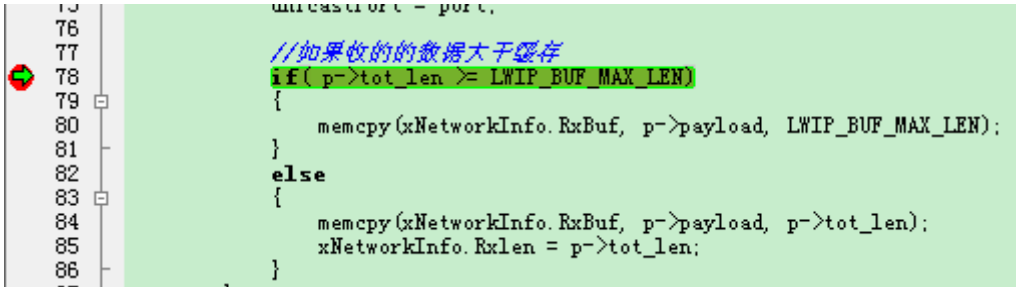
浅谈 LWIP 之 pbuf 链表数据接收问题

这几天忙里偷闲又研究了 LWIP 的包缓冲区 pbuf，发现一个一直使用的数据接收的错误，具体是在接收回调函数里简单粗暴的直接使用拷贝语句来复制数据包内容：

```
//如果收的的数据大于缓存
if( p->tot_len >= LWIP_BUF_MAX_LEN)
{
    memcpy(xNetworkInfo.RxBuf, p->payload, LWIP_BUF_MAX_LEN);
}
else
{
    memcpy(xNetworkInfo.RxBuf, p->payload, p->tot_len);
    xNetworkInfo.Rxlen = p->tot_len;
}
```

因为公司的机子一直没有碰到数据包比较大的情况，所以一直没发现上面的接收代码有问题，直到昨天才想起 pbuf 是采用链表式内存来保存网络数据的，当数据包长度大于一定值时，问题就出现了。以发送 0x00~0x4F 的 80 字节为例：

LWIP 在 UDP 回调函数里的 pbuf 链表指针 p 的 tot_len 为 80，但 len 字段只有 78，意味着 80 字节的数据不能被一个 pbuf 保存完



p	0x200018E8 (memp_m...	R4	struct pbuf *
next	0x20001970 (memp_m...	0x200018E8	struct pbuf *
payload	0x20001922 (memp_m...	0x200018EC	void *
tot_len	80	0x200018F0	u16_t
len	78	0x200018F2	u16_t
type	'.' (0x03)	0x200018F4	u8_t
flags	'\0' (0x00)	0x200018F5	u8_t
ref	1	0x200018F6	u16_t

这时如果直接采用 memcpy 来拷贝的话，最后的 2 字节无疑是错误的！下面给出了数据读取后数组最后 2 字节的错误内容：

```
76
77              //如果收的的数据大于缓存
78              if( p->tot_len >= LWIP_BUF_MAX_LEN)
79              {
80                  memcpy(xNetworkInfo.RxBuf, p->payload, LWIP_BUF_MAX_LEN);
81              }
82              else
83              {
84                  memcpy(xNetworkInfo.RxBuf, p->payload, p->tot_len);
85                  xNetworkInfo.Rxlen = p->tot_len;
86              }
87          }
```

[71]	0x47
[72]	0x48
[73]	0x49
[74]	0x4A
[75]	0x4B
[76]	0x4C
[77]	0x4D
[78]	0x00
[79]	0x00
[80]	0x00
[81]	0x00

再深入分析下这个问题，由于 LWIP 的数据接收采用的是 pbuf 链表，当数据包比较大时，可以认为 LWIP 将数据包分块然后连成一个链表，最后 LWIP 把这个链表的链表头传递到回调函数，这时用户如果想获取这个链表的内容，就必须读取链表的每个 payload 才能确保数据正确！下面贴上本渣自己写的代码：

```
/**
  ********************************************************************************
  * 函数名称 : Get_PBUF_Payload()
  * 函数功能 : pbuf链表数据提取
  * 入口参数 : p : 数据包pbuf链表
  *            pBuffer : 数据接收缓冲区
  *            maxLen : 最大接收长度
  * 返 回 值 : 实际接收长度
  ********************************************************************************
  */
static uint32_t Get_PBUF_Payload(struct pbuf* p, uint8_t* pBuffer, uint32_t maxLen)
{
    uint32_t remainLen;
    uint32_t readLen = 0;

    /* 遍历链表 */
    for(; p != NULL; p = p->next)
    {
        remainLen = maxLen - readLen;   // 获取缓冲区剩余长度

        /* 如果当前pbuf的数据块长度比缓冲区的剩余长度大 */
        if(p->len >= remainLen)
        {
            /* 只拷贝剩余长度大小的数据，并退出链表遍历操作 */
            memcpy(&pBuffer[readLen], p->payload, remainLen);
            readLen += remainLen;
            break;
        }

        /* 拷贝当前pbuf的数据块到缓冲区 */
        memcpy(&pBuffer[readLen], p->payload, p->len);
        readLen += p->len;
    }
    return readLen;
}
```

当调用上面的函数接收后，接收数据正确了：

```
83              /* 读取接收到的数据 */
84              xNetworkInfo.Rxlen = Get_PBUF_Payload(p, xNetworkInfo.RxBuf, LWIP_BUF_MAX_LEN);
85          }
86          pbuf_free(p);
```

[70]	0x46
[71]	0x47
[72]	0x48
[73]	0x49
[74]	0x4A
[75]	0x4B
[76]	0x4C
[77]	0x4D
[78]	0x4E
[79]	0x4F
[80]	0x00
[81]	0x00
[82]	0x00

上面的实验只是试验了数据包没有超出缓冲区的情况，下面以 512 字节来试验缓冲区大小为 256 的情况：

```
#define LWIP_BUF_MAX_LEN    256
```

进入到 Get_PBUF_Payload()函数后可以看到当前的 pbuf 链表总长度为 512：

```
34  static uint32_t Get_PBUF_Payload(struct pbuf* p, uint8_t* pBuffer, uint32_t maxLen)
35  {
36      uint32_t remainLen;
37      uint32_t readLen = 0;
38
39      /* 遍历链表 */
```

p	0x200018E8 (memp_m...	R6	struct pbuf *
next	0x200019F8 (memp_m...	0x200018E8	struct pbuf *
payload	0x20001922 (memp_m...	0x200018EC	void *
tot_len	512	0x200018F0	u16_t
len	78	0x200018F2	u16_t
type	'.' (0x03)	0x200018F4	u8_t
flags	'\0' (0x00)	0x200018F5	u8_t
ref	1	0x200018F6	u16_t

由于传递进来的 maxLen = 256，因此函数返回的成功读取的数据也是 256 而非 512！

```
56      }
57      return readLen;
58  }
```

readLen	256	R4	uint32_t

后来翻了下原子 F7 的例程，思路跟本渣的大致一样，不过原子兄的这个代码无法获取实际读取的数据长度：

```
for(q=p;q!=NULL;q=q->next)  //遍历完整个pbuf链表
{
    //判断要拷贝到UDP_DEMO_RX_BUFSIZE中的数据是否大于UDP_DEMO_RX_BUFSIZE的剩余空间，如果大于
    //的话就只拷贝UDP_DEMO_RX_BUFSIZE中剩余长度的数据，否则的话就拷贝所有的数据
    if(q->len > (UDP_DEMO_RX_BUFSIZE-data_len))
        memcpy(udp_demo_recvbuf+data_len,q->payload,(UDP_DEMO_RX_BUFSIZE-data_len));//拷贝数据
    else
        memcpy(udp_demo_recvbuf+data_len,q->payload,q->len);
    data_len += q->len;
    if(data_len > UDP_DEMO_RX_BUFSIZE) break; //超出TCP客户端接收数组,跳出
}
```

更多 LWIP 的 pbuf 知识请读者自行查阅《Design and Implementation of the lwIP TCP_IP Stack》，当然本渣是不会告诉你这本资料有中文翻译版本的。最后推荐下五木大神关于 LWIP 的见解《LwIP 协议栈源码详解》，资料问度娘。