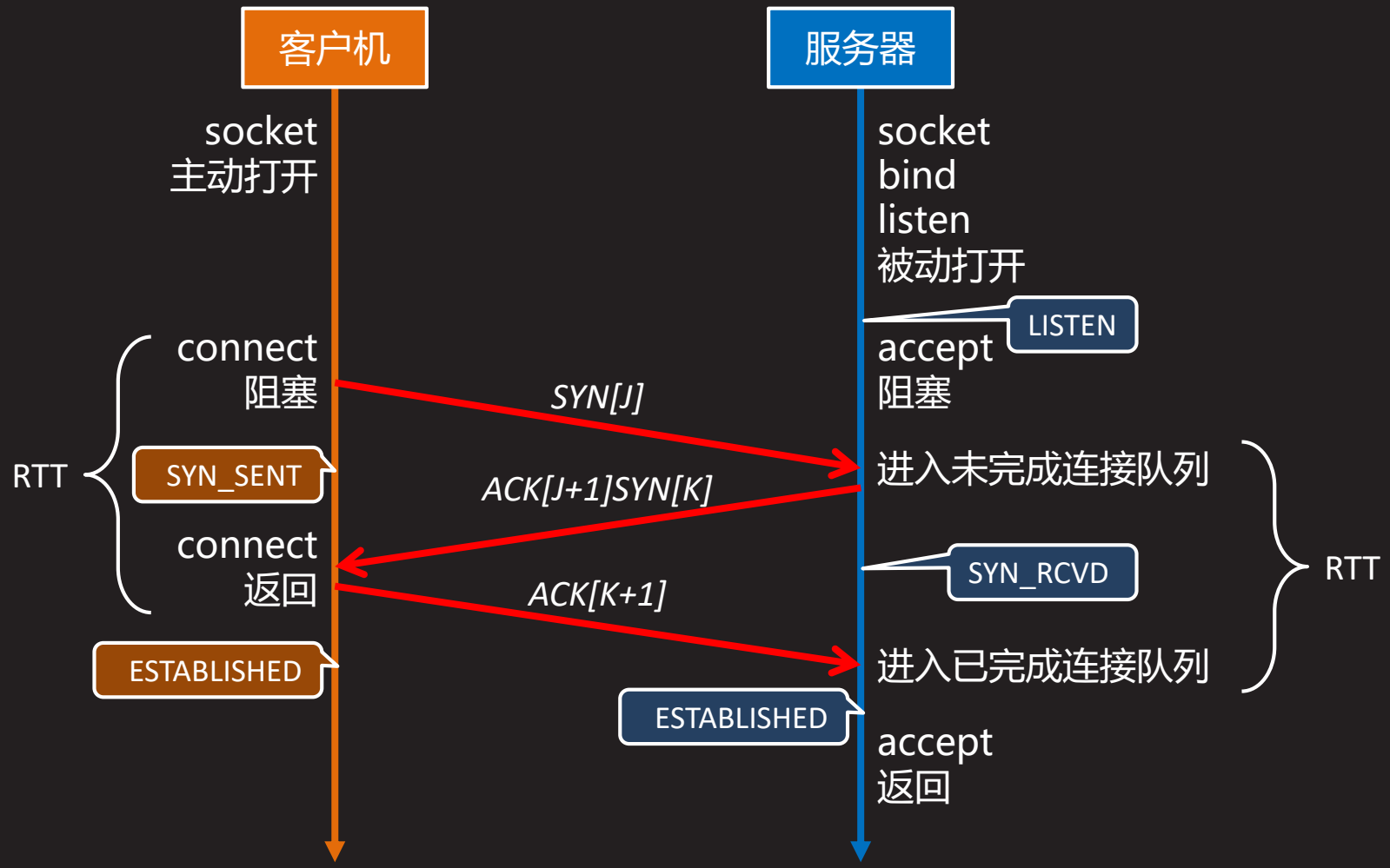
Technology Tarena 达内科技

- 例如
  - struct sockaddr_in addrcli = {};
    socklen_t addrlen = sizeof (addrcli);
    int connfd = accept (listenfd,
    (struct sockaddr*)&addrcli, &addrlen);
    if (connfd == -1) {
    perror ("accept"); exit (EXIT_FAILURE); }
    printf ("服务器已接受来自%s:%hu客户机的连接请求\n",
    inet_ntoa (addrcli.sin_addr),
    ntohs (addrcli.sin_port));

知识讲解

等待连接（续3）

- connect、accept与TCP三路握手



知识讲解

接收数据

- 通过指定套接字接收数据

```
#include <sys/socket.h>

ssize_t recv (int sockfd, void* buf, size_t len, int flags);
```

成功返回实际接收到的字节数，失败返回-1

  - *sockfd*：套接字描述符
  - *buf*：应用程序接收缓冲区
  - *len*：期望接收的字节数

知识讲解

接收数据（续1）

Technology Tarena 达内科技

- 通过指定套接字接收数据
  - ***flags***：接收标志，一般取0，还可取以下值

    MSG_DONTWAIT - 以非阻塞方式接受数据

    MSG_OOB - 接收带外数据

    MSG_PEEK - 只查看可接收的数据，函数返回后数据依然留在接收缓冲区中

    MSG_WAITALL - 等待所有数据，即不接收到***len***字节的数据，函数就不返回

- 如前所述，客户机或者服务器主动关闭连接，TCP协议栈向对方发送FIN分节，表示数据通信结束，接收到FIN分节的另一端执行被动关闭，一方面通过TCP协议栈向对方发送ACK应答，另一方面向应用程序传递文件结束符，此时recv函数返回0

接收数据（续2）

Technology Tarena 达内科技

- 阻塞与非阻塞
  - 套接字I/O的缺省方式都是阻塞的。对于TCP而言，如果接收缓冲区中没有数据，recv函数将会阻塞，直到有数据到来并被复制到***buf***缓冲区时才会返回。此时所接收到的数据可能比***len***参数期望接收的字节数少。除非调用recv函数时使用MSG_WAITALL标志，不接收到***len***字节的数据，函数就不返回。但即便使用了MSG_WAITALL标志，实际接收到的字节数在以下三种情况下仍然可能比期望的少
    - 函数被信号中断
    - 连接被对方终止
    - 发生套接字错误
  - MSG_DONTWAIT标志令接收过程以非阻塞方式进行，即便收不到数据，recv函数也会立即返回，返回值为-1，errno为EAGAIN或EWOULDBLOCK

知识讲解

接收数据（续3）

Technology Tarena 达内科技

- 例如
  - char buf[1024];
    ssize_t rcvd = recv (connfd, buf, sizeof (buf), 0);
    if (rcvd == -1) {
        perror ("recv");
        exit (EXIT_FAILURE);
    }
    if (rcvd == 0) {
        printf ("客户机已关闭连接\n");
        exit (EXIT_SUCCESS);
    }
    buf[rcvd] = '\0';
    printf ("%s\n", buf);

知识讲解

发送数据

Technology Tarena 达内科技

- 通过指定套接字发送数据

```
#include <sys/socket.h>

ssize_t send (int sockfd, const void* buf, size_t len,
    int flags);
```

成功返回实际被发送的字节数，失败返回-1

  - *sockfd*：套接字描述符
  - *buf*：应用程序发送缓冲区
  - *len*：期望发送的字节数

知识讲解

发送数据（续1）

Technology Tarena 达内科技

- 通过指定套接字发送数据
  - *flags*：发送标志，一般取0，还可取以下值

    `MSG_DONTWAIT` - 以非阻塞方式发送数据

    `MSG_OOB` - 发送带外数据

    `MSG_DONTROUTE` - 不查路由表，直接在本地网络中寻找目的主机

知识讲解

发送数据（续2）

Technology Tarena 达内科技

- 阻塞与非阻塞
  - 套接字I/O的缺省方式都是阻塞的。对于TCP而言，如果发送缓冲区中没有足够的空闲空间，send函数将会阻塞，直到其空闲空间足以容纳***len***字节的待发送数据，并在将全部待发送数据复制到发送缓冲区后才会返回
  - MSG_DONTWAIT标志令发送过程以非阻塞方式进行，即便发送缓冲区中一个字节的空闲空间都没有，send函数也会立即返回，返回值为-1，errno为EAGAIN或EWOULDBLOCK
  - 在非阻塞方式下，如果发送缓冲区中尚有少量空闲空间，则会将部分待发送数据复制到发送缓冲区，同时返回复制到发送缓冲区中的字节数

知识讲解

发送数据（续3）

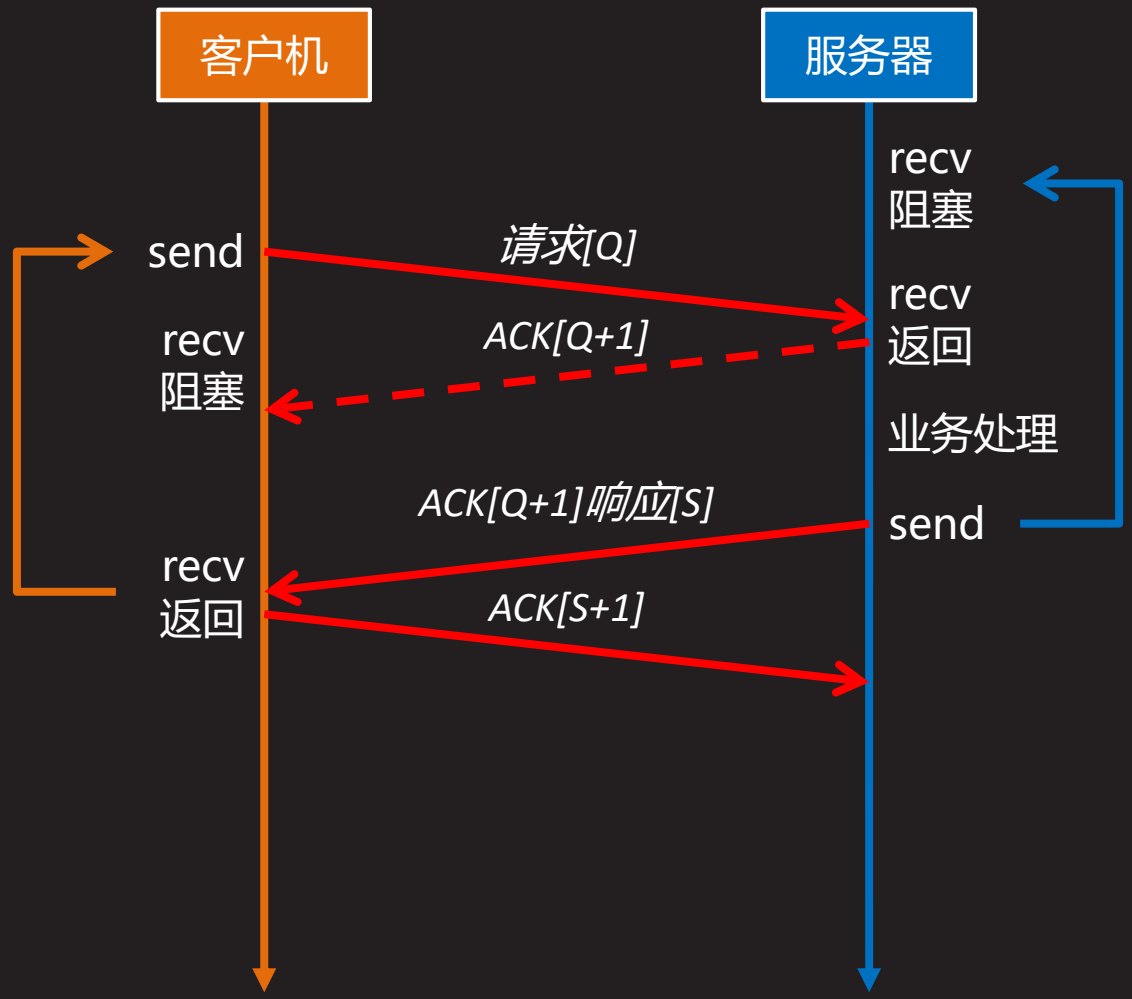
Technology Tarena 达内科技

- 例如
  - char buf[1024];
    gets (buf);
    ssize_t sent = send (connfd, buf, strlen (buf) *
        sizeof (buf[0]), 0);
    if (sent == -1) {
        perror ("send");
        exit (EXIT_FAILURE);
    }

知识讲解

发送数据（续4）

- send与recv



知识讲解

Technology Tarena 达内科技

编程模型

编程模型

- 基于TCP协议实现网络通信的编程模型

步骤	服务器		客户机		步骤
1	创建套接字	socket	socket	创建套接字	1
2	准备地址结构	sockaddr_in	sockaddr_in	准备地址结构	2
3	绑定地址	bind	——	——	——
4	启动侦听	listen	——	——	——
5	等待连接	accept	connect	请求连接	3
6	接收请求	recv	send	发送请求	4
7	发送响应	send	recv	接收响应	5
8	关闭套接字	close	close	关闭套接字	6

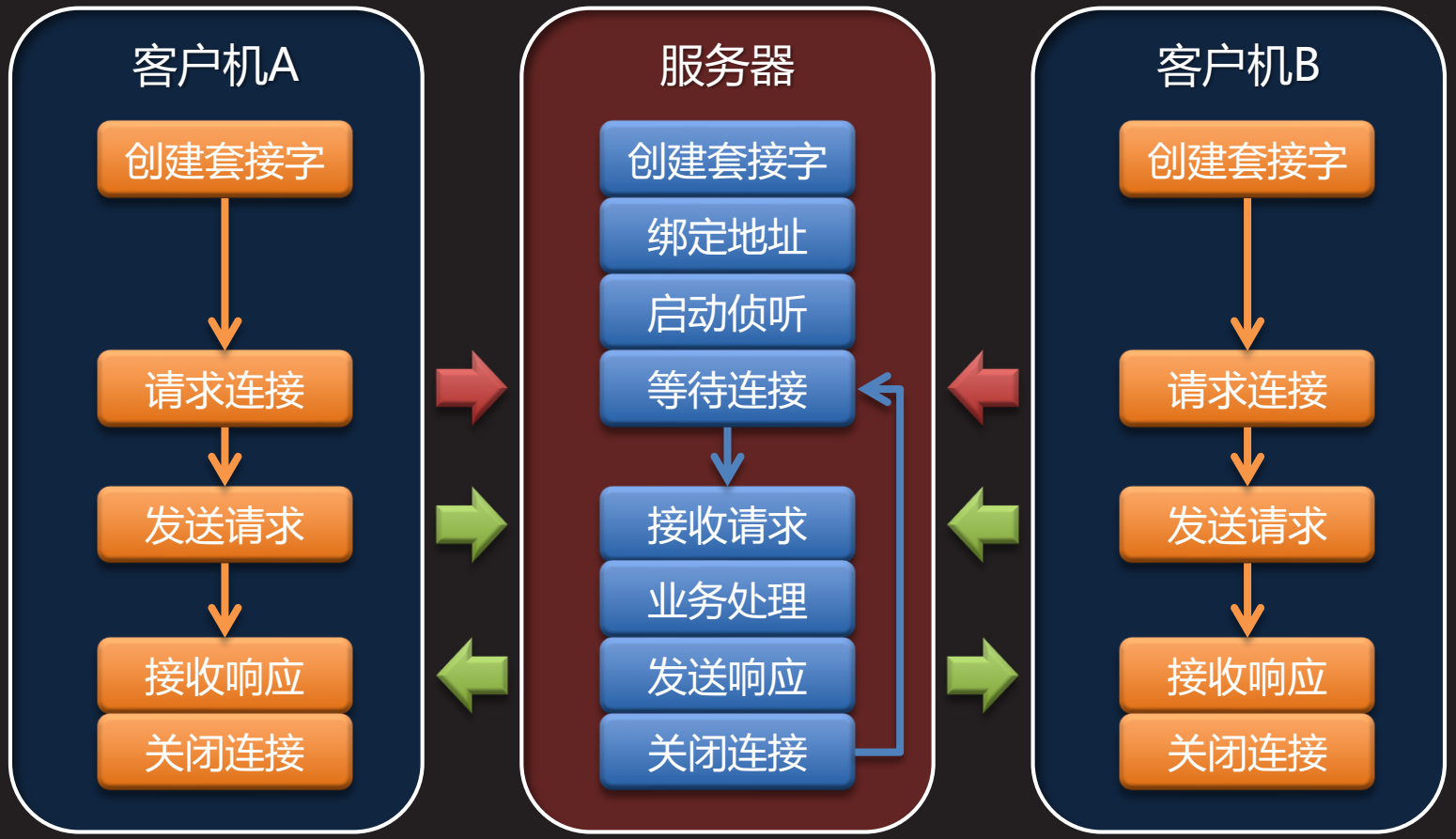
知识讲解

Technology Tarena 达内科技

服务模型

迭代服务

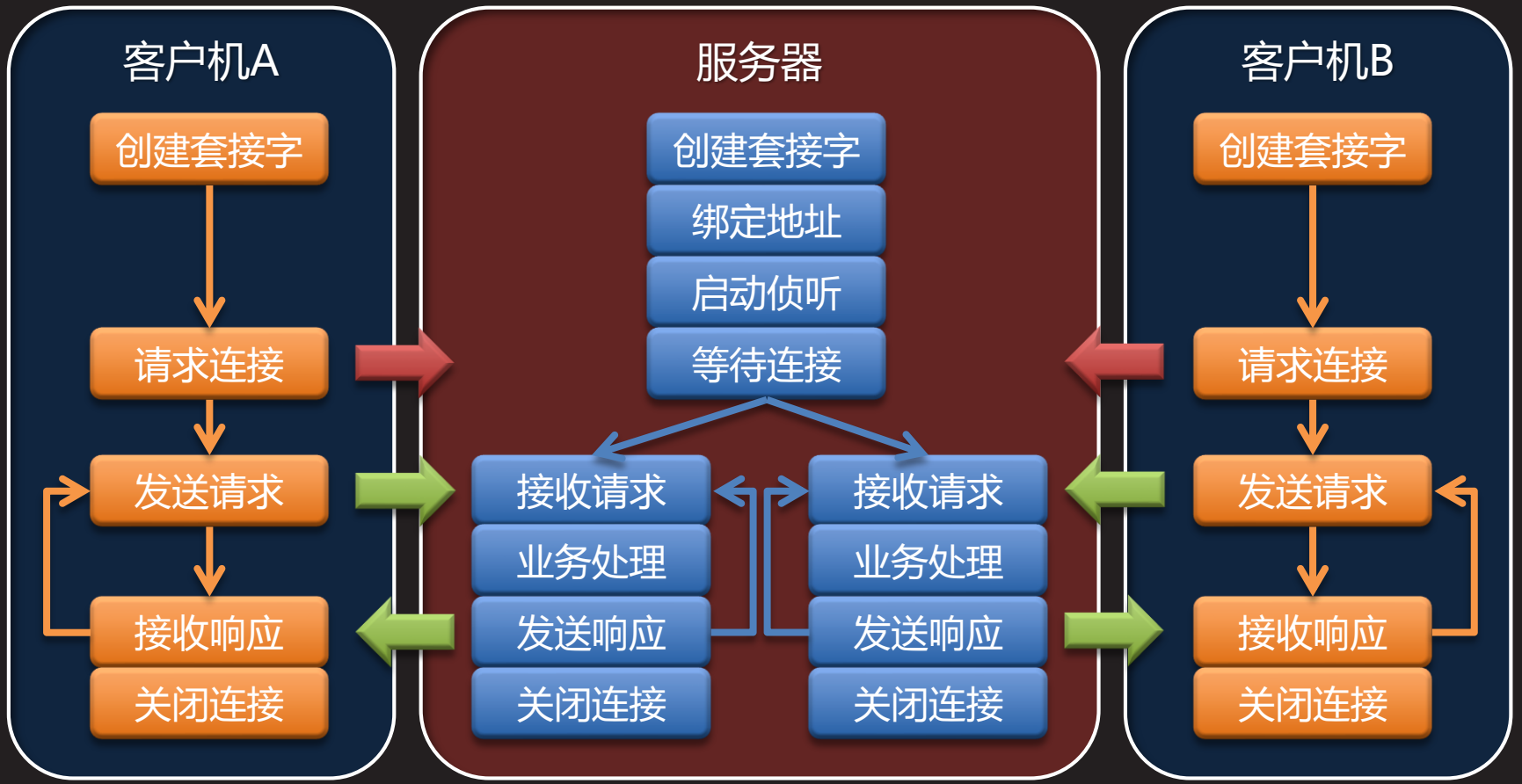
- 服务器在单线程中以循环迭代的方式依次处理每个客户机的业务需求。迭代模型的前提是针对每个客户机的处理时间必须足够短暂，否则会延误对其它客户机的响应



知识讲解

并发服务

- 主进程阻塞在accept函数上。每当一个客户机与服务器建立连接，accept函数返回，即通过fork函数创建子进程，主进程继续等待新的连接，子进程处理客户机业务



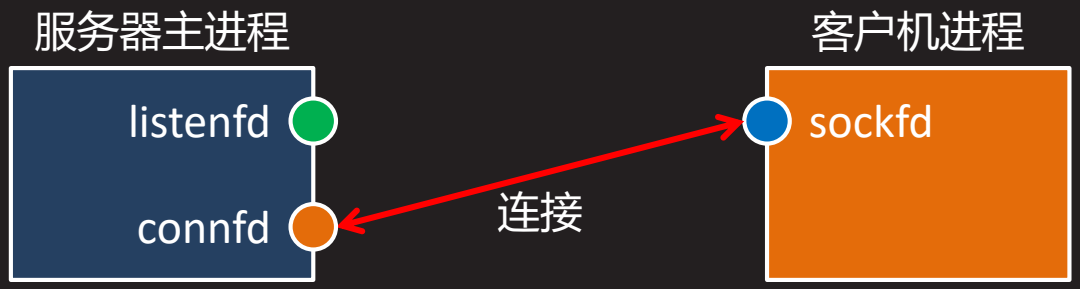
知识讲解

并发服务（续1）

Technology Tarena 达内科技

- 首先服务器主进程阻塞于针对侦听套接字的accept调用，客户机进程通过connect函数向服务器发起连接请求

- 客户机的连接请求被系统内核接受，服务器主进程从accept函数中返回，同时得到可用于通通信的连接套接字

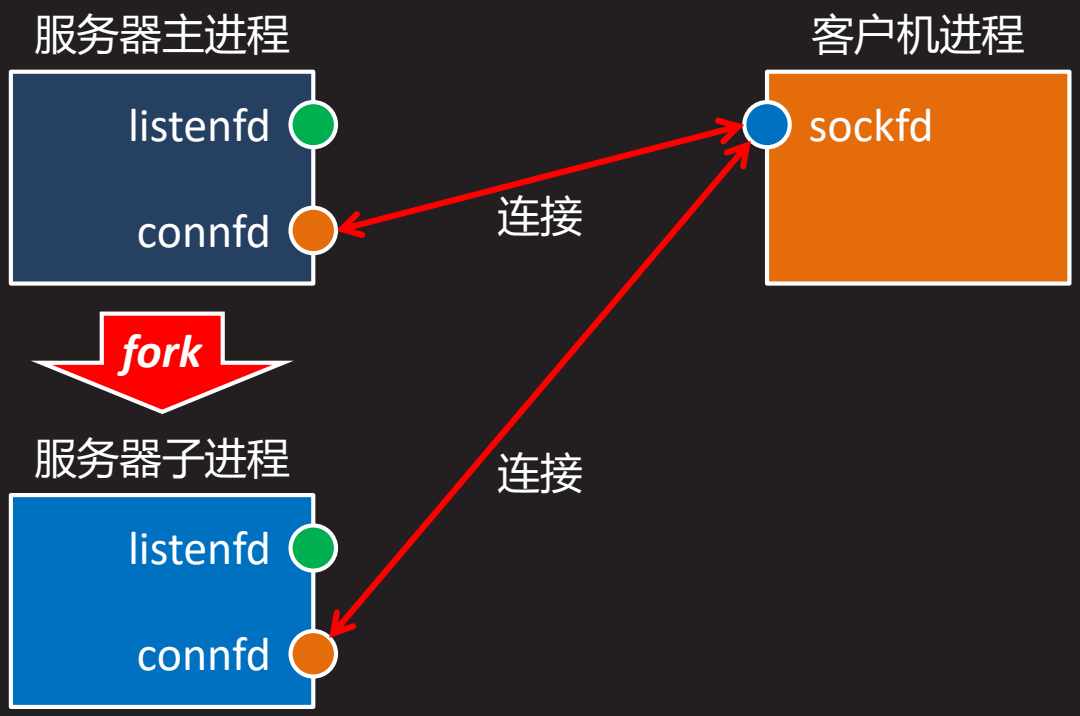


知识讲解

并发服务（续2）

- 服务器主进程调用fork函数创建子进程，子进程复制父进程的文件描述符表，因此子进程也有侦听和连接两个套接字描述符

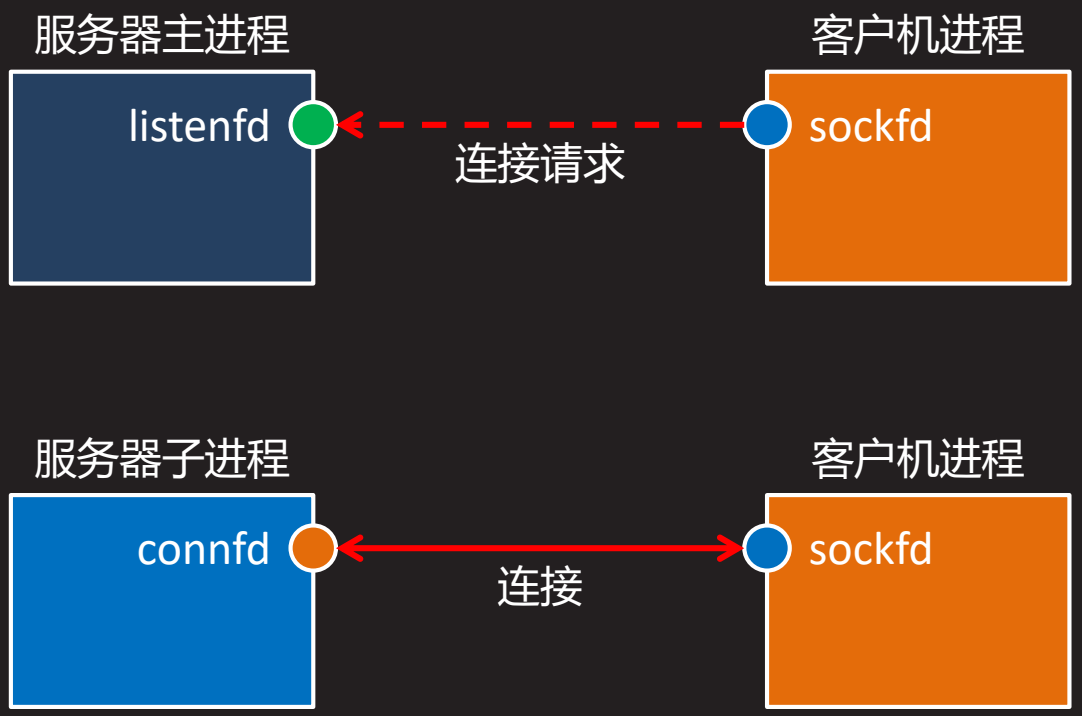
知识讲解



并发服务（续3）

- 服务器主进程关闭连接套接字；服务器子进程关闭侦听套接字。主进程通过循环继续阻塞于针对侦听套接字的accept调用，而子进程则通过连接套接字与客户机通信

知识讲解



并发服务（续4）

Technology Tarena 达内科技

- 套接字描述符与普通文件描述符一样，是带有引用计数的。在一个套接字描述符上调用close函数，并不一定真的关闭了该套接字，而只是将其引用计数减一。只有当套接字描述符的引用计数被减到零时，才真的会释放该套接字对象所占用的资源，并向对方发送FIN分节。因此服务器主进程关闭连接套接字，并不会影响子进程通过该套接字与客户机通信。同理，服务器子进程关闭侦听套接字也不会影响主进程通过该套接字继续等待连接
- 如果服务器主进程在创建子进程后不关闭连接套接字，一方面将耗尽其可用文件描述符；另一方面在子进程结束通信关闭连接套接字时，其描述符上的引用计数只会由2变成1，而不会变成0，TCP协议栈将永远保持此连接

知识讲解

基于TCP协议的客户机与服务器

【参见：tcpsvr.c、tcpcli.c】

课堂练习

- 基于TCP协议的客户机与服务器

Technology Tarena 达内科技

通信终止

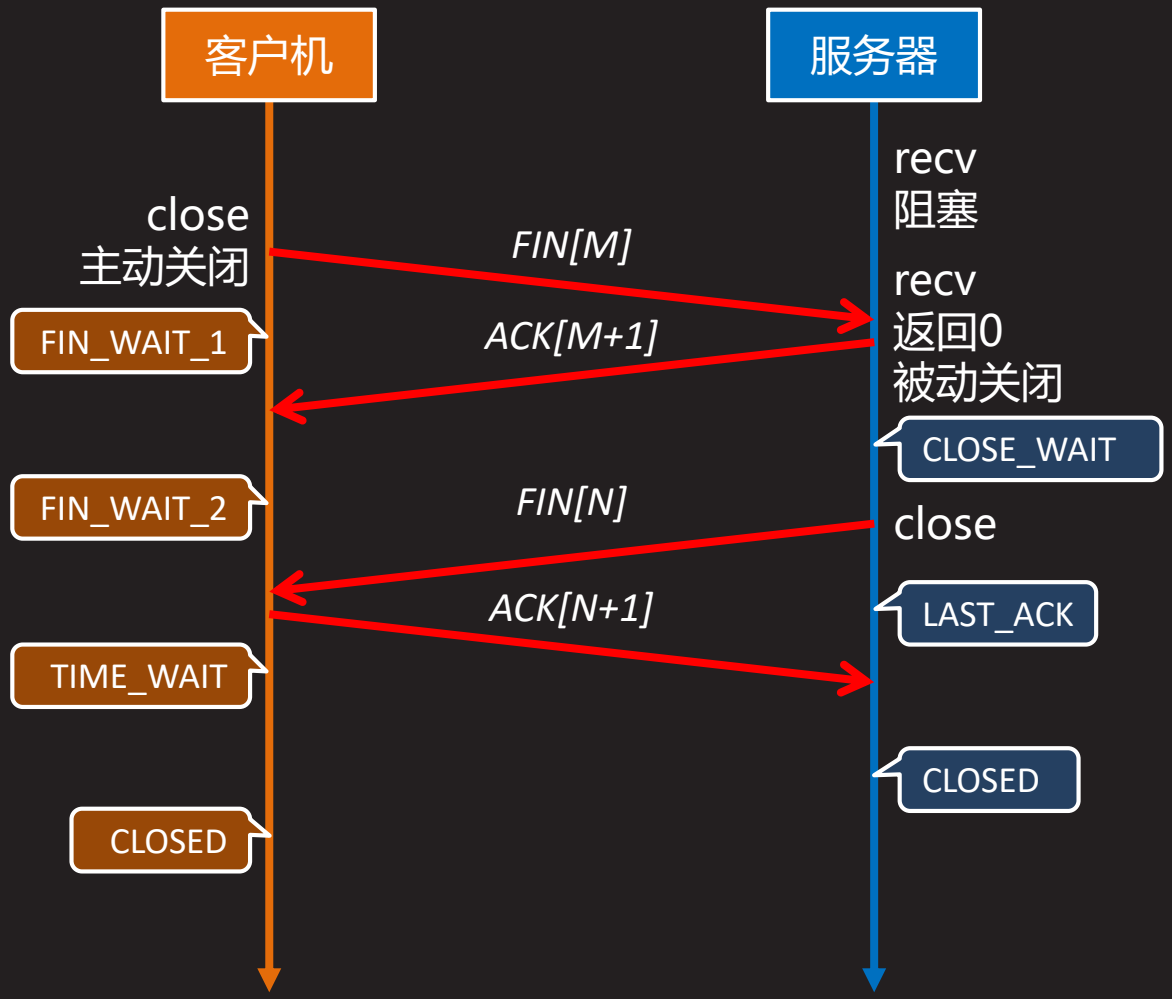
通信终止

Technology Tarena 达内科技

- 客户机主动终止通信过程
  - 在某个特定的时刻，客户机认为已经不再需要服务器继续为其提供服务了。于是它在接收完最后一个响应包以后，通过close函数关闭了与服务器通信的套接字。客户机的TCP协议栈向服务器发送FIN分节并得到对方的ACK应答
  - 服务器端专门负责与该客户机通信的子进程，此刻正试图通过recv函数接收下一个请求包，结果却因为收到来自客户机的FIN分节而返回0。于是该子进程退出收发循环，同时通过close函数关闭连接套接字，导致服务器的TCP协议栈向客户机发送FIN分节，使对方进入TIME_WAIT状态，并在收到对方的ACK应答以后，自己进入CLOSED状态
  - 随着收发循环的退出，服务器子进程终止，并在服务器主进程的太平间信号处理函数中被回收。通信过程宣告结束

通信终止（续1）

- 客户机主动终止通信过程



知识讲解

通信终止（续2）

Technology Tarena 达内科技

- 服务器主动终止通信过程
  - 服务器端专门负责和某个特定客户机通信的子进程，在运行过程中出现错误，不得不调用close函数关闭连接套接字，或者直接退出，甚至被信号杀死。于是服务器的TCP协议栈向客户机发送FIN分节并得到对方的ACK应答
    - 如果客户机这时正试图通过recv函数接收响应包，那么该函数会返回0。客户机可据此判断服务器已宕机，直接通过close函数关闭与该服务器通信的套接字，终止通信过程
    - 如果客户机这时正试图通过send函数发送请求包，那么该函数并不会失败，但会导致对方以RST分节响应，该响应分节甚至会先于FIN分节被紧随其后的recv函数收到并返回-1，而errno则为ECONNRESET。这也是终止通信的条件之一
  - 任何试图向已接收到RST分节的套接字做写操作的进程，都会收到SIGPIPE(13)信号，该信号的默认处理是杀死进程

通信终止（续3）

Technology Tarena 达内科技

- 服务器主机不可达(主机崩溃、网络中断、路由失效等)
  - 在服务器主机不可达的情况下，无论是客户机还是服务器，它们的TCP协议栈都不可能再有任何数据分节的交换。因此，客户机通过send函数发送完请求包以后，会阻塞在recv函数上等待来自服务器的响应包
  - 这时客户机的TCP协议栈会持续地重传数据分节，试图得到对方的ACK应答。源自伯克利的实现最多重传12次，最长等待9分钟。当TCP最终决定放弃时，会通过recv函数向用户进程返回失败，并置errno为ETIMEOUT或EHOSTUNREACH或ENETUNREACH
- 服务器主机崩溃后重启
  - 服务器主机崩溃后重启，它的TCP协议栈丢失了崩溃前所有的连接信息，对所接收到的任何数据一律响应RST分节

知识讲解

UDP客户机/服务器


UDP客户机/服务器
UDP协议的基本特征
常用函数
编程模型
服务模型
UDP协议的基本特征
接收数据从
发送数据到
编程模型
迭代服务

Technology
Tarena
达内科技

UDP协议的基本特征

UDP协议的基本特征

Technology Tarena 达内科技

- UDP不提供客户机与服务器的连接
  - UDP的客户机与服务器不必存在长期关系。一个UDP的客户机在通过一个套接字向一个UDP服务器发送了一个数据报之后，马上可以通过同一个套接字向另一个UDP服务器发送另一个数据报。同样，一个UDP服务器也可以通过同一个套接字接收来自不同客户机的数据报
- UDP不保证数据传输的可靠性和有序性
  - UDP的协议栈底层不提供诸如确认、超时重传、RTT估算以及序列号等机制。因此UDP数据报在网络传输的过程中，可能丢失，也可能重复，甚至重新排序。应用程序必须自己处理这些情况

知识讲解

UDP协议的基本特征（续1）

Technology Tarena 达内科技

- UDP不提供流量控制
  - UDP的协议栈底层只是一味地按照发送方的速率发送数据，全然不顾接收方的缓冲区是否装得下
- UDP是记录式传输协议
  - 每个UDP数据报都有一定长度，一个数据报就是一条记录。如果数据报正确地到达了目的地，那么数据报的长度将被传递给接收方的应用进程
  - 发送方逐个数据报地发送，接收方逐个数据报地接收，不存在一次接收部分数据报或多个数据报的可能
- UDP是全双工的
  - 在一个UDP套接字上，应用程序在任何时候都既可以发送数据也可以接收数据

知识讲解

Technology Tarena 达内科技

常用函数

接收数据从

Technology Tarena 达内科技

- 从指定的地址结构接收数据

```
#include <sys/socket.h>

ssize_t recvfrom (int sockfd, void* buf, size_t len,
    int flags, struct sockaddr* src_addr,
    socklen_t* addrlen);
```

成功返回实际接收到的字节数，失败返回-1

- *sockfd*：套接字描述符
- *buf*：应用程序接收缓冲区
- *len*：期望接收的字节数

知识讲解

接收数据从（续1）

Technology Tarena 达内科技

- 从指定的地址结构接收数据
  - ***flags***：接收标志，一般取0，还可取以下值
    MSG_DONTWAIT - 以非阻塞方式接受数据
    MSG_OOB - 接收带外数据
    MSG_PEEK - 只查看可接收的数据，函数返回后数据依然留在接收缓冲区中
    MSG_WAITALL - 等待所有数据，即不接收到***len***字节的数据，函数就不返回
  - ***src_addr***：输出数据报发送者的地址结构，可置NULL
  - ***addrlen***：输入***src_addr***参数所指向内存块的字节数，输出数据报发送者地址结构的字节数，可置NULL
- recvfrom函数返回0，表示接收到一个空数据报(只有IP和UDP包头而无数据内容)，与对方是否关闭套接字无关

接收数据从（续2）

Technology Tarena 达内科技

- 例如
  - char buf[1024];
    struct sockaddr_in addrcli = {};
    socklen_t addrlen = sizeof (addrcli);
    ssize_t rcvd = recvfrom (sockfd, buf, sizeof (buf), 0,
        (struct sockaddr*)&addrcli, &addrlen);
    if (rcvd == -1) {
        perror ("recvfrom");
        exit (EXIT_FAILURE);
    }
    buf[rcvd] = '\0';
    printf ("%s\n", buf);

知识讲解

发送数据到

Technology Tarena 达内科技

- 向指定的地址结构发送数据

```
#include <sys/socket.h>

ssize_t sendto (int sockfd, const void* buf, size_t len,
    int flags, const struct sockaddr* dest_addr,
    socklen_t addrlen);
```

成功返回实际被发送的字节数，失败返回-1

- *sockfd*：套接字描述符
- *buf*：应用程序发送缓冲区
- *len*：期望发送的字节数

知识讲解

发送数据到（续1）

- 向指定的地址结构发送数据
  - ***flags***：发送标志，一般取0，还可取以下值
    `MSG_DONTWAIT` - 以非阻塞方式发送数据
    `MSG_OOB` - 发送带外数据
    `MSG_DONTROUTE` - 不查路由表，直接在本地网络中寻找目的主机
  - ***dest_addr***：数据报接收者的地址结构
  - ***addrlen***：数据报接收者地址结构的字节数

知识讲解

发送数据到（续2）

Technology Tarena 达内科技

- 例如
  - char buf[1024];
    gets (buf);
    struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons (8888);
    addr.sin_addr.s_addr = inet_addr ("192.168.182.48");
    ssize_t sent = sendto (sockfd, buf, strlen (buf) *
    sizeof (buf[0]), 0, (struct sockaddr*)&addr, sizeof (addr));
    if (sent == -1) {
    perror ("send");
    exit (EXIT_FAILURE);
    }

Technology Tarena 达内科技

编程模型

编程模型

Technology Tarena 达内科技

- 基于UDP协议的无连接编程模型

步骤	服务器		客户机		步骤
1	创建套接字	socket	socket	创建套接字	1
2	准备地址结构	sockaddr_in	sockaddr_in	准备地址结构	2
3	绑定地址	bind	——	——	——
4	接收请求	recvfrom	sendto	发送请求	3
5	发送响应	sendto	recvfrom	接收响应	4
6	关闭套接字	close	close	关闭套接字	5

知识讲解

- UDP服务器的阻塞焦点不在accept函数上，而在recvfrom函数上。任何一个UDP客户机通过sendto函数发送的请求数据都可以被recvfrom函数返回给UDP服务器，其输出的客户机地址结构***src_addr***可直接被用于向客户机返回响应时调用sendto函数的输入***dest_addr***

编程模型（续1）

Technology Tarena 达内科技

- 基于UDP协议的有连接编程模型

步骤	服务器		客户机		步骤
1	创建套接字	socket	socket	创建套接字	1
2	准备地址结构	sockaddr_in	sockaddr_in	准备地址结构	2
3	绑定地址	bind	connect	建立连接	3
4	接收请求	recvfrom	send/write	发送请求	4
5	发送响应	sendto	recv/read	接收响应	5
6	关闭套接字	close	close	关闭套接字	6

知识讲解

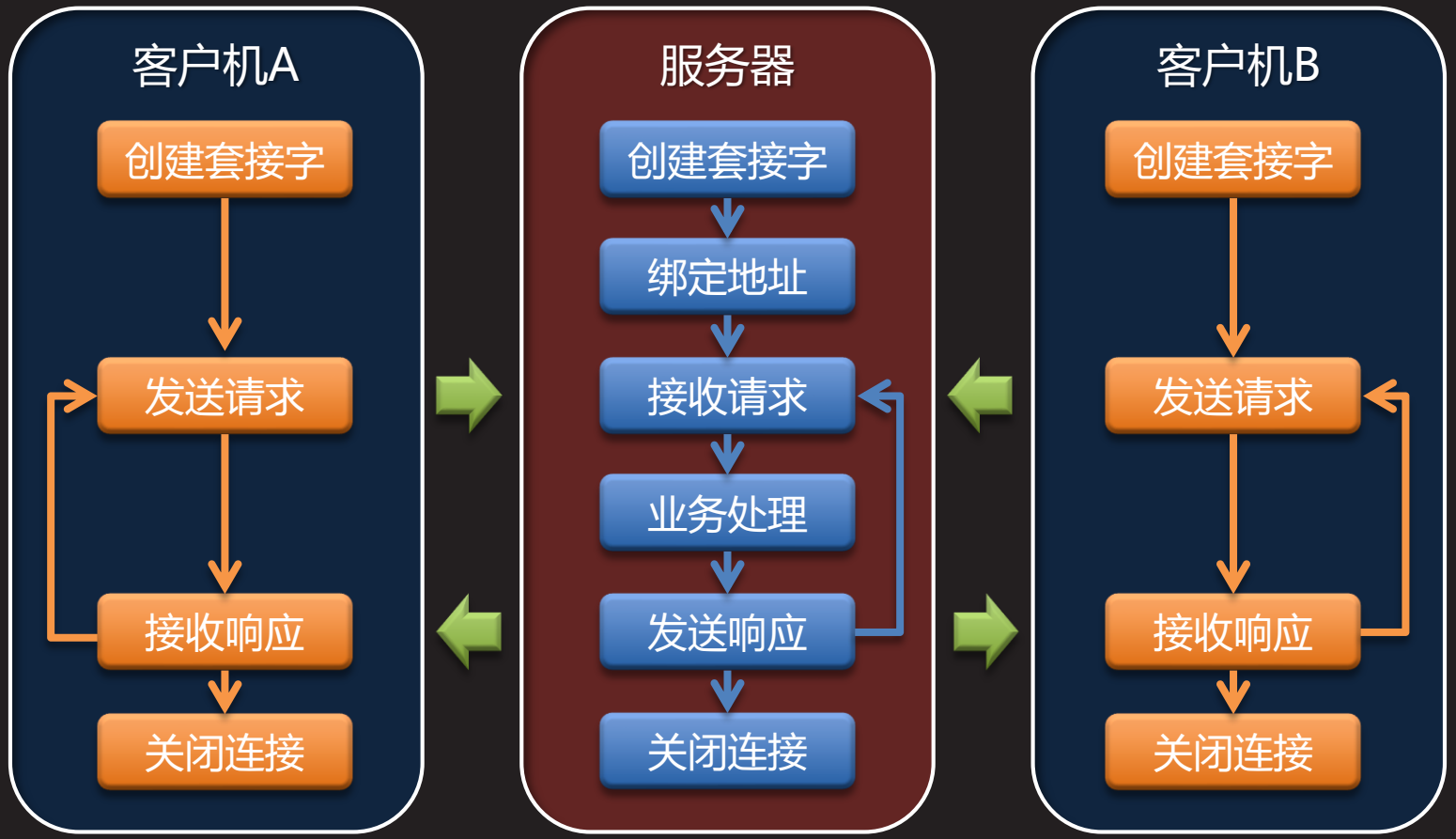
- UDP的connect函数与TCP的connect函数完全不同，既无三路握手，亦无虚拟电路，而仅仅是将传递给该函数的对方地址结构缓存在套接字对象中。此后收发数据时，可不使用recvfrom/sendto函数，而是使用recv/send或者read/write函数，直接和所连接的对方主机通信

Technology Tarena 达内科技

服务模型

迭代服务

- 基于UDP协议建立通信的客户机和服务器，不需要维持长期的连接。因此UDP服务器在一个单线程中，以循环迭代的方式即可处理来自不同客户机的业务需求



知识讲解

Technology Tarena 达内科技

基于UDP协议的客户机与服务器

【参见：udpsvr.c、udpcli.c、udpcon.c】

课堂练习

- 基于UDP协议的客户机与服务器

基于并发进程的网络银行

Technology Tarena 达内科技

需求分析

需求分析

Technology Tarena 达内科技

- 基于并发进程的网络银行，包括客户机和服务器两个组成部分。其中客户机负责提供用户界面，接收用户的输入，并向其提供输出，服务器负责后台业务逻辑的处理
- 每一种银行业务皆被实现为一个独立的进程，谓之业务服务。服务器主进程负责各业务服务进程的启动与终止
- 客户机与各业务服务之间通过TCP协议在同一台或不同机器上通信，故谓之网络银行。客户机根据所需要的业务种类连接相应的业务服务。后者负责创建专门与该客户机通信的交易子进程，接收客户机的请求、完成业务处理，并向客户机返回响应，直至关闭连接，终止进程
- 业务数据需要在服务器端持久化，采用文件系统数据库

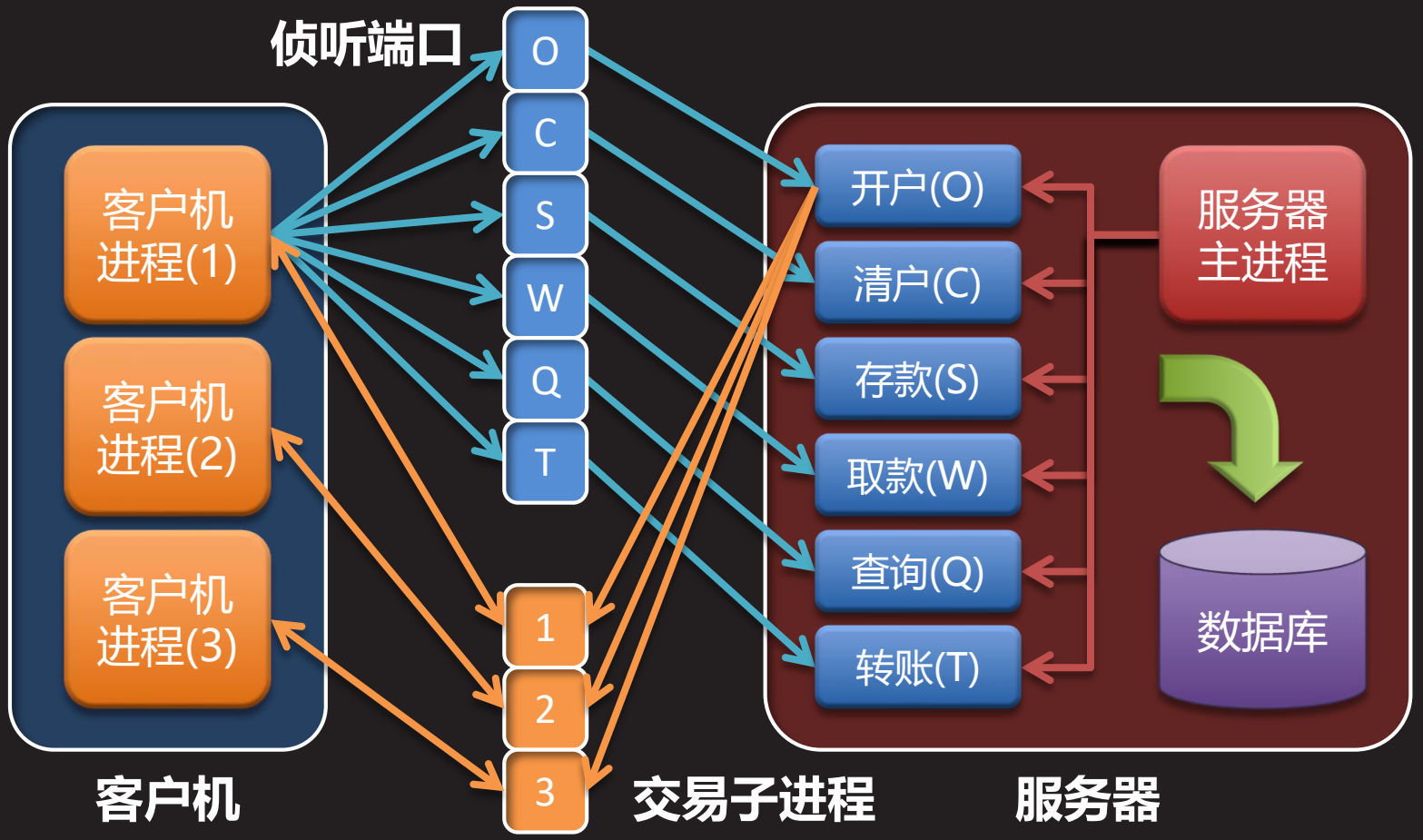
Technology Tarena 达内科技

概要设计

概要设计

Technology Tarena 达内科技

知识讲解



- 每个业务服务均以相应的业务标识作为侦听端口号
- 每个业务服务都用独立的子进程处理客户机的业务

Technology Tarena 达内科技

开发计划

开发计划

Technology Tarena 达内科技

1. 编写网络通信模块：network.h、network.c
2. 编写太平间信号处理模块：mortuary.h、mortuary.c
3. 修改各业务服务模块：open.c、close.c、save.c、withdraw.c、query.c、transfer.c
4. 修改服务器主进程模块：server.c
5. 修改客户机模块：client.c
6. 修改构建脚本：makefile
7. 链接、运行、测试

知识讲解

基于并发进程的网络银行

【参见：bank/】

课堂练习

- 基于并发进程的网络银行

Technology Tarena 达内科技

总结和答疑