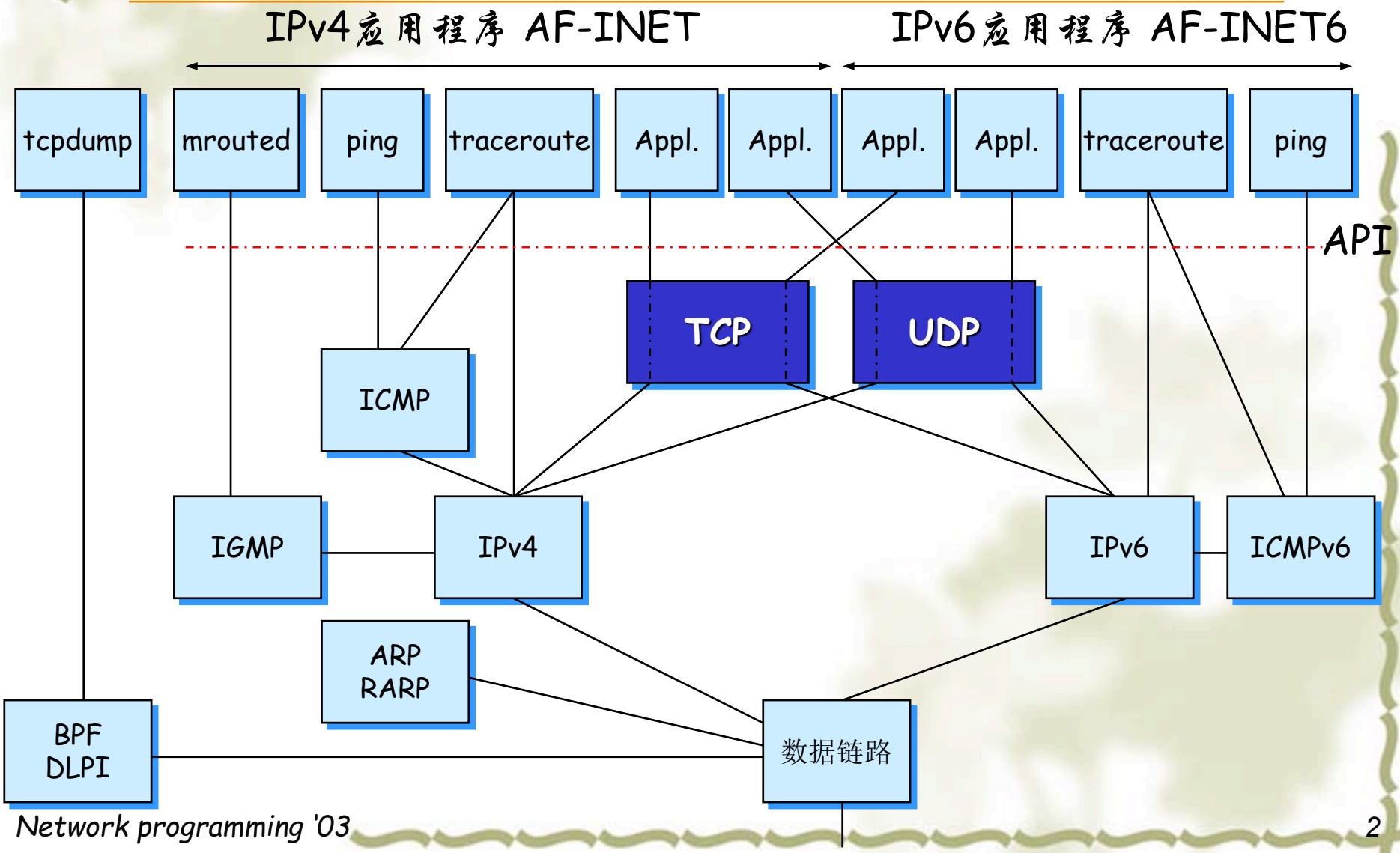
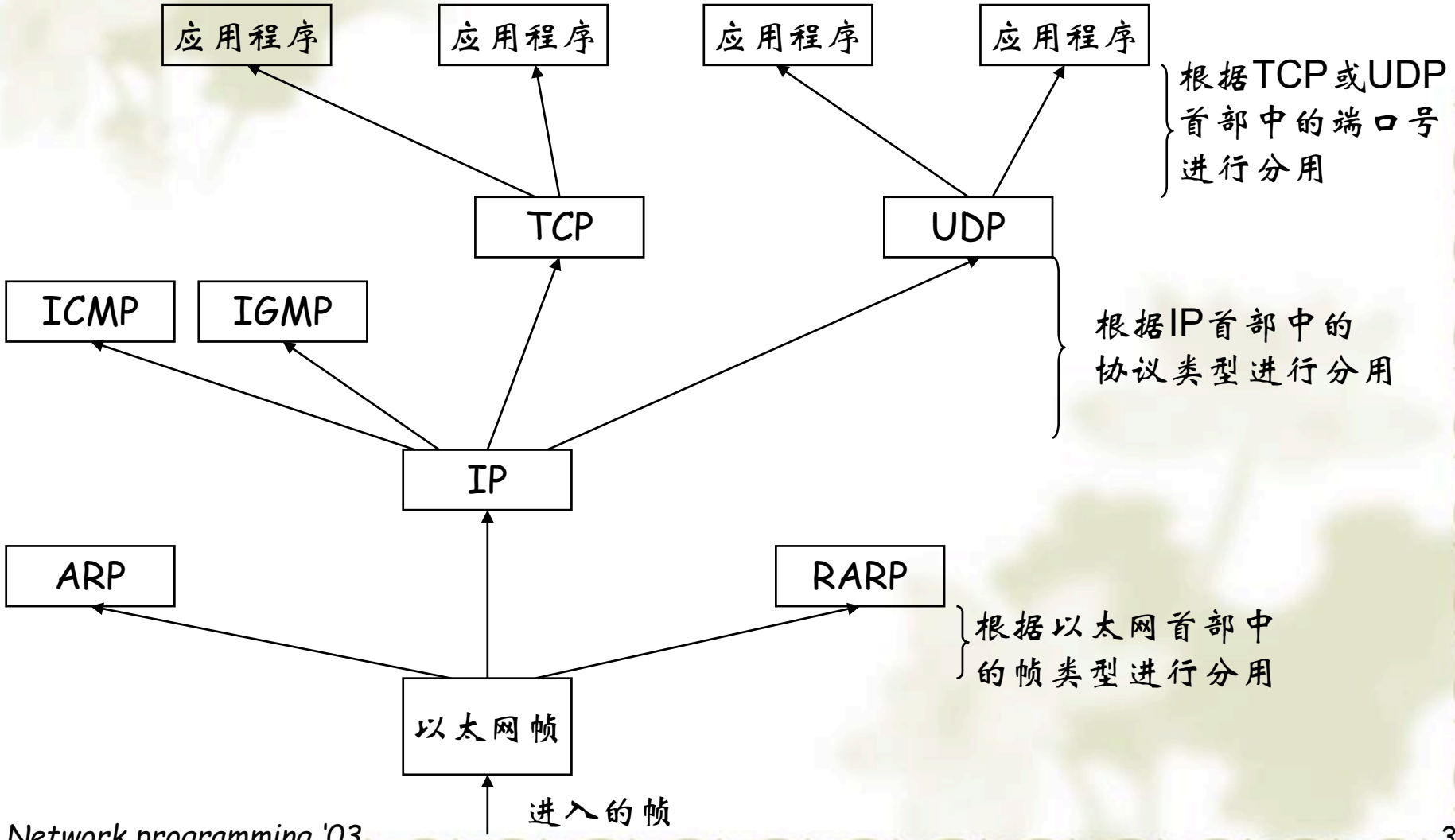
《互联网络程序设计》
复习材料

任立勇

TCP/IP协议族概貌


IPv4应用程序 AF-INET
IPv6应用程序 AF-INET6
tcpdump
mrouted
ping
traceroute
Appl.
Appl.
Appl.
Appl.
traceroute
ping
API
TCP
UDP
ICMP
IGMP
IPv4
IPv6
ICMPv6
ARP
RARP
BPF
DLPI
数据链路

以太网数据帧分用过程


应用程序
应用程序
应用程序
应用程序
根据TCP或UDP
首部中的端口号
进行分用
TCP
UDP
ICMP
IGMP
根据IP首部中的
协议类型进行分用
IP
ARP
RARP
根据以太网首部中
的帧类型进行分用
以太网帧
进入的帧

TCP协议

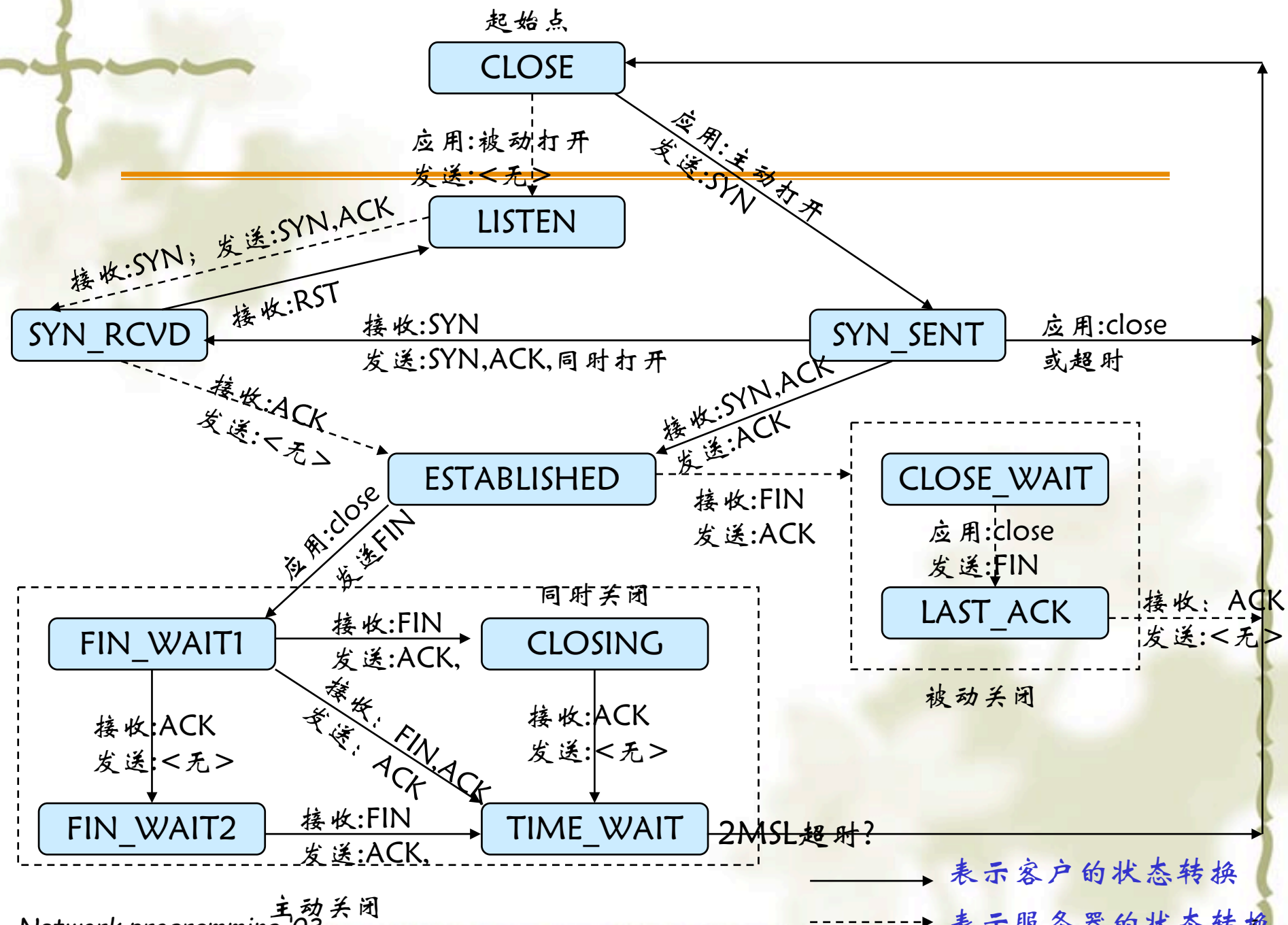
- TCP是面向连接的。
- TCP提供可靠性，实现了丢失重传。RTT的估算。
- TCP通过给所发送数据的每一个段管理一个序列号进行排序。
- TCP提供流量控制和拥塞控制：通告窗口、拥塞窗口。
- TCP的连接是全双工的。

TCP协议数据段格式（续）

- RST位：表示请求重置连接。当TCP协议接收到一个不能处理的数据段时，向对方TCP协议发送这种数据段，表示这个数据段所标识的连接出现了某种错误，请求对方TCP协议将这个连接清除。有3种情况可能导致TCP协议发送RST数据段（1）SYN数据段指定的目的端口处没有接收进程等待；（2）TCP协议想放弃一个已经存在的连接；（3）TCP接收到一个数据段，但是这个数据段所标识的连接不存在。接收到RST数据段的TCP协议立即将这条连接非正常断开，并向应用程序报告；

TCP连接的建立

- TCP连接的过程：
  - 服务器必须准备好接受外来的连接。通过调用socket, bind, listen函数完成。称为被动打开。
  - 客户通过调用connect进行主动打开。这引起客户TCP发送一个SYN分节，告诉服务器客户将在连接中发送的数据的初始序列号。
  - 服务器必须确认客户的SYN，同时自己也得发送一个SYN分节。服务器以单个分节向客户发送SYN和对客户的SYN的ACK。
  - 客户必须确认服务器的SYN。


起始点
CLOSE
应用:被动打开
发送:<无>
LISTEN
应用:主动打开
发送:SYN
接收:SYN; 发送:SYN,ACK
接收:RST
SYN_RCVD
接收:SYN
发送:SYN,ACK,同时打开
SYN_SENT
应用:close
或超时
接收:ACK
发送:<无>
接收:SYN,ACK
发送:ACK
ESTABLISHED
接收:FIN
发送:ACK
CLOSE_WAIT
应用:close
发送:FIN
LAST_ACK
接收: ACK
发送:<无>
被动关闭
应用:close
发送FIN
同时关闭
FIN_WAIT1
接收:FIN
发送:ACK,
CLOSING
接收:ACK
发送:<无>
接收:ACK
发送:<无>
接收:FIN,ACK
发送:ACK
FIN_WAIT2
接收:FIN
发送:ACK,
TIME_WAIT
2MSL超时?
主动关闭
表示客户的状态转换
表示服务器的状态转换

TCP 的TIME_WAIT状态

- 难点：执行主动关闭的那端进入这种状态。这个端点在该状态的持续时间是2MSL（最长分节生命周期）。
- 存在TIME_WAIT状态有两个理由：
  - 实现终止TCP全双工连接的可靠性（假设最后一个ack丢失的情况）.
  - 允许老的重复分节在网络中消逝.

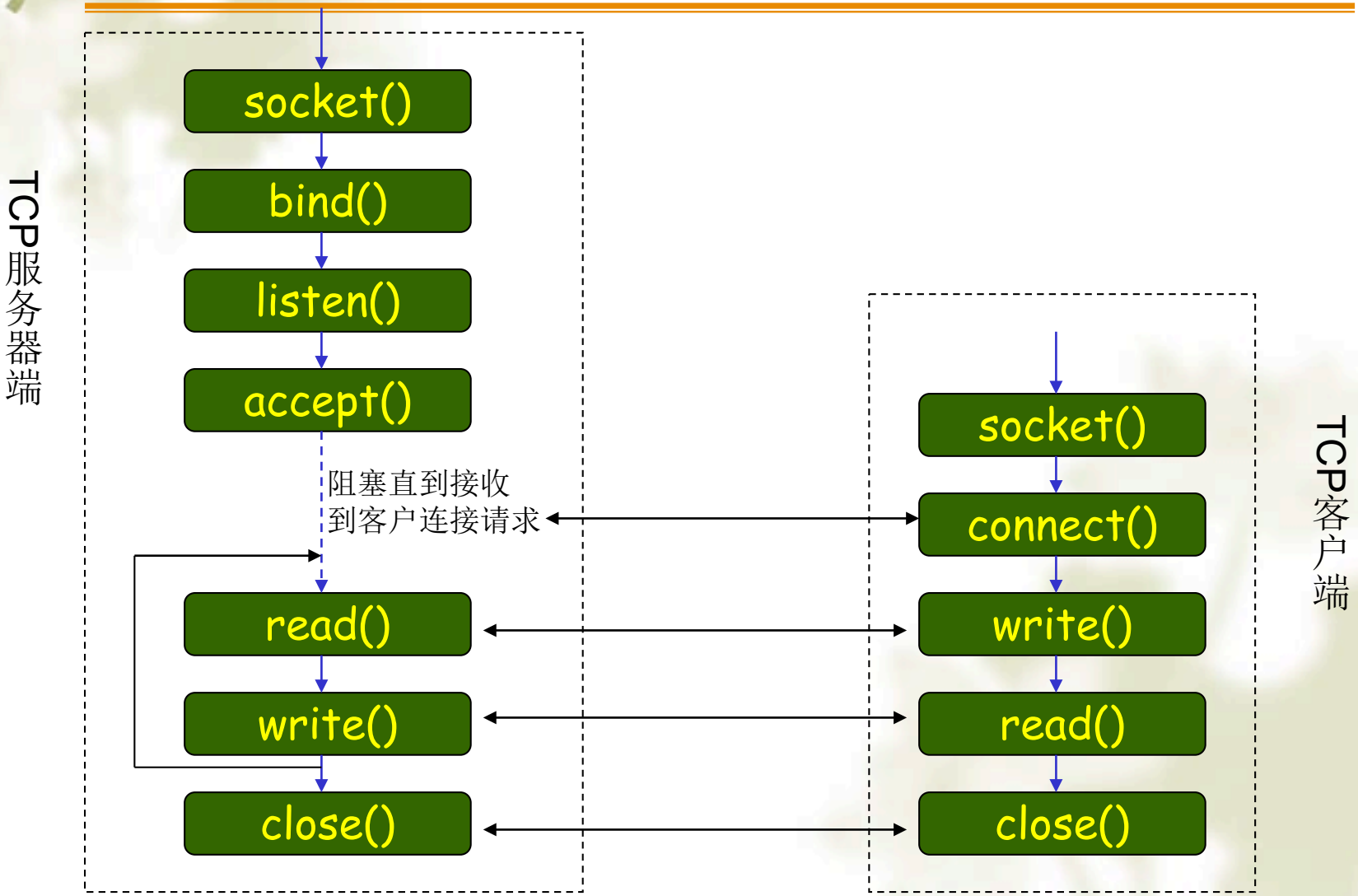
基本套接字函数－connect

```
#include <sys/socket.h>
int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

返回：0－成功；-1－出错；

- 函数connect激发TCP的三路握手过程；仅在成功或出错返回；错误有以下几种情况：
  - 如果客户没有收到SYN分节的响应（总共75秒），则返回ETIMEDOUT。
  - 如果对客户的SYN的响应是RST，则表明该服务器主机在该端口上没有进程在等待。函数返回错误ECONNREFUSED；
  - 如果客户发出的SYN在中间路由器上引发一个目的地不可达ICMP错误，则如第一种情况，连续发送SYN，直到规定时间，返回EHOSTUNREACH或ENETUNREACH。

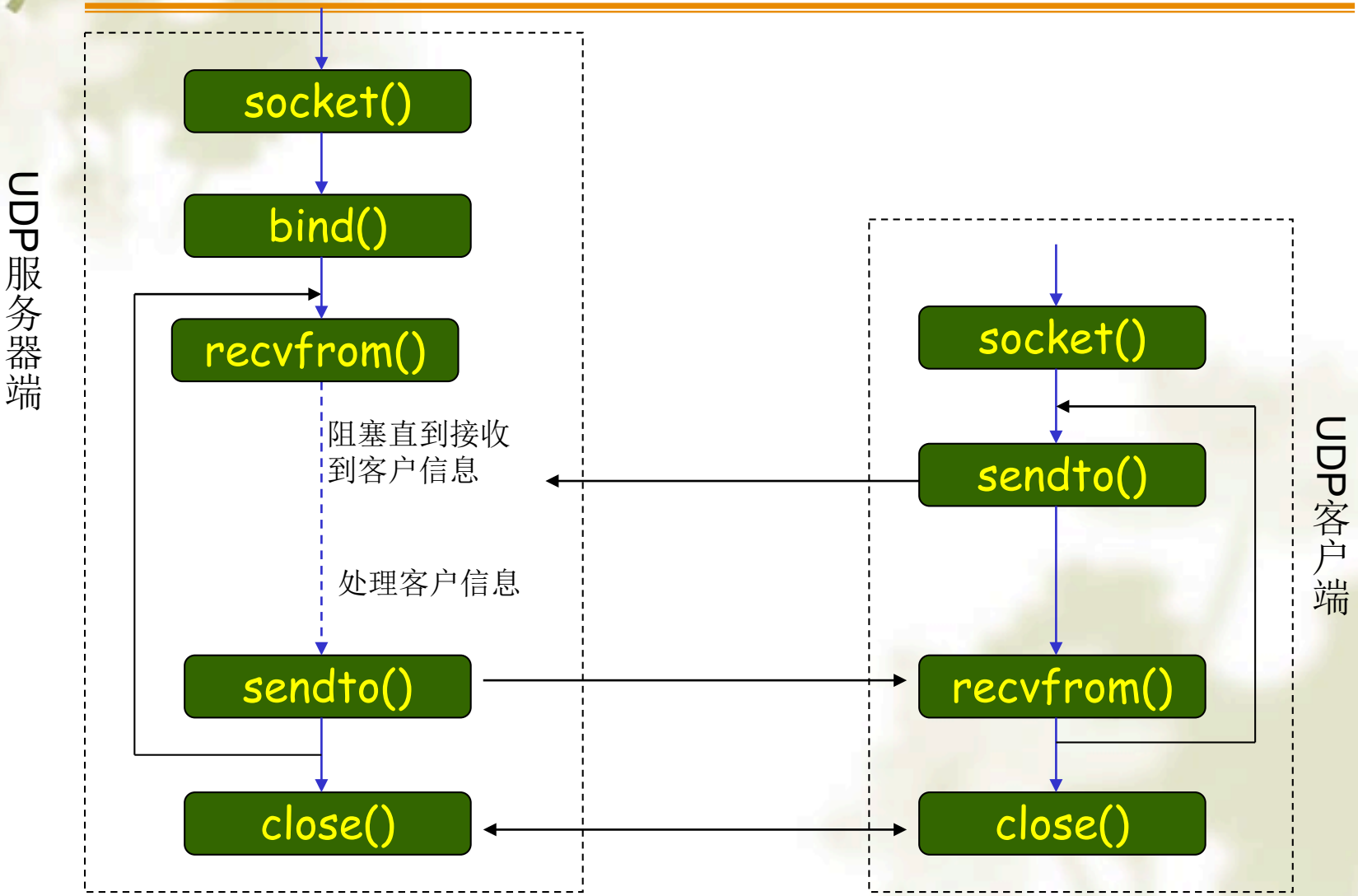
TCP套接字编程（cont.）



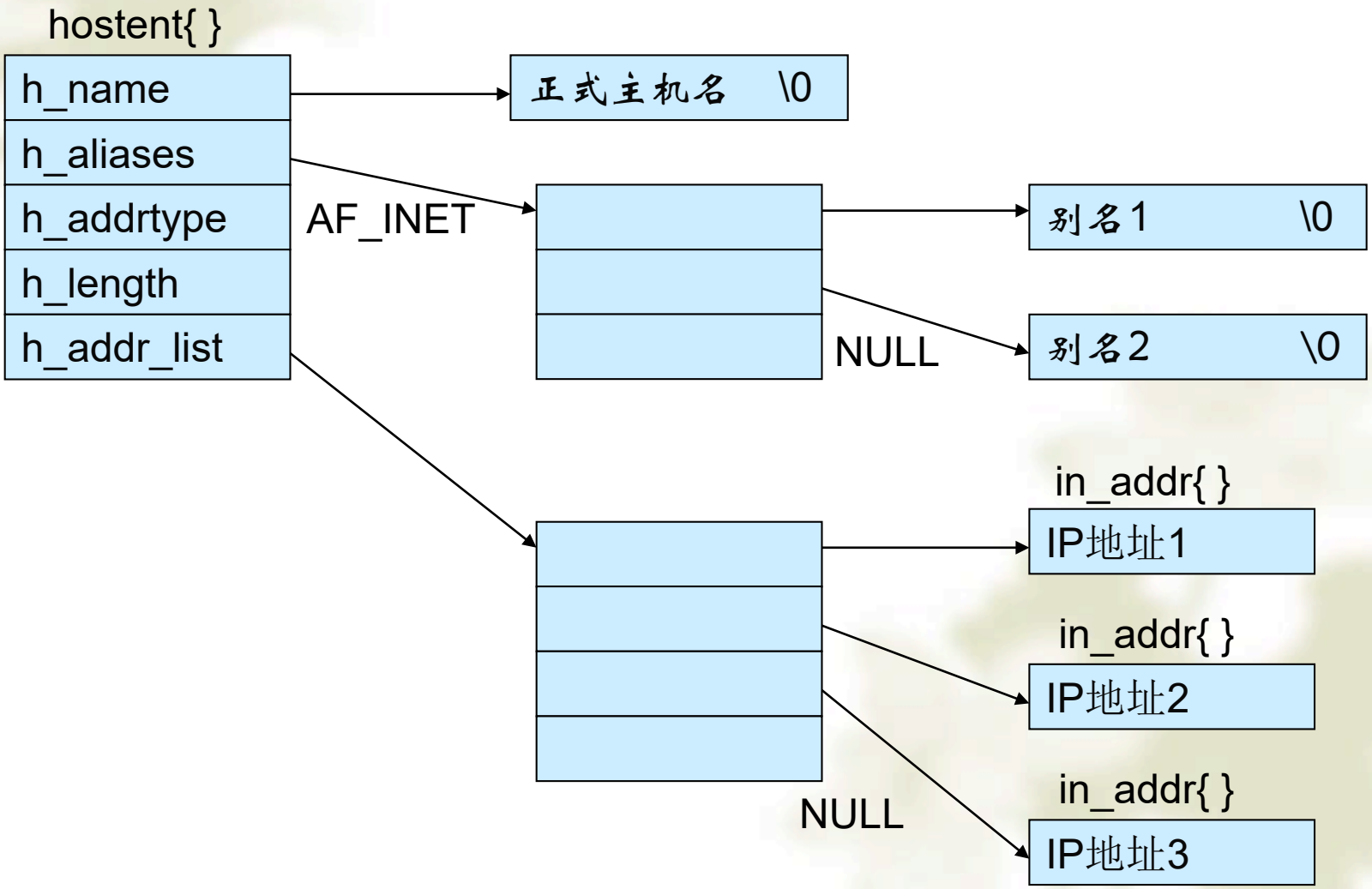
服务器主机崩溃后重启

- 在这种情况下，如果客户在主机崩溃重启前不主动发送数据，那么客户是不会知道服务器已崩溃的。在服务器重启后：
- 客户向服务器发送一个数据分节；
- 由于服务器重启后丢失了以前的连接信息（尽管在服务端口上有进程监听，但连接套接字所在的端口无进程等待），因此导致服务器主机的tcp响应RST；
- 当客户tcp收到RST，向客户返回错误，ECONNRESET
- 如果客户对服务器的崩溃很关心，即使客户不主动发送数据，就需要其他技术支持（如套接口选项SO_KEEPALIVE或某些客户／服务器心跳函数）。

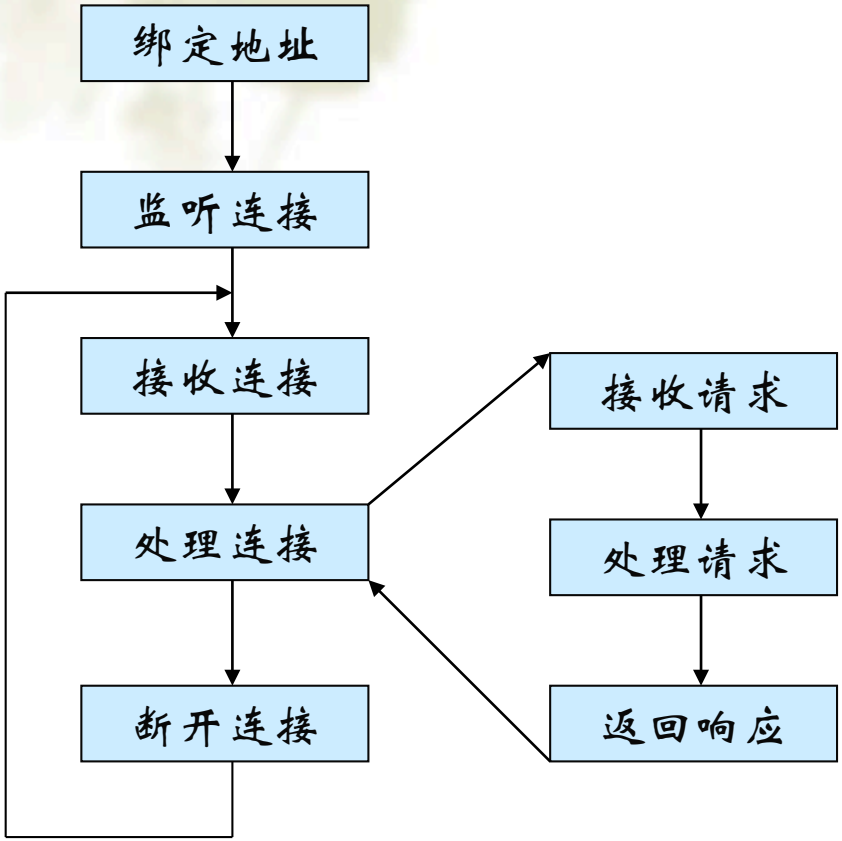
UDP套接字编程（Cont.）



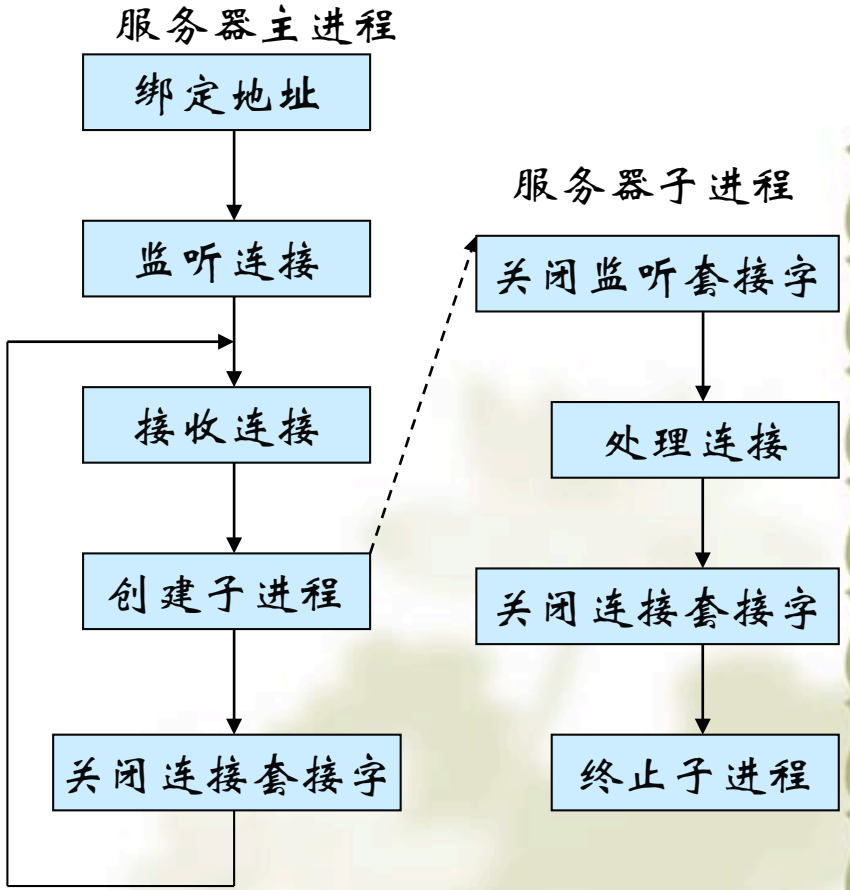
gethostbyname 返回的信息



迭代服务器 VS. 并发服务器

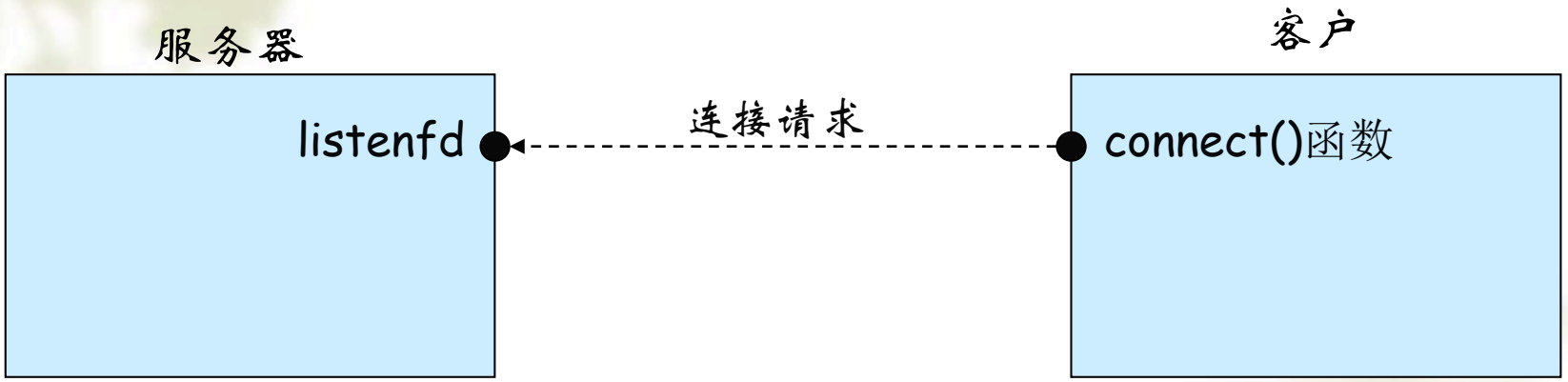


TCP迭代服务器



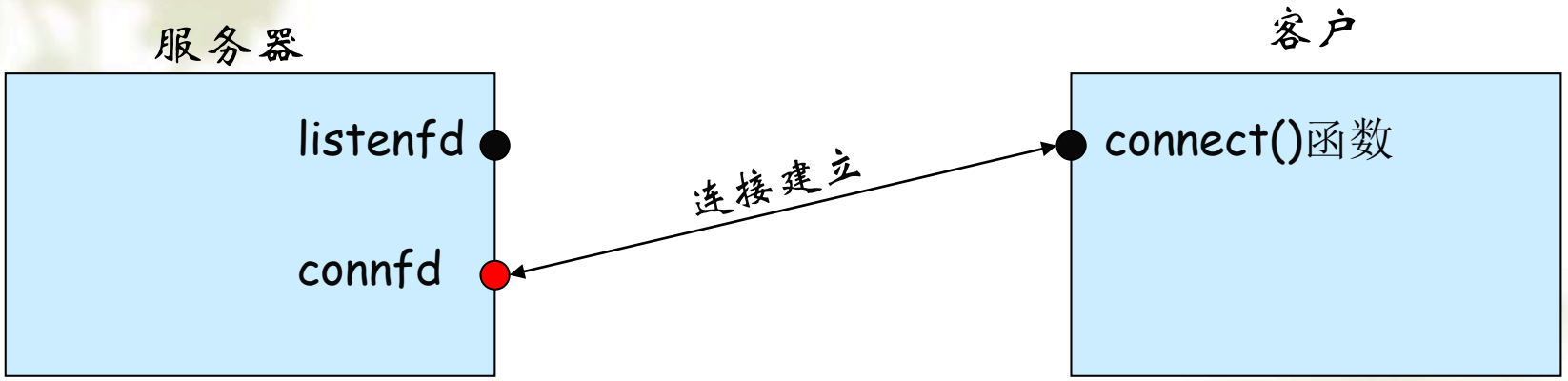
TCP并发服务器

多进程并发服务器状态图



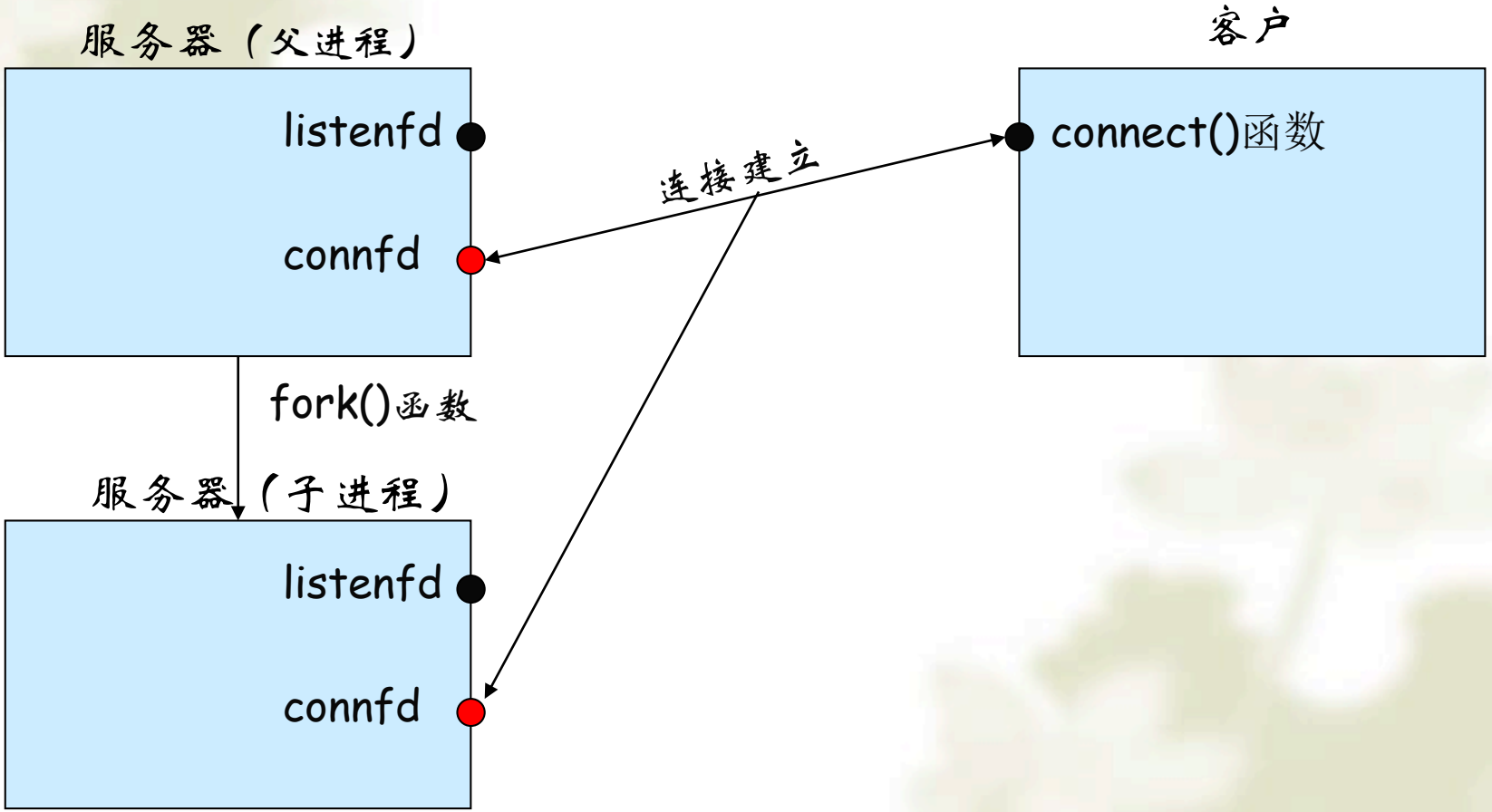
客户/服务器状态图（调用accept函数时）

多进程并发服务器状态图（cont.）



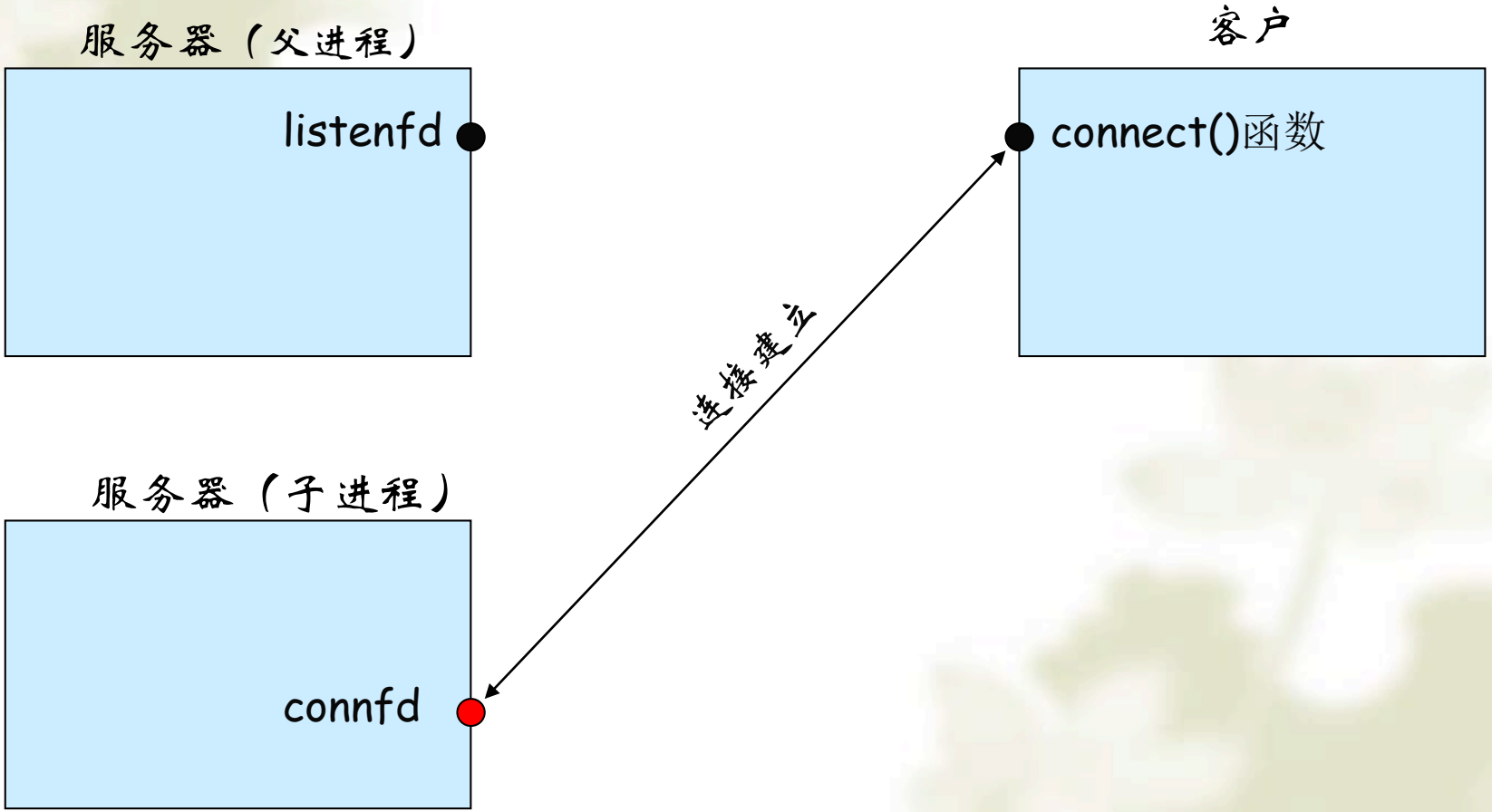
客户/服务器状态图（调用accept函数后）

多进程并发服务器状态图（cont.）



客户/服务器状态图（调用fork函数后）

多进程并发服务器状态图（cont.）



客户/服务器状态图（父进程关闭连接套接字，
子进程关闭监听套接字）

一点说明

- 从以上模板看出，产生新的子进程后，父进程要关闭连接套接字，而子进程要关闭监听套接字，主要原因是：
  - 关闭不需要的套接字可节省系统资源，同时可避免父子进程共享这些套接字可能带来的不可预计的后果；
  - 另一个更重要的原因，是为了正确地关闭连接。和文件描述符一样，每个套接字描述符都有一个“引用计数”。当fork函数返回后，listenfd和connfd的引用计数变为2，而系统只有在某描述符的“引用计数”为0时，才真正关闭该描述符。

I/O模型

- Unix下可用的i/o模型
  - 阻塞i/o
  - 非阻塞i/o
  - i/o复用（select和poll）
  - 信号驱动i/o（SIGIO）
  - 异步i/o（posix.1的aio_系列函数）
- 一个输入操作一般有两个阶段
  - 等待数据准备好
  - 从内核到进程拷贝数据

select函数（2）

- 该函数有三种执行结果：
  - 永远等待下去：仅在有一个或以上描述字准备好i/o才返回，为此，我们将timeout设置为空指针。
  - 等待固定时间：在有一个描述字准备好时返回，但不超过由timeout参数指定的秒数和微秒数。
  - 根本不等待，检查描述字后立即返回，这称为轮询。这种情况下，timeout必须指向结构timeval，且定时器的值必须为0。
- 在前两种情况的等待中，如果进程捕获了一个信号并从信号处理程序返回，那么等待一般被中断。

select函数（5）

- 套接字可读的条件主要有：
  - 套接字接收缓冲区中的数据字节数大于等于套接字接收缓冲区低潮限度的当前值；
  - 如果对方tcp发送数据，套接字就变为可读且read返回大于0的值；
  - 如果对方tcp发送一个FIN（对方进程终止），套接字就变为可读且read返回0;
  - 如果tcp发送一个RST（对方主机崩溃并重新启动），套接字就变为可读且read返回-1。
- 接收和发送低潮限度的目的是：在select返回可读或可写条件之前，应用进程可以对有多少数据可读或有多大空间可写进行控制。

select函数（6）

- 套接字可写的条件主要有：
  - 套接字发送缓冲区的可用空间大于等于套接字发送缓冲区的低潮限度；
  - 套接字的写这一半关闭，对套接字的写将产生SIGPIPE信号；
  - 有一个套接字错误待处理
- 套接字的异常条件：
  - 套接口带外数据的到达；
  - 控制状态信息的存在，可从一个已置为分组方式的伪终端读到；

SO_KEEPALIVE选项（tcp）

- 这个选项的目的是检测对方主机（或进程是否崩溃）。却省情况下，如果2小时内在此套接字上的任一方向都没有数据交换，tcp就自动给对方发一个保持存活探测分节，这是一个对方必须响应的tcp分节：会导致三种情况：
  - 对方以期望的ACK响应。应用进程不会得到通知（因为一切正常）。
  - 对方以RST响应，说明对方已崩溃且已重新启动（对方已无连接套接字信息）。此时，套接字错误被置为ECONNRESET，套接字本身被关闭；
  - 对方对保持存活探测分节无任何响应，如果连续发送9个探测分节（每75s一个）无响应，则套接字错误为ETIMEDOUT，如果收到ICMP错误消息，则返回EHOSTUNREACH。

SO_KEEPALIVE选项（续）

- 存活探测分节的间隔时间可以更改。但必须注意，大多数内核是以整个内核为基维护这些时间参数的，而不是以每个套接字为基来维护的，因此，如果将无活动周期从2小时改为15分钟，则将影响到主机上所有打开了此选项的套接字。
- 此选项一般由服务器使用。服务器使用此选项是因为它们需要知道无数据交换的客户是否还存活。否则，如果客户主机崩溃，服务器进程阻塞于等待对方tcp的客户请求，而不知道，并将亟需等待永远不会到达的请求。这称为半开连接。保持存活选项将检测出这些半开连接并终止它们；
- 大多数rlogin和telnet服务器设置此选项。有些服务器，如FTP服务器，提供一个分钟数量级的超时。这是由应用进程本身来完成的，与客户tcp套接字的这个选项无关。

高级I/O函数：套接字超时

- 有三种方法给套接字上的I/O操作设置超时：
  - 使用select阻塞在等待I/O上，select内部有一个时间限制，以此代替在read或write调用上阻塞。
  - 使用新的SO_RCVTIMEO和SO_SNDTIME套接字选项。但并不是所有的实现都支持这两个选项。
  - 调用alarm，在到达指定时间时产生SIGALRM信号。这涉及到信号处理，而且可能与进程中其他已有的alarm调用冲突；

通过原始套接字发送数据报

- 原始套接字的输出遵循以下规则：
  - 如果套接字已经连接，可以调用write、writev、send来发送数据，否则需要调用sendto或sendmsg；
  - 如果IP_HDRINCL选项未设置，则内核写的数据起始地址指IP头部之后的第一个字节。因为这种情况下，内核构造IP头部，并将它安在来自进程的数据之前。内核将IPv4头部的协议字段设置成用户在调用socket函数所给的第三个参数；
  - 如果设置了IP_HDRINCL，则内核写的数据起始地址指IP头部的第一个字节。用户所提供的数据大小值必须包括头部的字节数。此时进程构造除了以下两项外的整个IP头部；（a）IPv4标识字段可以设为0，要求内核设置该值；（b）IPv4头部校验和由内核来计算和存储。IPv4数据报首部各个字段的内容均是网络字节序（对linux而言）
  - 对于超出外出接口的MTU的分组，内核将其分片。

libpcap：分组捕获函数库

- libpcap是一个与实现无关的访问操作系统所提供的分组捕获机制的分组捕获函数库。目前它只支持分组的读取。
- 它同时支持Berkeley内核下的BPF、Solaris 2.x下的DLPI、SunOS4.1下的NIT、Linux下的SOCK_PACKET套接字以及其他若干操作系统，具有良好的兼容性。