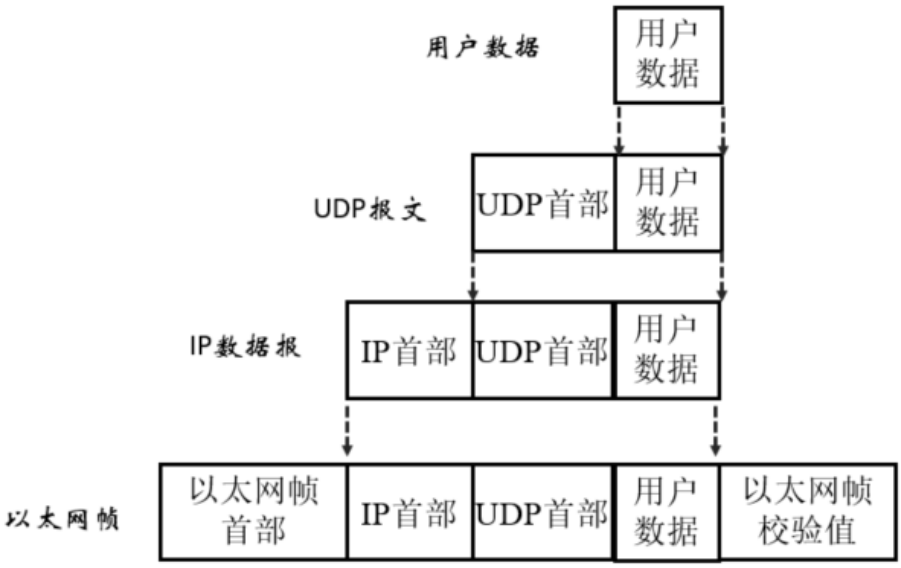
Lwip 中和 IP 分包相关的参数

前言

前不久接到一个客户的问题。在 H743 上需要通过 UDP 发送大的数据包，涉及到 IP 分包的问题。他们在测试的过程中遇到了只要发送 6KB 的 UDP 数据包就会出现 hardfault 的问题。拿到这个问题的时候，调试得到了和客户一样的现象，程序进入 hardfault，并且是由 Lwip 协议栈的 ip_reass_free_complete_datagram 函数触发。后经过一番调试，搞清楚了问题的原因，要说清楚，我们得先来看看 Lwip 中 IP 分包的实现。

IP 分包

因为以太网对一次传输数据的最大长度规定为 1500 字节，所以如果 UDP 的数据包大于这个长度，就会在 IP 层进行分包后，再通过以太网传输。用户数据通过 UDP 发送，数据封装的过程如下图：



IP 首部一般是 20 字节，UDP 首部 8 字节，1500-20-8=1472，所以当发送的用户数据超过 1472 个字节后，IP 层就会分包。分包和重组的工作都是在 IP 层自动完成，对于 UDP 层来说，不用关心这个过程。

Lwip 中 IP 数据重组的实现

Lwip 中通过 ip_reassdata 数据结构来描述一份正在被重组的 IP 数据报，其中的 p 字段指向第一个数据分片。Lwip 中可以同时处理多个 IP 数据报的重组，每个 IP 数据报对应一个 ip_reassdata 数据结构，并且通过结构中的 next 指针构成一个单向的链表。reassdatagrams，指向 ip_reassdata 结构组成的链表的开头。

当以太网底层驱动接收到数据后，将数据传递给 Lwip 协议栈，ip_input 函数会根据当前是 IPv4 还是 IPV6 调用对应的处理函数。在 ip4_input 中，通过检查 IP 首部中的标志位和分片偏移量，来判断当前是否是一个 IP 分片包。如果是就调用 ip4_reass 进行重组。

在整个 IP 数据报重组完成之前，接收到的 IP 分片包都保存在 buffer 中，如果其中的某个分片丢失，则重组就无法完成，而这些不全的 IP 分包数据不能一直占着 buffer，所以在 Lwip 中定义了 ip_reassdata 的生存时间，每 1 秒执行一次 ip_reass_tmr 函数，将生存时间减 1，当减到 0 后就删除对应的 ip_reassdata 数据结构，以及其上挂的所有数据分片 pbuf。

Lwip 中 IP 数据分片的实现

发送的过程比接收的过程相对简单，大致就是：UDP 层将要发送的数据组装在一个 pbuf 中，然后调用 ip_output 发送数据，当然在这个过程中，协议栈会添加相应的 IP 首部数据。如果 IP 数据报的长度大于以太网的 MTU 时，Lwip 协议栈的 IP 层就会进行分片，将数据报分成两个或者更多进行发送。IP 分片的工作是在 ip_frag 函数中完成的。

代码的实现在这里就不详细展开介绍了。我们这里主要讲的是，我们在应用 Lwip 的时候，哪些配置参数是需要注意的。

Lwip 的相关配置参数

opt.h 里有所有的 Lwip 默认配置，另外 lwipopts.h 中是应用程序中对 lwip 协议栈的配置。协议栈首先会去检查 lwipopts.h 中的参数，没有定义的再去检查 opt.h。所以 lwipopts.h 中的配置会覆盖 opt.h 中的配置。这里我们只看和 IP 分片重组相关的部分。

IP_REASSEMBLY 和 IP_FRAG：如果需要支持 IP 重组和分片功能，这两个宏一定要设置为 1。

IP_REASS_MAXAGE：IP 分片包的生存时间，超过这个时间还没有收到所有的 IP 分片，则重组失败，已经收到的 IP 分片也将从协议栈中删除。以秒为单位。

MEMP_NUM_REASSDATA：可以同时进行 IP 重组的 IP 数据报的个数。这个数值是指整个 IP 数据报的个数，不是指 IP 分片的个数。

IP_REASS_MAX_PBUFS：指在 ip_reassdata 链表中挂接的，等待重组的 pbuf 的总个数。

MEMP_NUM_FRAG_PBUF：可同时发送的 IP 分片个数。

MEM_SIZE : heap 大小，发送的数据越大，这个 size 就需要越大。

PBUF_POOL_SIZE：pbuf pool 中 buffer 的个数，跟接收数据的大小有关。

PBUF_POOL_BUFSIZE：pbuf pool 中每个 buffer 的大小，跟接收数据的大小有关。

问题分析

我们再回到一开始遇到的问题。发送 6KB 的 UDP 数据，通过 wireshark 抓包可以看到，这 6KB 的数据被分成了 5 个 IP 分片发出来。但程序中只设置了 4 个 ETH_RX_Buffer，每个 buffer 有 1536 个字节，虽然 4 个 buffer 加起来有 6144 字节，看起来刚好够我们发送的 6KB。但我们要知道，STM32MAC 层在接收数据的时候，一个以太网帧的数据可以放在多个 ETH_RX_Buffer 中，但一个 ETH_RX_Buffer 不能放多个帧的数据。简单点说，就是 MAC 层即使只收到 1 个字节的数据，在当前的配置下，它也会占用掉一个 1536 字节的 buffer。那现在 PC 端发来了 5 个 IP 分片，分别在 5 个以太网帧中，现在问题就很清楚了，因为我们只设置了 4 个 ETH_RX_Buffer，所以轮到第 5 个以太网帧的数据过来的时候，ETH_RX_Buffer 已经被占完了，没有空余的 buffer 来接收第 5 个以太网帧。

这里就不得不提到，当前 STM32Cube_FW_H7_V1.6.0 的以太网底层驱动还有一个 bug。在 H7 的驱动中，从底层驱动到 lwip 协议栈，使用了“零拷贝”的机制，也就是说，ETH_RX_Buffer 的地址直接传递给 lwip 协议栈进行数据处理，不会再进行数据的拷贝。但是，当前的驱动中，在 low_level_input 函数中，没有等 lwip 协议栈处理完 ETH_RX_Buffer 中的数据，接收 descriptor 立刻就被还给 ETH DMA 了。这样当连续收到大量数据的时候，就会发生后面的数据把前面未处理完的数据覆盖掉的情况（ST 官方已经在修改这个问题，在后续版本中会更新，当前可以通过增加 ETH_RX_Buffer 来解决这个问题）。

```
static struct pbuf * low_level_input(struct netif *netif)
{
  struct pbuf *p = NULL;
  ETH_BufferTypeDef RxBuff;
  uint32_t framelength = 0;
  struct pbuf_custom* custom_pbuf;

  if (HAL_ETH_IsRxDataAvailable(&heth))
  {
    HAL_ETH_GetRxDataBuffer(&heth, &RxBuff);
    HAL_ETH_GetRxDataLength(&heth, &framelength);

    /* Build Rx descriptor to be ready for next data reception */
    HAL_ETH_BuildRxDescriptors(&heth);

#if !defined(DUAL_CORE) || defined(CORE_CM7)
    /* Invalidate data cache for ETH Rx Buffers */
    SCB_InvalidateDCache_by_Addr((uint32_t *)RxBuff.buffer, framelength);
#endif

    custom_pbuf  = (struct pbuf_custom*)LWIP_MEMPOOL_ALLOC(RX_POOL);
```

好，我们再回到前面的问题上，MAC 接收到第 5 个以太网帧后，由于 H7 底层驱动的 bug,就会将第一个 ETH_RX_Buffer 中的数据覆盖掉，这样导致 lwip 协议栈在处理时，因为收到了不完整，且被破坏的数据，出现了 hardfault。如果将 ETH_RX_DESC_CNT 和对应 ETH_RX_Buffer 个数增加，就可以解决这个问题。当然 MPU 配置中，Descriptor 对应的 region 范围也要调整大小，这里就不详说。

尾声

这个问题，如果继续增加发送的数据大小，就会发现还会遇到其他问题。随着数据增大，IP 分片也增多，这样在 ip_reassdata 中挂接的 pbuf 也增多，所以还需要关注 IP_REASS_MAX_PBUFS 的大小，相应进行调整。

而在发送大的数据出现 IP 分片的时候，就需要关注 MEMP_NUM_FRAG_PBUF，MEM_SIZE 以及发送 descriptor 的个数了。

重要通知 – 请仔细阅读

意法半导体公司及其子公司（“ST”）保留随时对ST 产品和/ 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于ST 产品的最新信息。ST 产品的销售依照订单确认时的相关ST 销售条款。

买方自行负责对ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的ST 产品如有不同于此处提供的信息的规定，将导致ST 针对该产品授予的任何保证失效。

ST 和ST 徽标是ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。