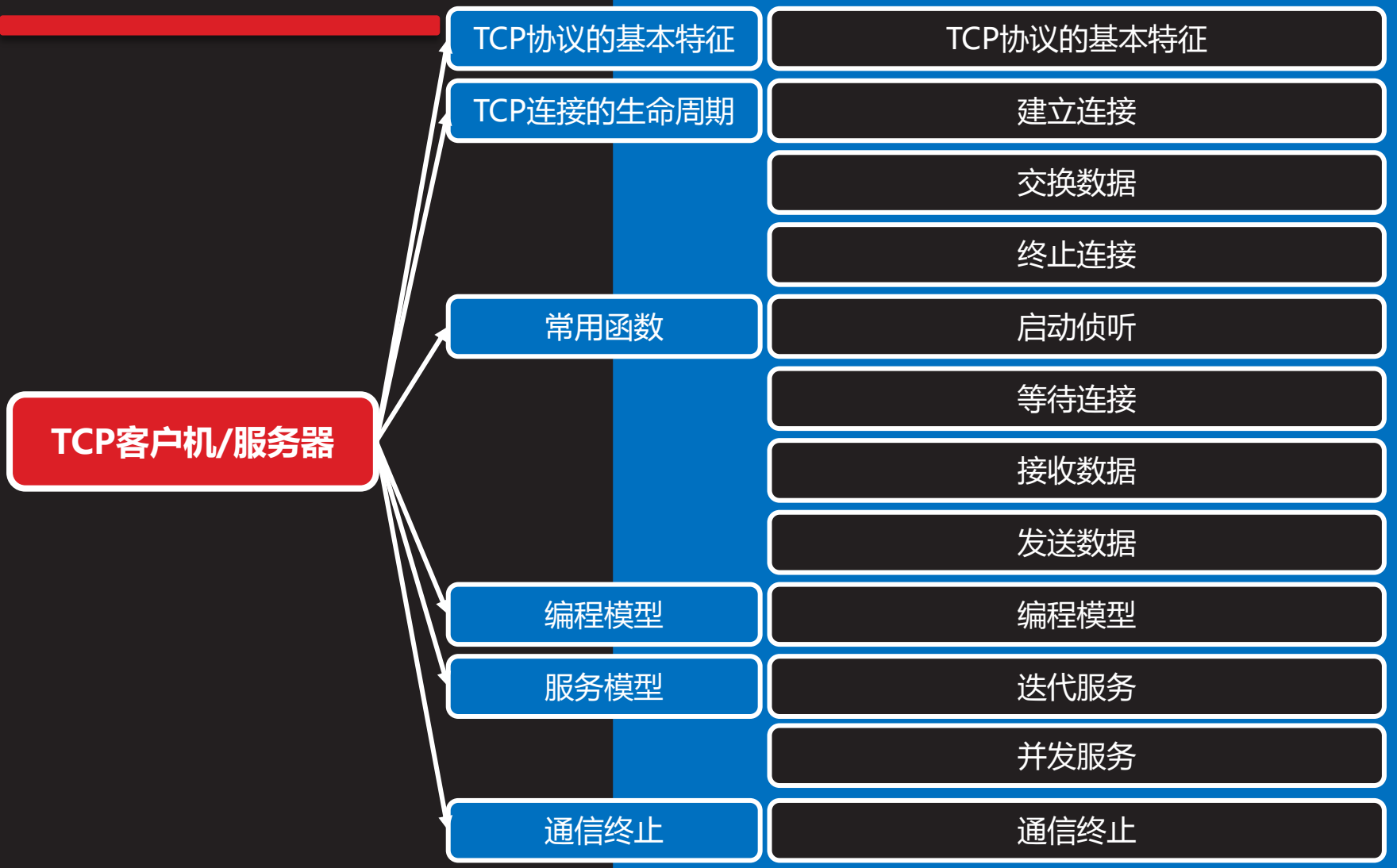
Technology Tarena 达内科技

Unix系统高级编程

TCP客户机/服务器 Unit25

TCP客户机/服务器

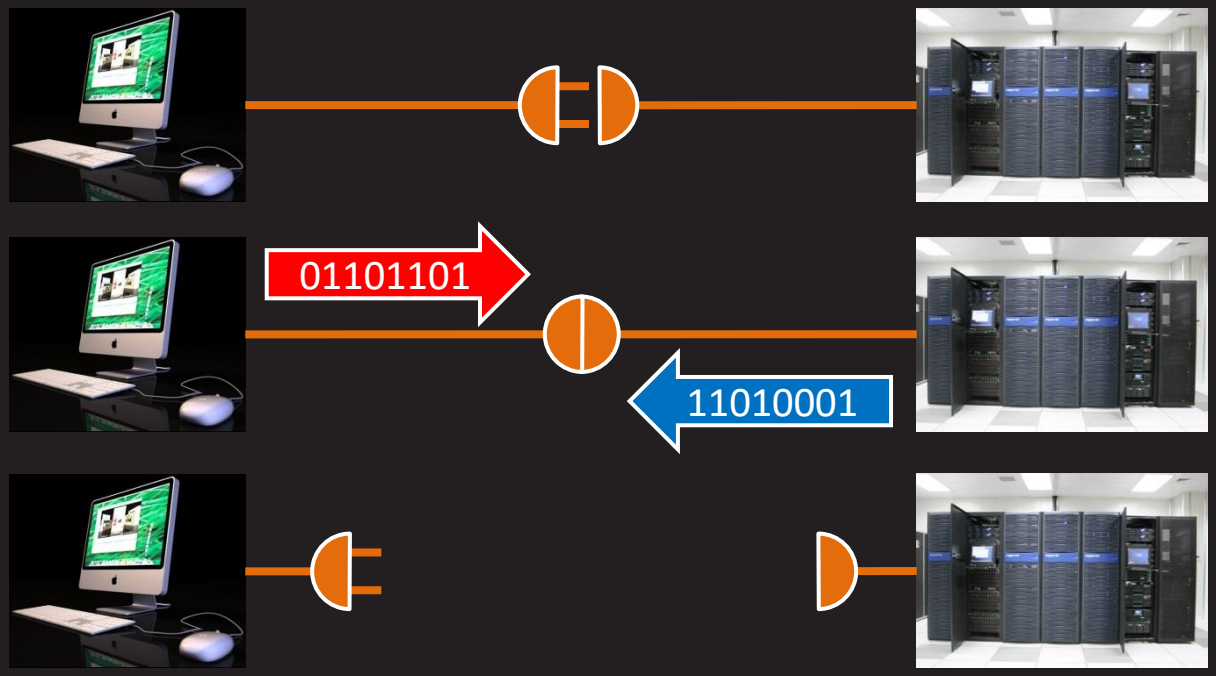

TCP客户机/服务器
TCP协议的基本特征
TCP连接的生命周期
常用函数
编程模型
服务模型
通信终止
TCP协议的基本特征
建立连接
交换数据
终止连接
启动侦听
等待连接
接收数据
发送数据
编程模型
迭代服务
并发服务
通信终止

TCP协议的基本特征

TCP协议的基本特征

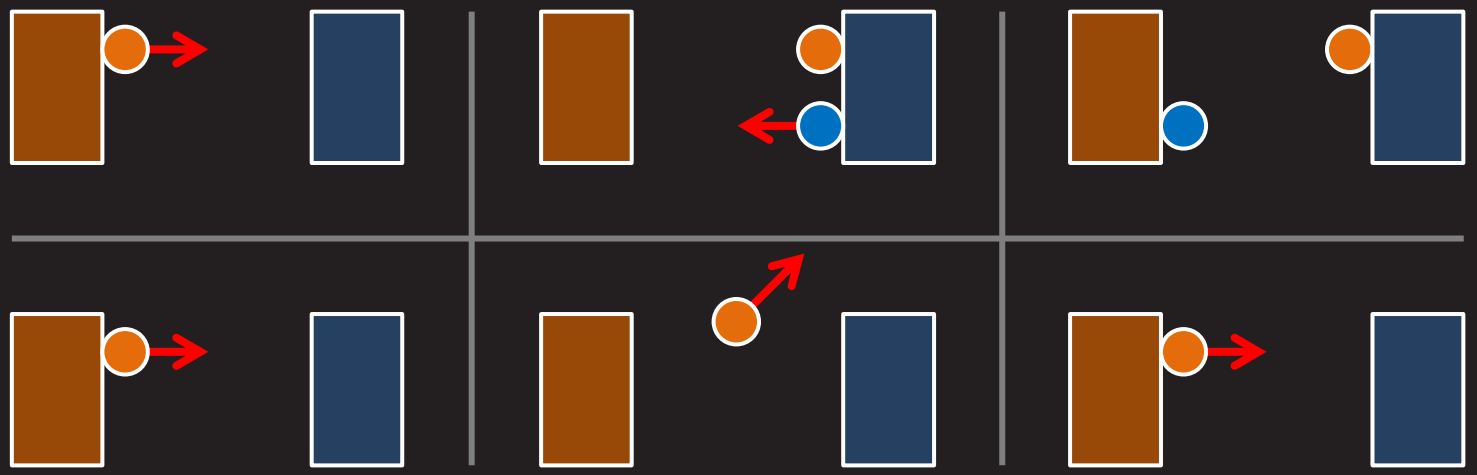
- TCP提供客户机与服务器的连接
  - 一个完整TCP通信过程需要依次经历三个阶段
    - 首先，客户机必须建立与服务器的连接，所谓虚电路
    - 然后，凭借已建立好的连接，通信双方相互交换数据
    - 最后，客户机与服务器双双终止连接，结束通信过程

知识讲解



TCP协议的基本特征（续1）

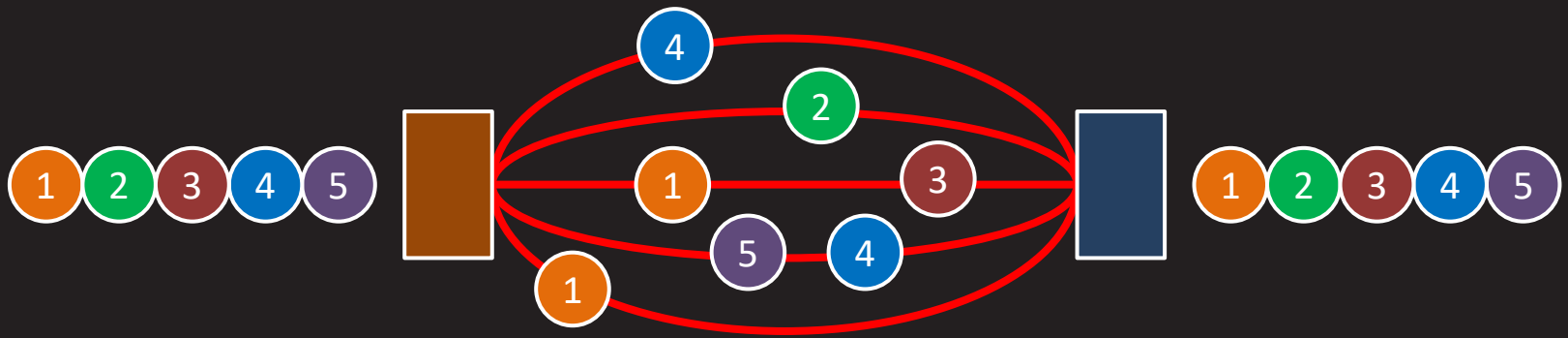
- TCP保证数据传输的可靠性
  - TCP的协议栈底层在向另一端发送数据时，会要求对方在一个给定的时间窗口内返回确认。如果超过了这个时间窗口仍没有收到确认，则TCP会重传数据并等待更长的时间。只有在数次重传均告失败以后，TCP才会最终放弃。TCP含有用于动态估算数据往返时间(Round-Trip Time, RTT)的算法，因此它知道等待一个确认需要多长时间



知识讲解

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

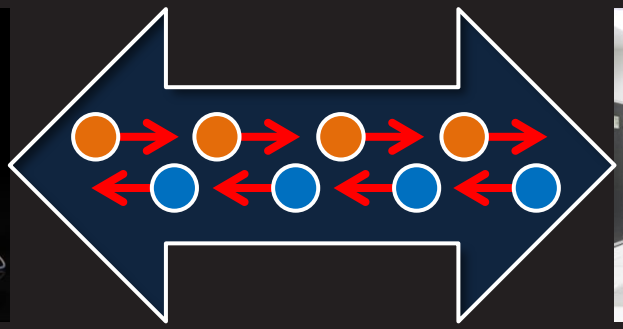
    - 不定长记录+分隔符

    - 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

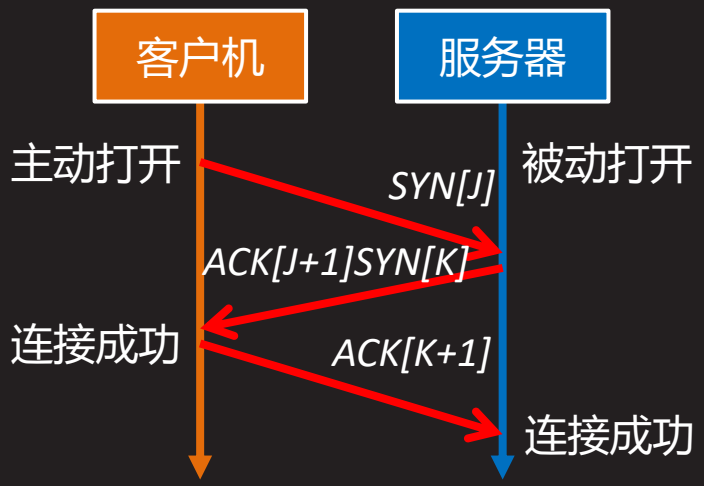


Technology Tarena 达内科技

TCP连接的生命周期

建立连接

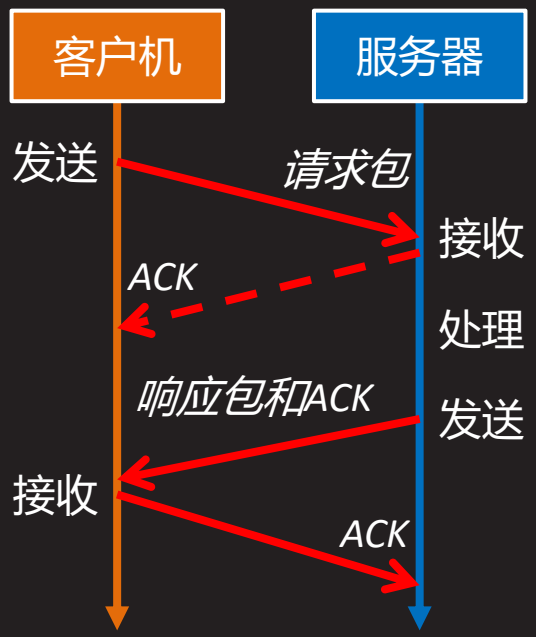
- 被动打开
  - 服务器必须首先做好准备随时接受来自客户机的连接请求
- 三路握手
  - 客户机的TCP协议栈向服务器发送一个SYN分节，告知对方自己将在连接中发送数据的初始序列号，谓之主动打开
  - 服务器的TCP协议栈向客户机发送一个单个分节，其中不仅包括对客户机SYN分节的ACK应答，还包含服务器自己的SYN分节，以告知对方自己在同一连接中发送数据的初始序列号
  - 客户机的TCP协议栈向服务返回ACK应答，以表示对服务器所发SYN的确认



知识讲解

交换数据

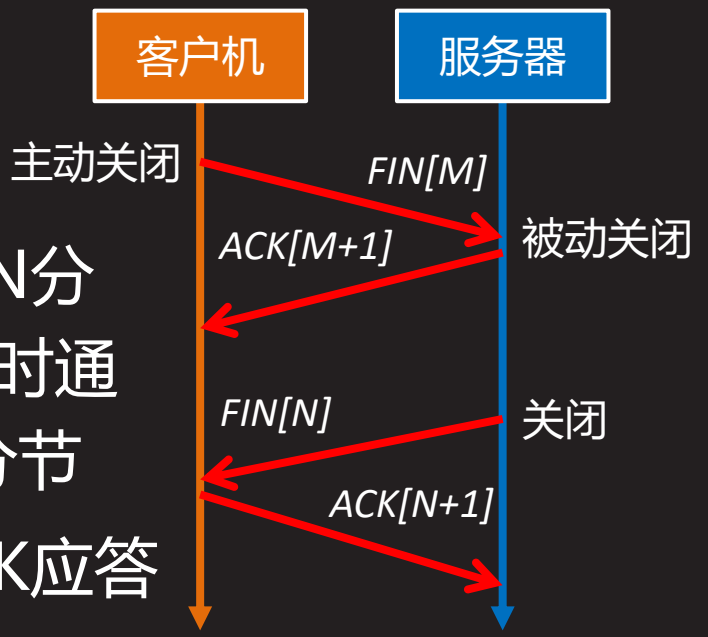
- 一旦连接建立，客户机即可构造请求包并发往服务器
- 服务器接收并处理来自客户机的请求包，构造响应包
- 服务器向客户机发送响应包，同时捎带对客户机请求包的ACK应答。但如果服务器处理请求和构造响应的时间长于200毫秒，则应答也可能先于响应发出
- 客户机接收来自服务器的响应包，同时向对方发送ACK应答



知识讲解

终止连接

- 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

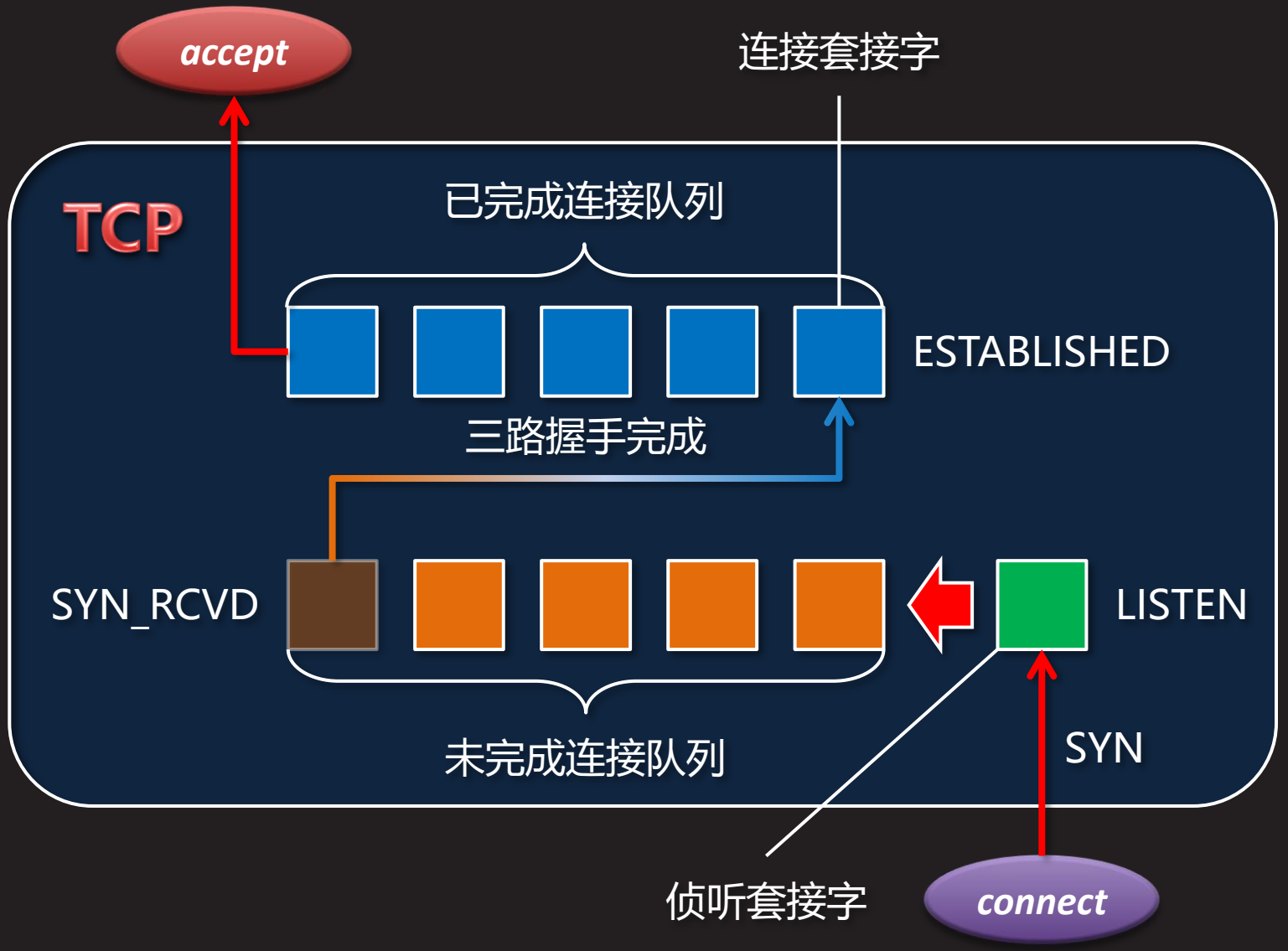
知识讲解

启动侦听（续1）

- 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）



启动侦听（续3）

- 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）

- 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

- 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）

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

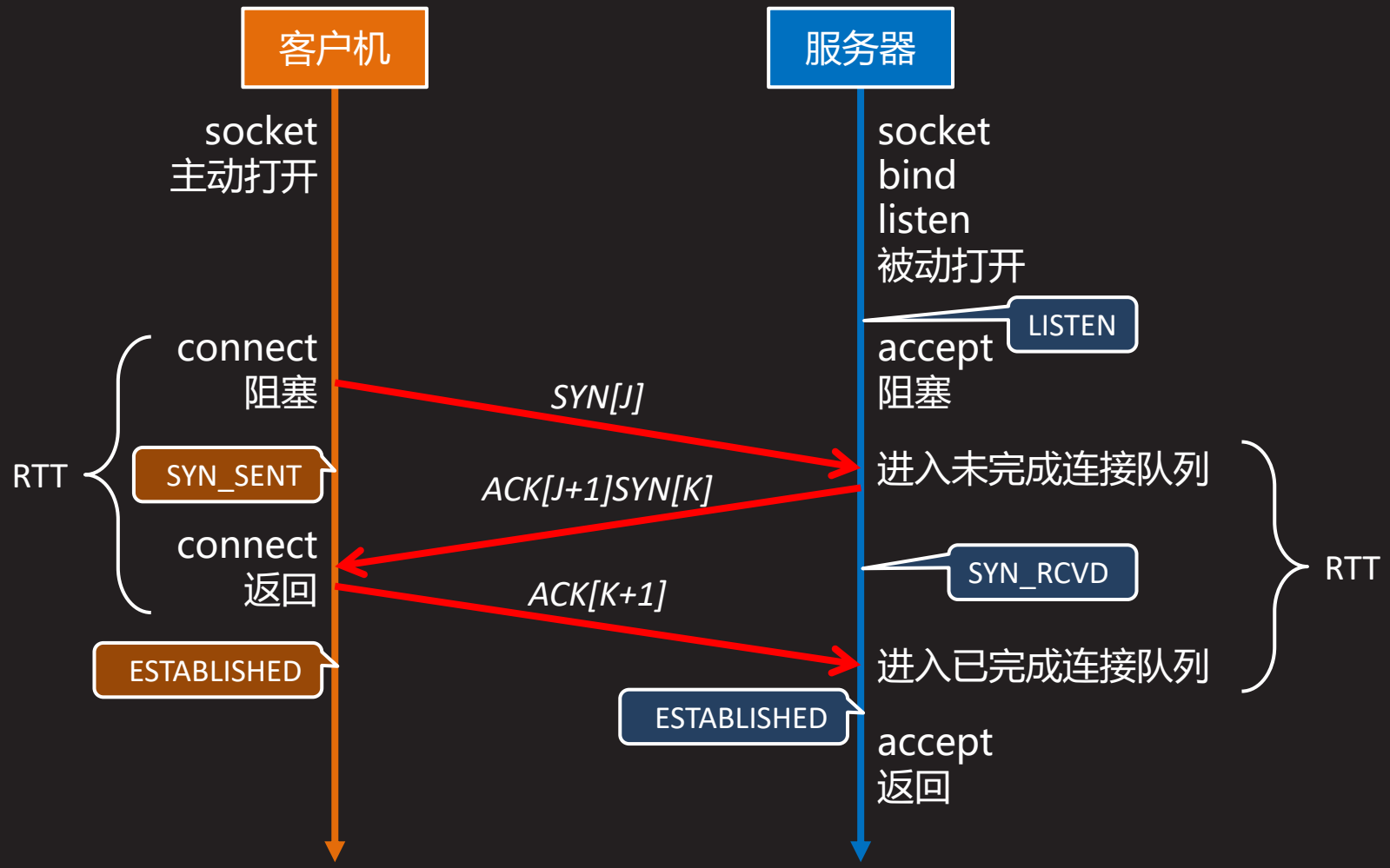
等待连接（续2）

- 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）

- 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）

- 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）

- 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

- 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）

- 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）

- 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

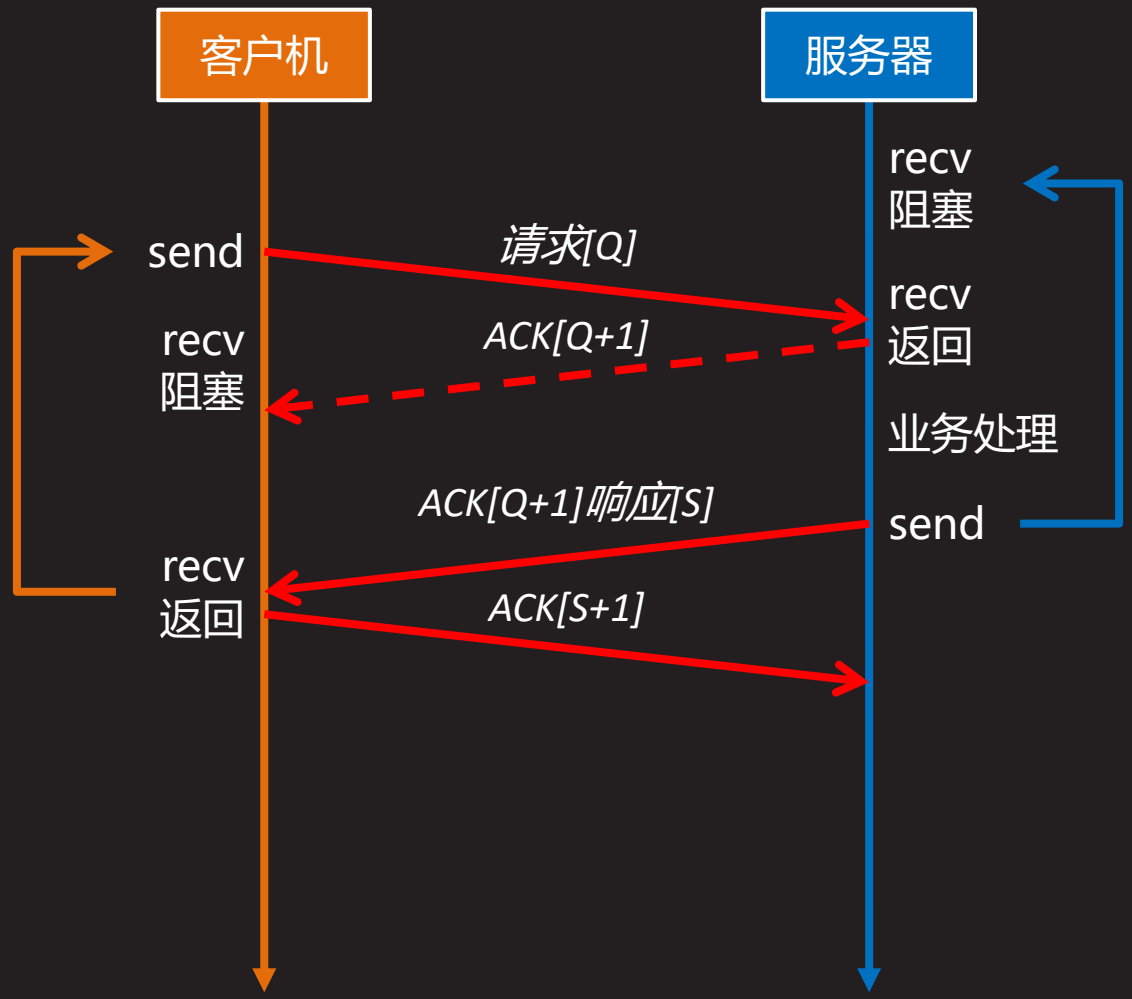
知识讲解

发送数据（续3）

- 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);
    }

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

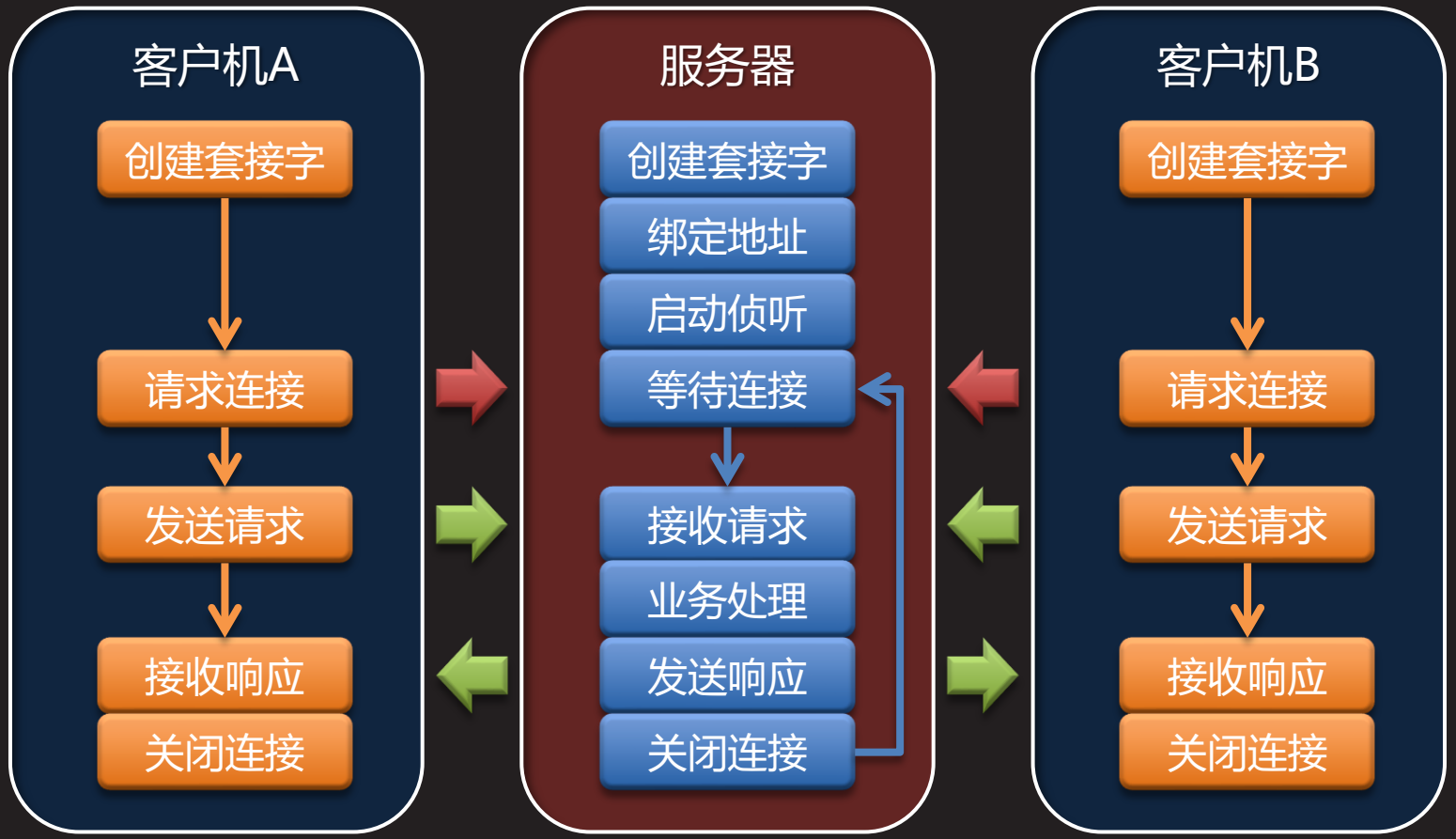
知识讲解

Technology Tarena 达内科技

服务模型

迭代服务

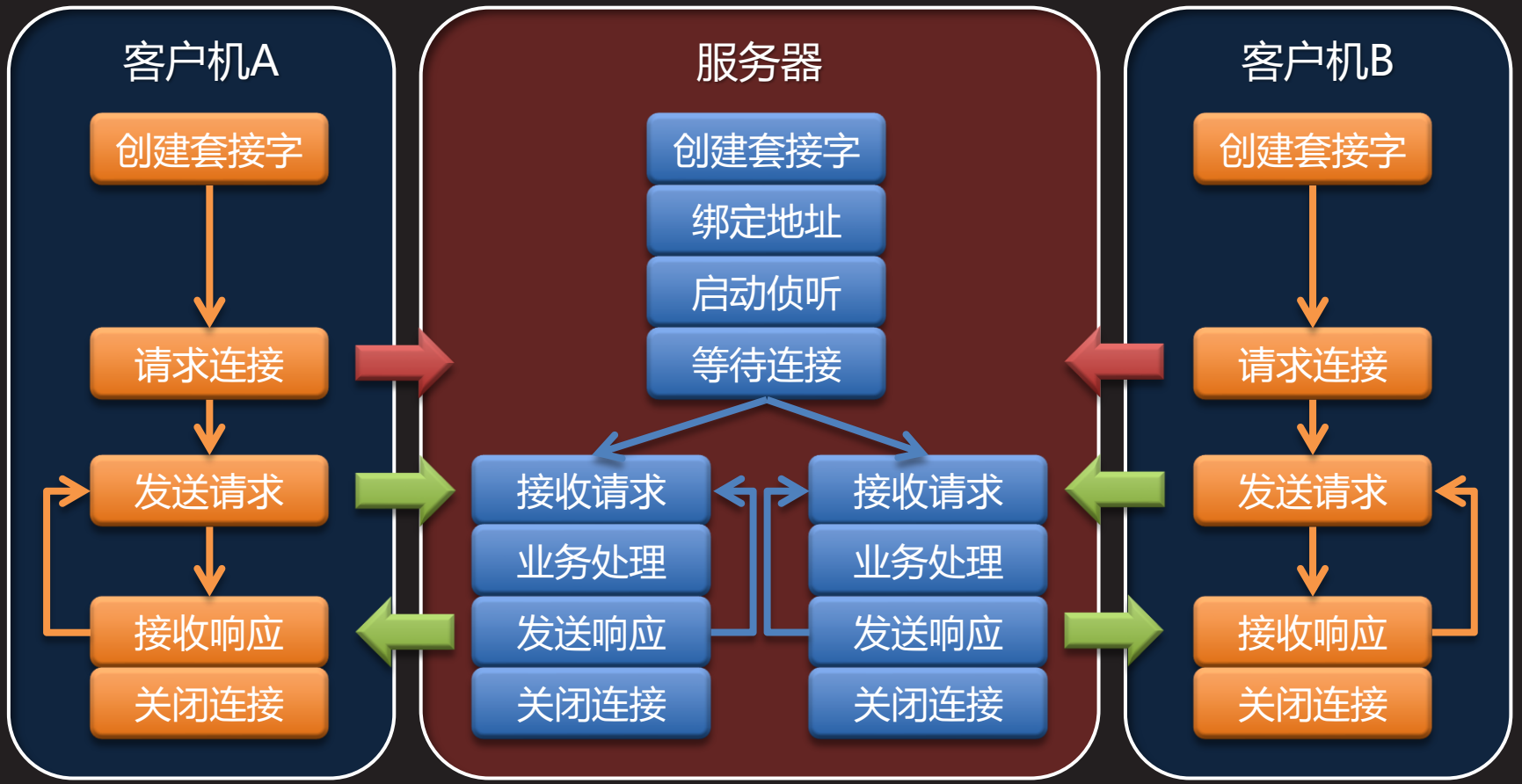
- 服务器在单线程中以循环迭代的方式依次处理每个客户机的业务需求。迭代模型的前提是针对每个客户机的处理时间必须足够短暂，否则会延误对其它客户机的响应



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

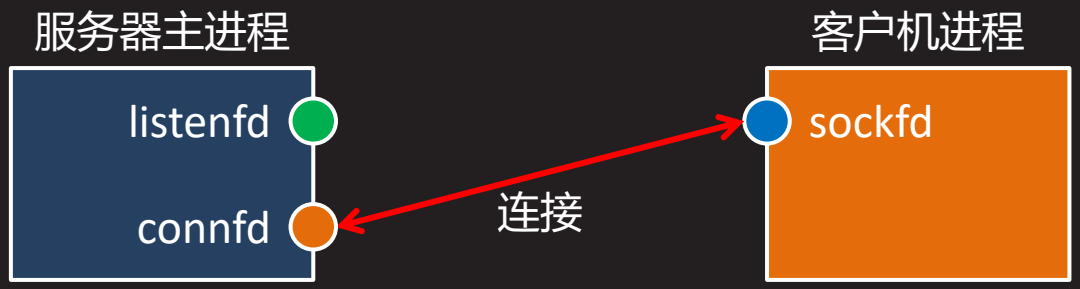


知识讲解

并发服务（续1）

- 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

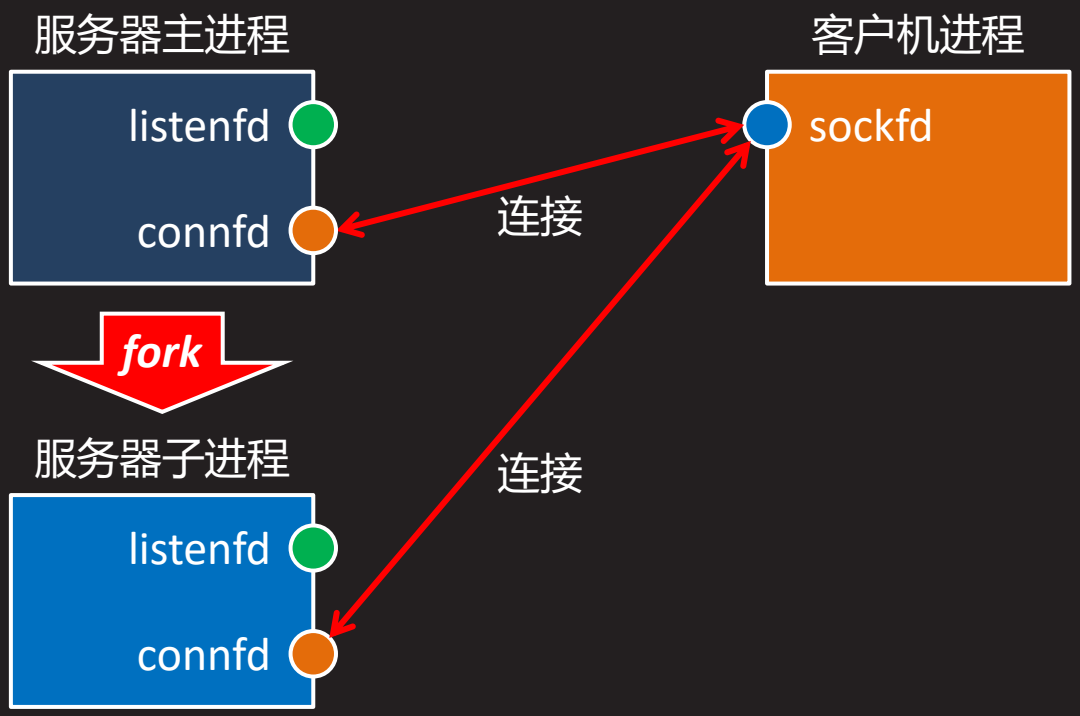


知识讲解

并发服务（续2）

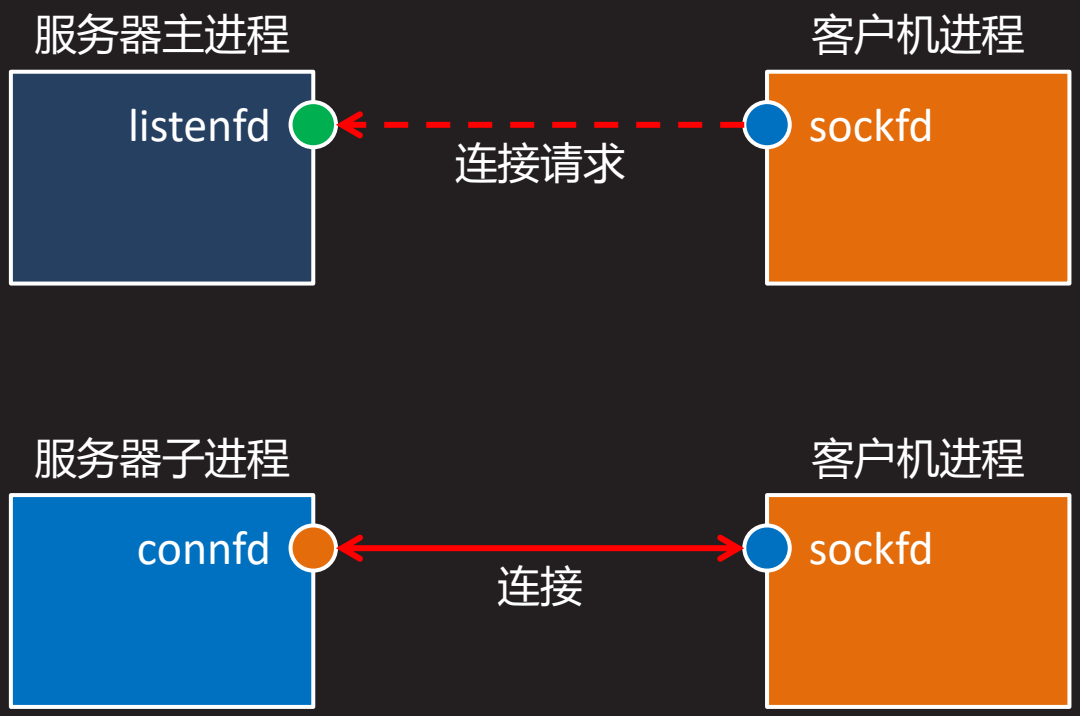
- 服务器主进程调用fork函数创建子进程，子进程复制父进程的文件描述符表，因此子进程也有侦听和连接两个套接字描述符

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）

- 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

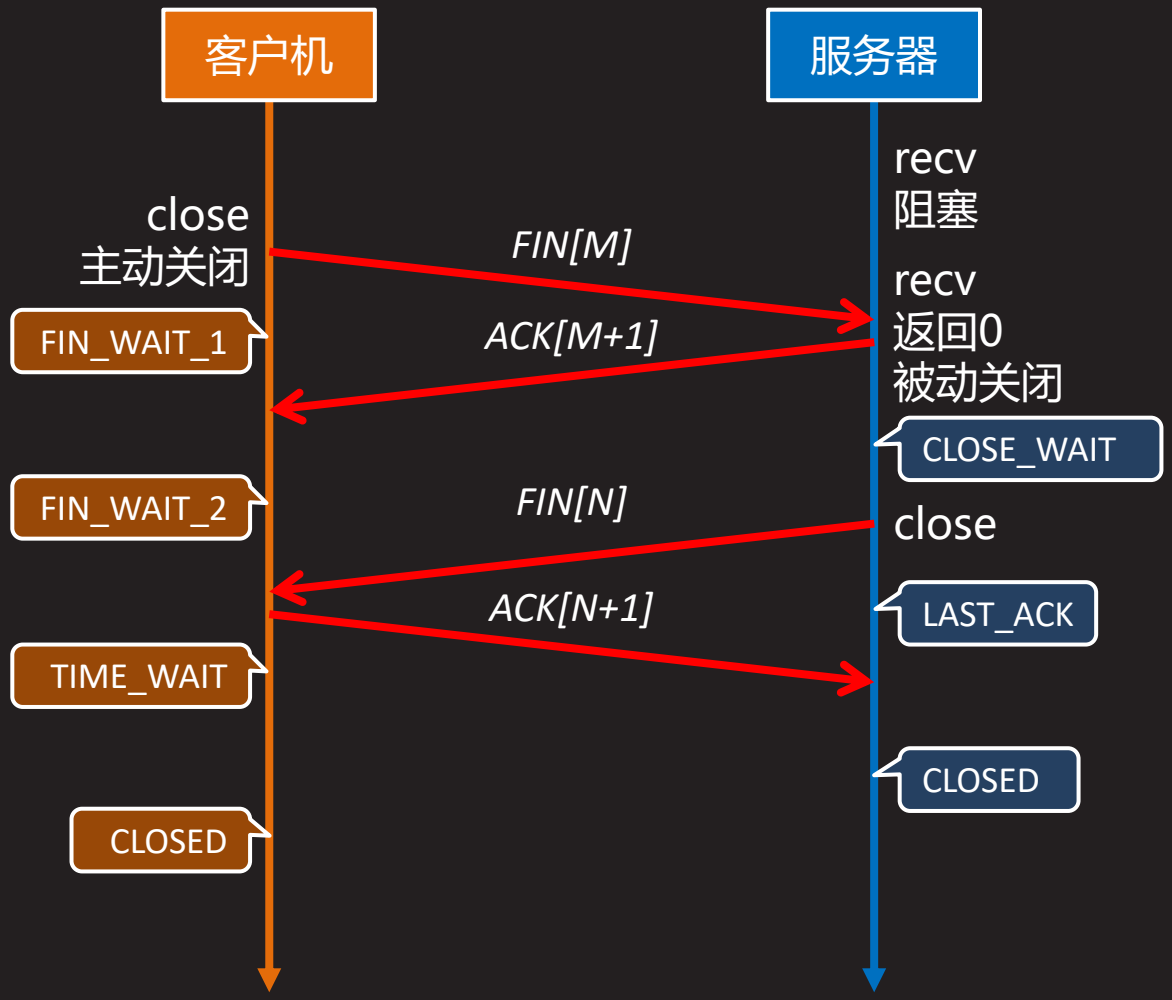
通信终止

通信终止

- 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）

- 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）

- 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

Technology Tarena 达内科技

总结和答疑