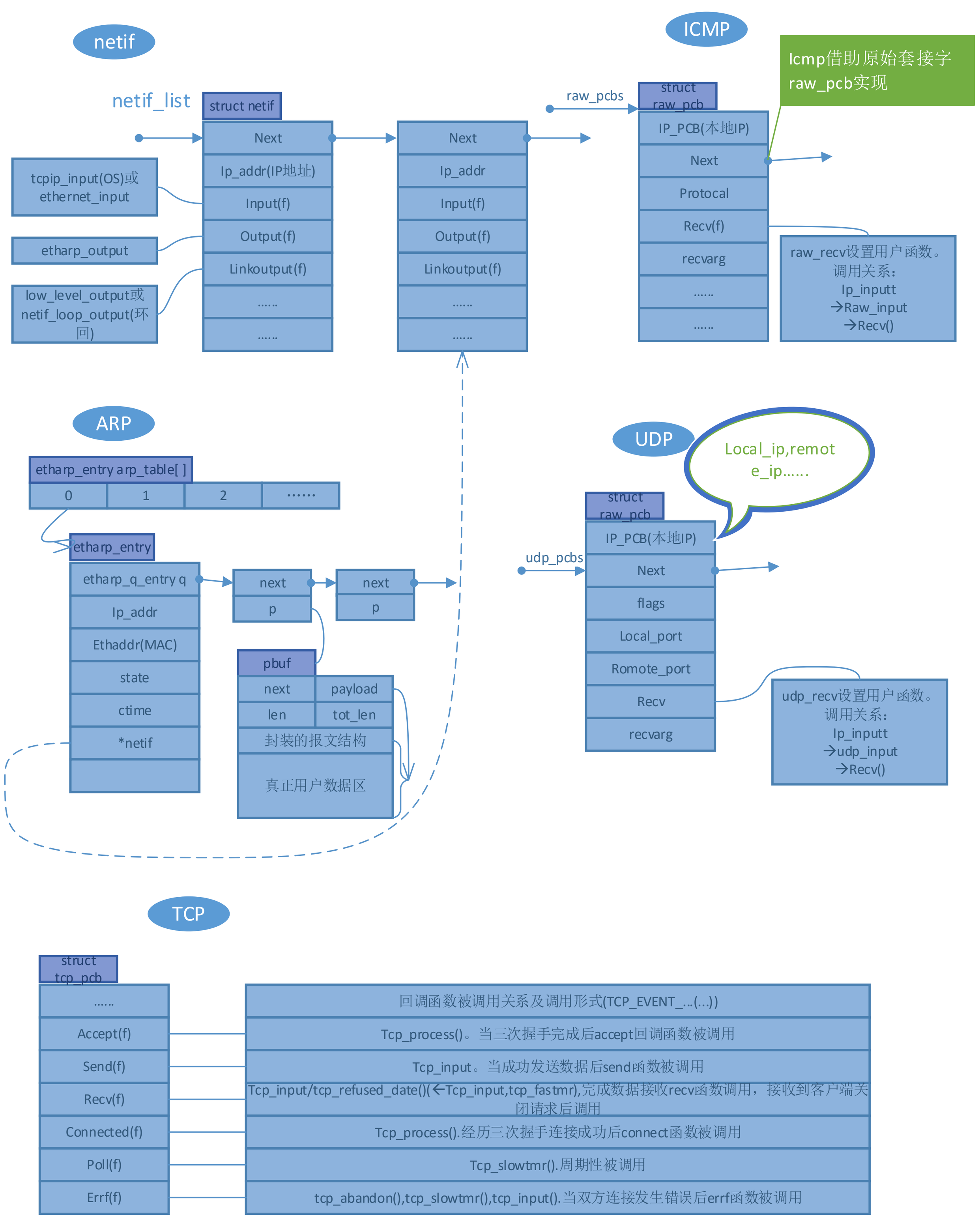
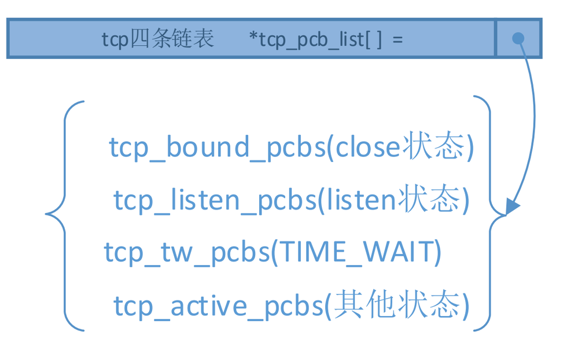
RAW API编程

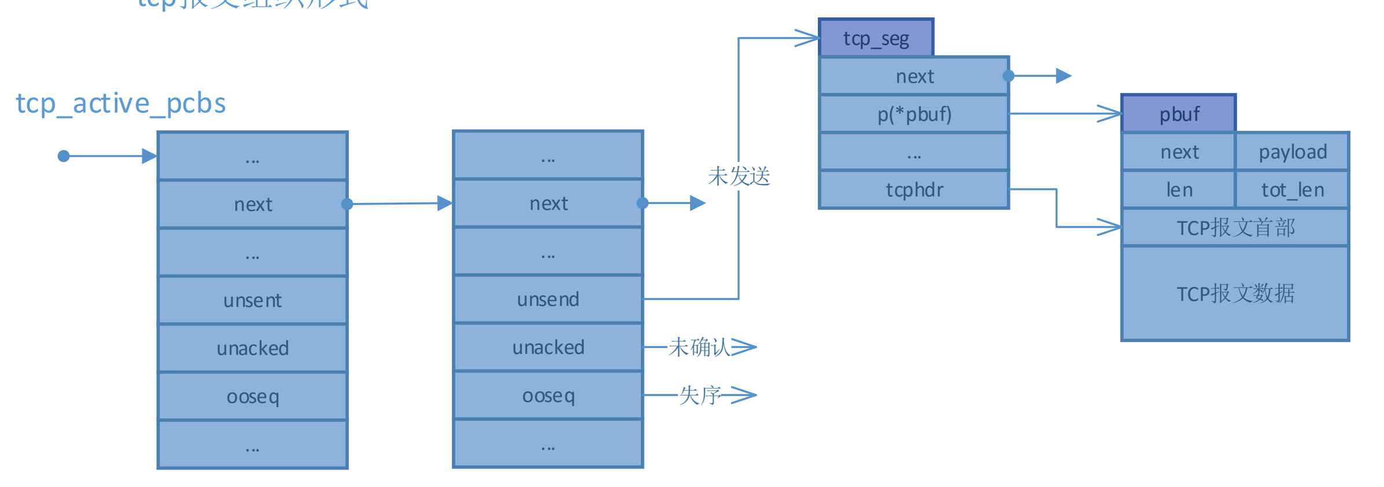


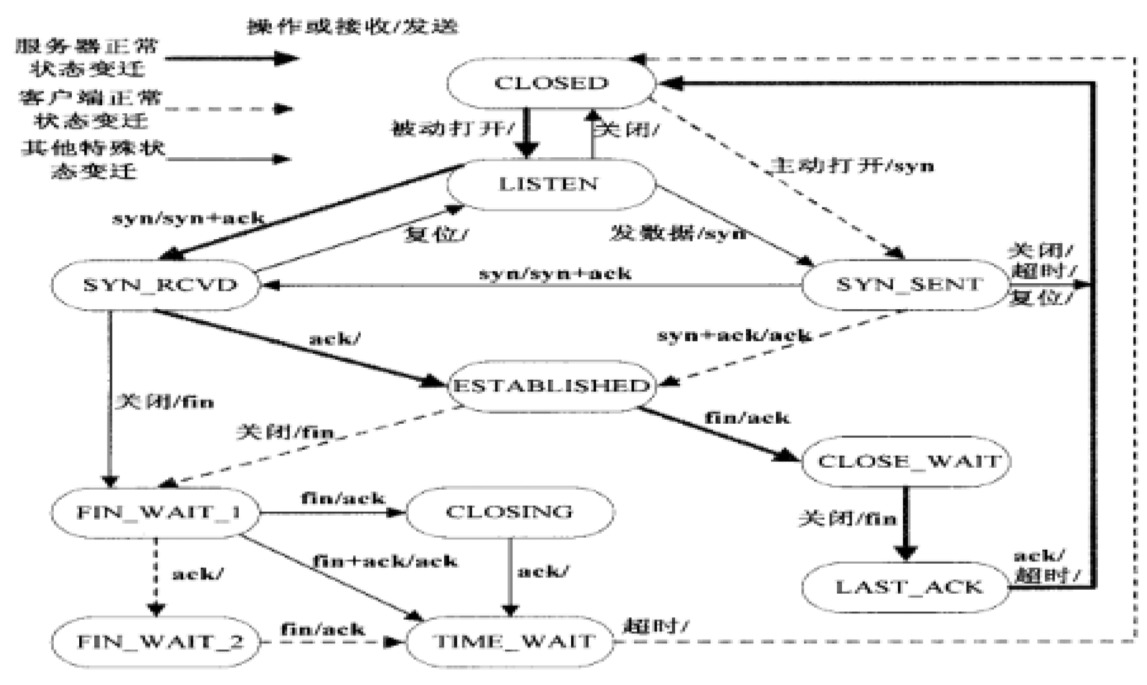


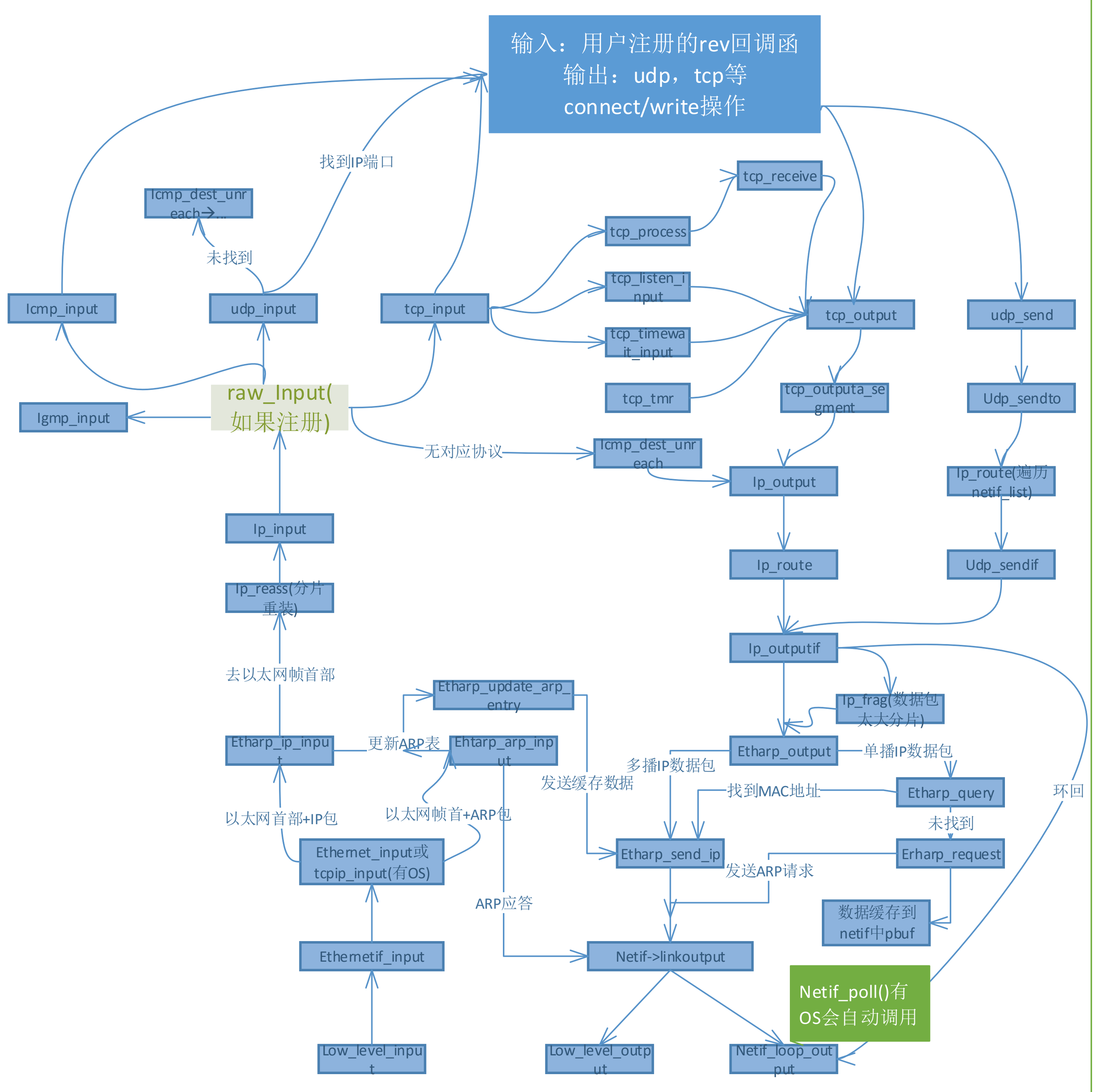
Tcp定时器与诸如ARP定时器，udp定时器不同，这些定时器初始化时开启。Tcp定时器新建一个pcb结构时注册开启，如下：

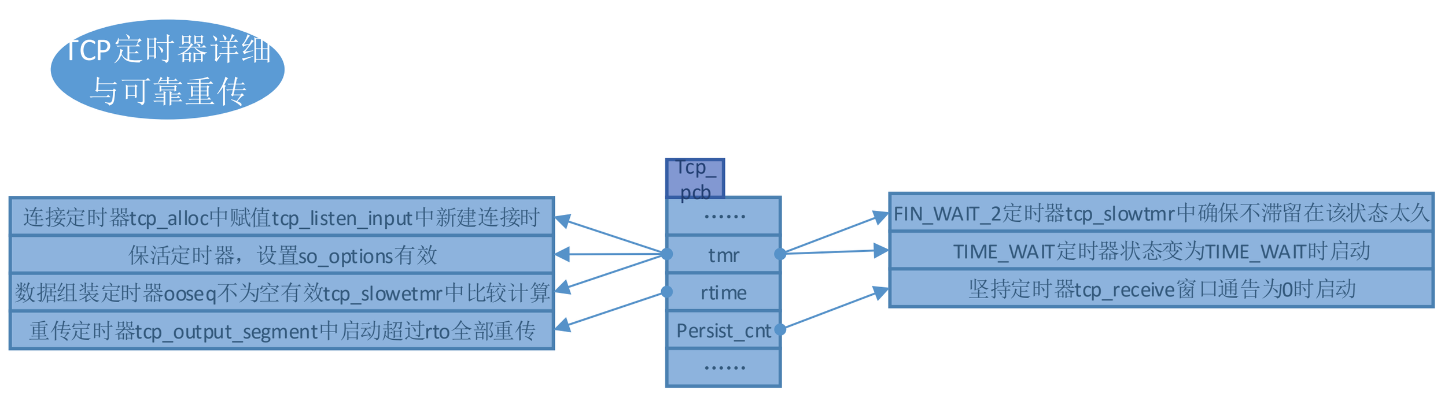
…→TCP_REG(...)→Tcpip_timer
→Tcp_tmier()
→Tcp_slowtmr()(功能性检查发送)
→Tcp_fastmr()(发送缓存的数据和延迟发送的ack)

tcp报文组织形式

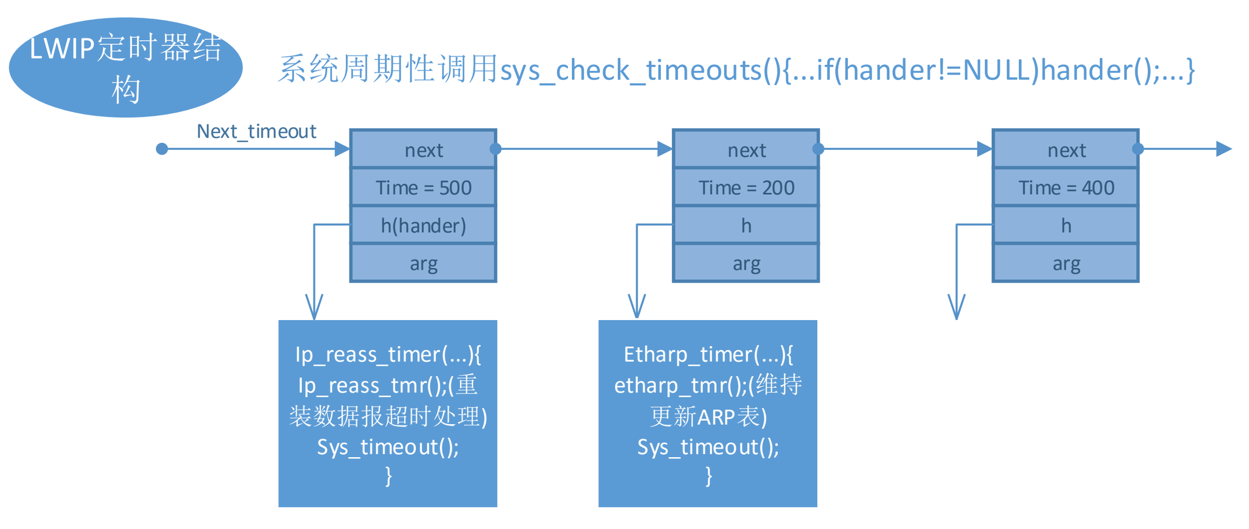






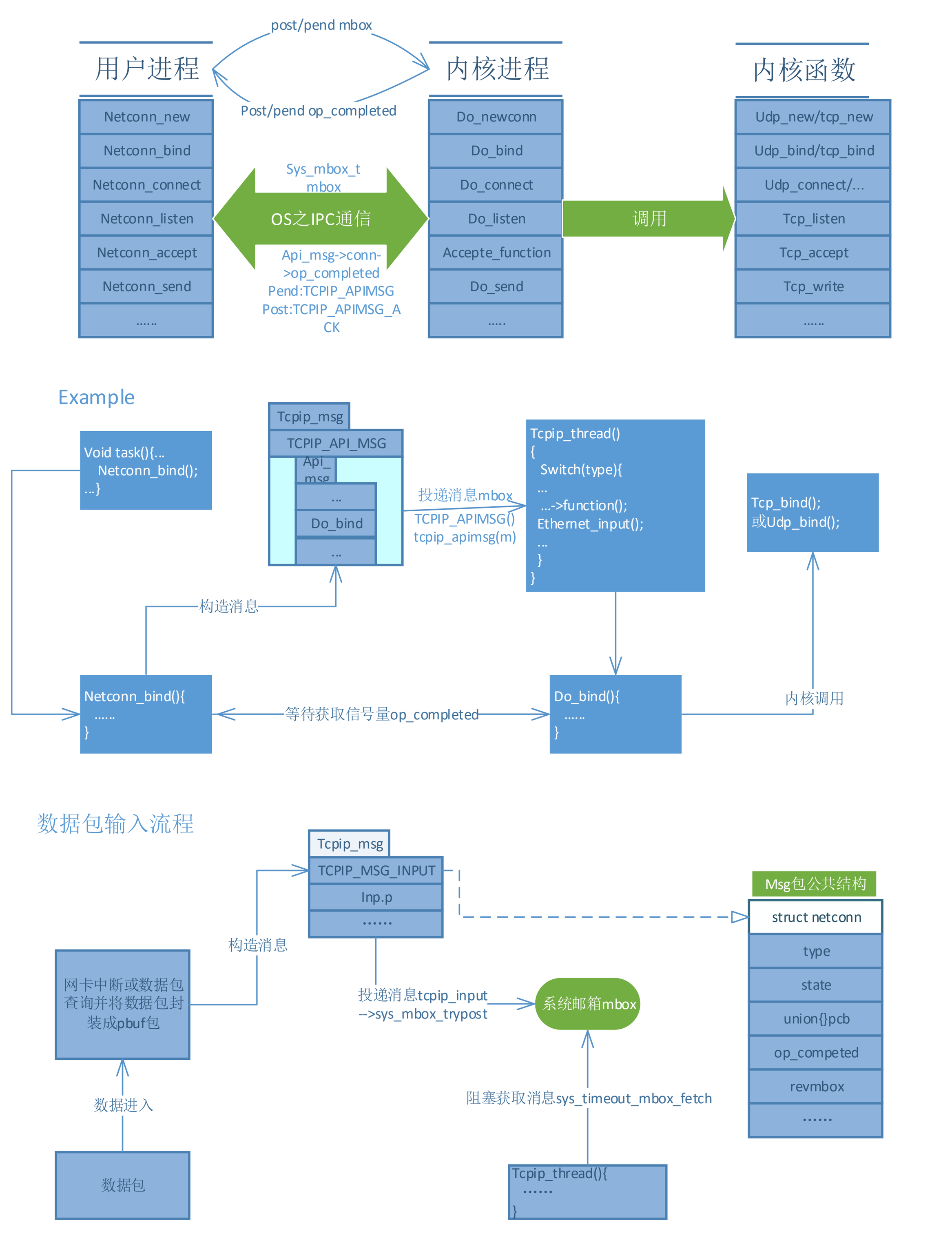


1.超时重传和RTT估计：相关函数tcp_receive,tcp_slowetmr→tcp_rexmit_rto
2.慢启动与拥塞处理：相关函数tcp_receive,tcp_listen_input
3.快速启动与恢复：相关函数tcp_receive,tcp_rexmit_fast
4.糊涂窗口避免:相关函数tcp_ack,tcp_output
5.零窗口探查：相关函数：坚持定时器和tcp_zero_window_probe
6.保活机制：保活定时器

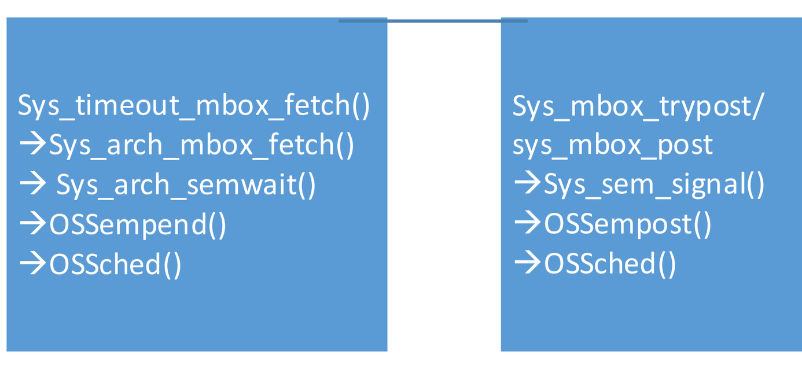


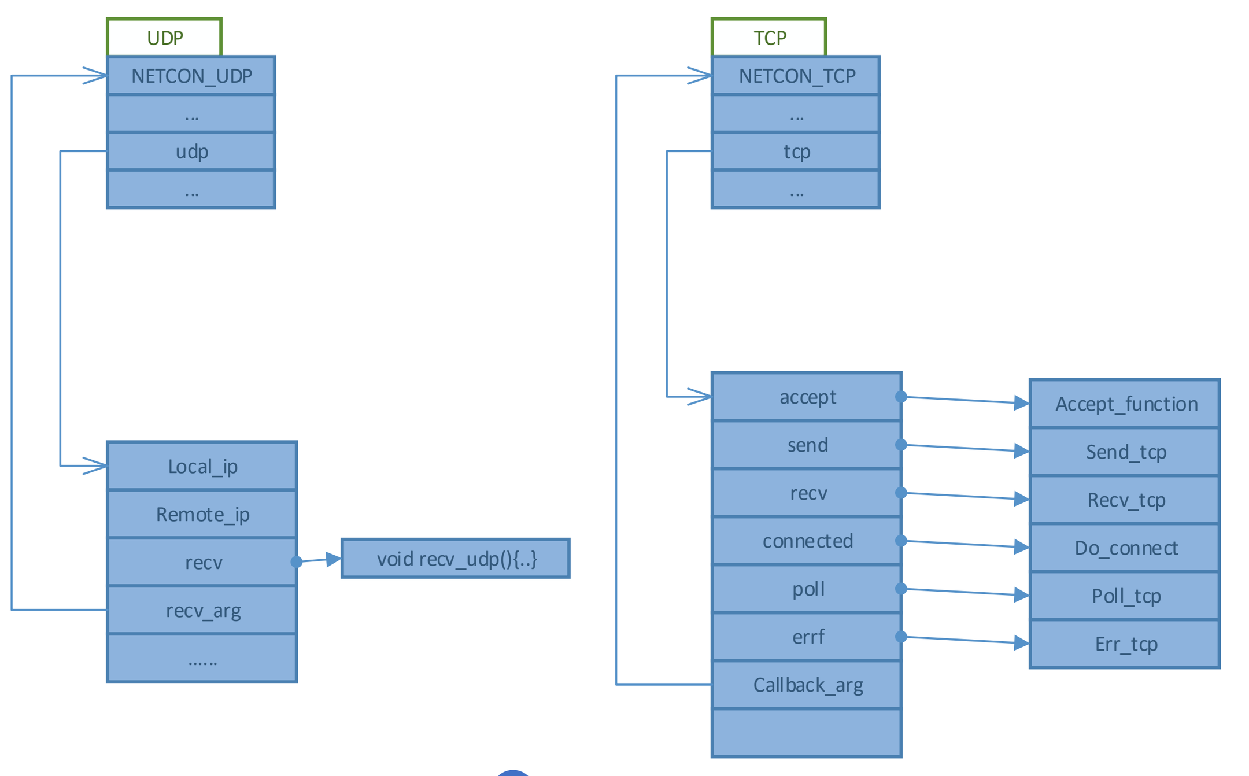
1.tcp_rst的发送及处理：
可能发送tcp_rst情况:tcp_close但数据未接收完。调用Tcp_abandon或tcp_abort。输入数据包未找到匹配的PCB块。Listen端口收到ACK。ACK错误的出现在数据窗口位置。处于SYN_SEND状态收到ACK。SYN_RCVD状态收到错误ACK Number返回号。用户已经调用tcp_close使pcb->flags = TF_RXCLOSE,但还是收到数据则发送RST报文。当tcp_listen_input收到新连接请求而分配PCB调用tcp_alloc中调用tcp_kill_timewait()或tcp_kill_prio();
接收到RST报文处理：tcp_process中确认收到合法的RST报文，则recv_flags = TF_RESET,然后由tcp_input负责清理控制块和控制块上链接的所有数据包。
2.pcb控制块回调函数recv函数写法：
err_t cb_recv(...pbuf_free(p);){if(p != NULL){...}else if(err == ERR_OK){tcp_close(pcb);}}
对else if(err == ERR_OK)的意义在于TCP_EVENT_CLOSE和TCP_EVENT_RECV公用一个回调函数，所有该回调函数要包括对接收到关闭连接事件的处理。
3.数据输出：(1)tcp_write后调用tcp_output(2)tcp_faster中调用tcp_output(3)tcp_input中输入处理完会尝试tcp_output。

Sequential模型



系统模拟层

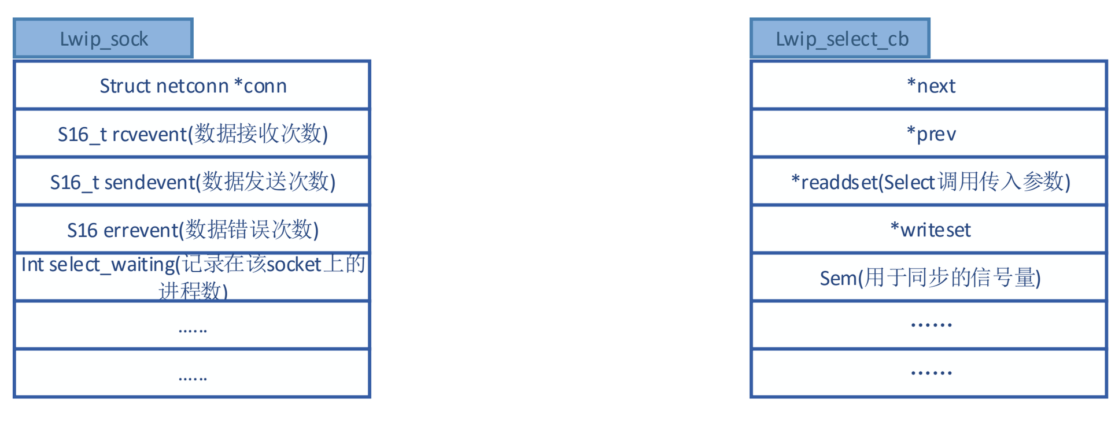




Sequential编程注意点

1.netconn_recv,如果函数返回值ERR_OK则正常处理数据，如果返回ERR_CLSD则对方关闭了连接，则用户也要复制关闭连接。别的错误亦不该忽略，对非ERR_OK的都可选择关闭。
2.netconn_close()只能关闭连接但并不会释放内存结构netconn，所以在调用netconn_close后应调用netconn_delete(newconn);

基于LWIP的Socket编程(sequential的再封装)



Struct lwip_sock [NUM_SOCKETS]可使用的socket数，和netconn数目相同
Struct lwip_select_cb *select_cb_list组织所有lwip_select_cb结构到链表上

Select原理