# LwIP 内存配置

## 前言

LwIP 在 lwipopts.h 和 opt.h 头文件中提供了多个配置选项。用户可以根据不同的性能需求和不同应用的内存限制对协议栈用到的内存配置进行调节。Opt.h 头文件中包括协议使能和设置，内存设置，调试选项等等。而 Lwipopts.h 头文件中集合了 opt.h 中一些常常需要改动的部分。一般情况下用户对 lwipopts.h 头文件进行修改就可以了。不管是对 lwipopts.h 还是 opt.h 进行修改，都必须保证是在已经对你所改动的内容足够的了解的情况下进行，所做的改动是正确的，否则有可能导致协议栈不能正常工作，或者效率低下。

## LwIP 的内存管理机制

在进行内存配置之前，我们有必要先了解 LwIP 的内存管理机制。

### Lwip 动态内存管理

LwIP 中可以使用两种动态内存分配的方法：Heap 和 Pool 的方式。

Heap 的方式，每次都根据你实际需要的大小分配一块内存出来用，用完以后再还回去。

Pool 的方式则是，预先将内存等分成若干份，每次分配时都拿出其中的一块或几块来。假设每等份是 256bytes，而你需要 300bytes 的内存空间，Pool 的方式就会给你分配两个 256bytes 的内存块（一共 512bytes）。虽然有点浪费，但这种方式分配内存速度很快，非常适合在接收数据时使用。

对于 Heap 的方式，程序默认是使用 LwIP 提供的 mem_malloc/mem_free 进行内存的分配和释放。这种方式下，程序需要预先分配一段内存空间用来做 heap 分配，这段预留的空间大小通过 MEM_SIZE 定义。

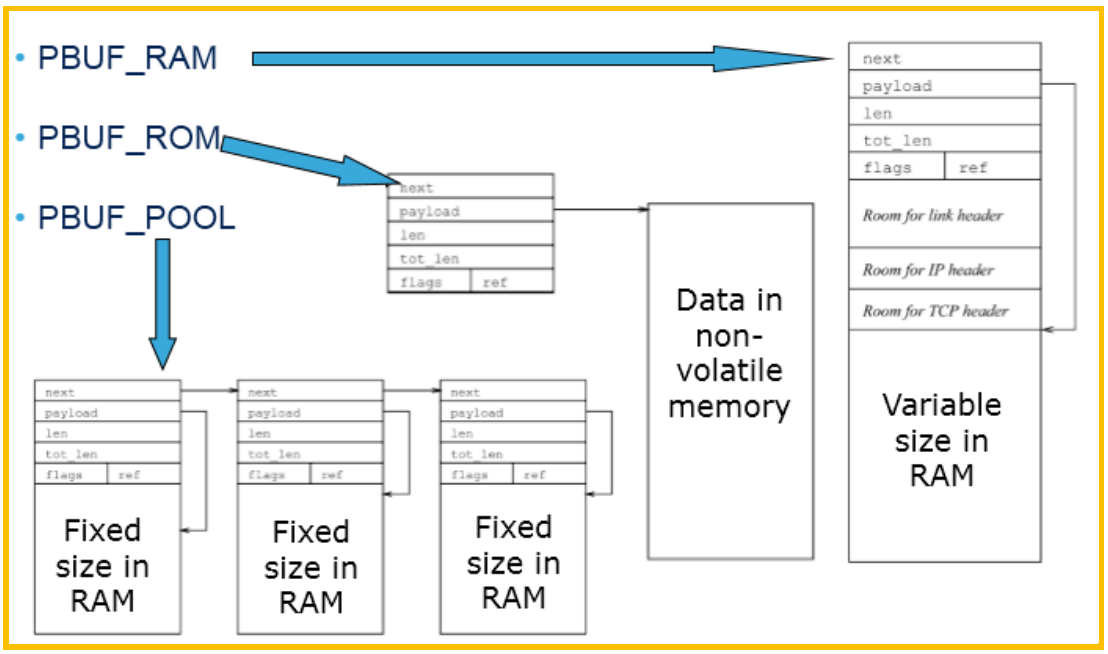
你也可以通过 C 标准库里的 malloc/free 函数进行内存的分配和释放。需要定义宏 MEM_LIBC_MALLOC。

### Pbuf 类型

前面说的 Heap 和 Pool 都是 LwIP 动态分配内存的方式。而 LwIP 采用了 pbuf 的形式管理内存中的信息，pbuf 结构既支持动态内存分配保存信息包内容，也支持让信息包数据驻留在静态存储区。pbufs 可以在一个链表中链接在一起，被称作一个 pbuf 链，这样一个信息包可以跨越几个 pbufs。

LwIP 有三种类型的 pbuf: PBUF_RAM, PBUF_ROM,PBUF_POOL。这三种类型拥有不同的使用目的。

- PBUF_RAM 类型的 pbuf 用于应用程序发送的数据被动态生成的情况。在这种情况下，pbuf 系统不仅为应用数据分配内存，还要给为这些数据预置的包头分配内存。包头大小在编译时是可配置的。MEM_SIZE 定义定义了这类 pbuf 的可用空间大小。
- PBUF_ROM 类型的 pbuf 用于应用程序要发送的数据放置在应用程序管理的存储区的情况。
- PBUF_POOL 主要用于网络设备驱动层，因为分配一个 pbuf 的操作可以快速完成，所以非常适合用于中断处理。




## 内存配置选项

1. 接收数据缓存的大小

网络接口接收到数据包，通过以太网专用 DMA 放到专门的缓冲区。然后在 low_level_input 函数中，被拷贝到 PBUF_POOL 中，再将指向该 PBUF_POOL 的指针传递给 LwIP 协议栈做进一步的处理。这里用于拷贝接收到的数据的 PBUF_POOL 的大小由下面这两个配置选项决定：PBUF_POOL_SIZE 和 PBUF_POOL_BUFSIZE。

PBUF_POOL_SIZE：定义可用的 PBUF_POOL 的个数

PBUF_POOL_BUFSIZE：定义每个 PBUF_POOL 的大小

PBUF_POOL_SIZE * PBUF_POOL_BUFSIZE 的值就是接收数据内存总的大小

用户需要根据接收的数据包的平均大小来设置这两个值。PBUF_POOL_BUFSIZE 设置的太小，可能每个数据包都要多个 pbuf 来保存；设置太大，很少的数据也会占用一个较大的 pbuf 造成浪费。

2.发送数据缓存的大小

LwIP 通过 Heap 的方式可分配的总内存空间大小由 MEM_SIZE 定义，如果应用程序需要发送大量数据，而且这些数据需要拷贝到 LwIP 协议栈中，那么这个值尽量设置大些。

3.连接

LwIP 协议栈中通过 PCB（Protocol Control Blocks）的方式管理各个连接。创建新的 PCB 时，也是通过 memory pool 的方式进行内存分配。

MEMP_NUM_UDP_PCB：定义可以创建的 UDP 连接个数

MEMP_NUM_TCP_PCB：定义可以创建的 TCP 连接个数

MEMP_NUM_TCP_PCB_LISTEN：可以创建 listening TCP 连接的个数

MEMP_NUM_NETCONN：使用 netconn 和 socket 编程时，该值的大小会影响可以同时创建的连接的个数

MEMP_NUM_NETBUF：使用 netconn 和 socket 编程时，该值设置太小，可能导致接收数据时分配内存失败，从而不能同时为几个连接的数据收发服务。

4. TCP 选项

TCP_MSS ：该值规定了 TCP 数据包数据部分的最大长度

TCP_SND_BUF：一个 TCP 连接的发送缓存空间大小。改变这个值只影响一个 TCP 连接可用的发送缓存空间大小。总的发送缓存空间是不会变的（由 MEM_SIZE 决定）。如果同时活动的 TCP 连接个数很多，这个值不宜设置的太大。

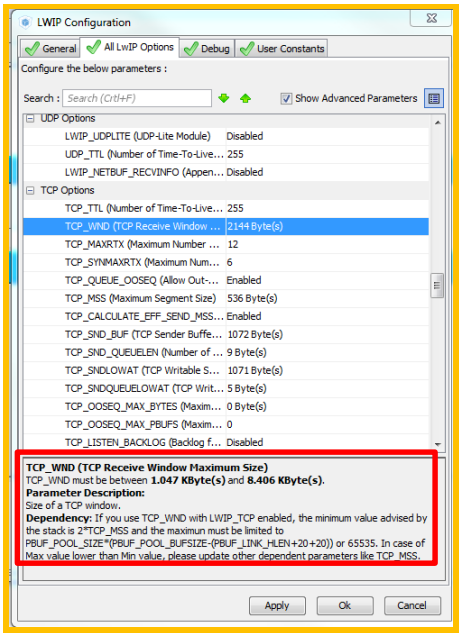
TCP_SND_QUEUELEN：TCP 发送队列中最多的 pbuf 个数

TCP_WND：TCP 接收窗口大小

## 配置正确性检查

LwIP 中的很多配置并不是孤立的，在 lwipopts.h 和 opt.h 的某些配置选项前面都有说明需要满足的条件。在修改这些参数时需要特别注意。另外 LwIP 还提供了“sanity checks”，在编译时（LwIPv1.4.1，之前的版本是在 lwip_sanity_check 函数中进行检查）对 lwipopts.h 和 opt.h 中的一些关键的配置进行检查，如果发现错误就会通过 error 信息进行提示。该功能可以通过宏 LWIP_DISABLE_MEMP_SANITY_CHECKS 关闭，建议在调试时打开。

另外一个可以在配置 LwIP 众多选项时给你提供帮助的工具就是 CubeMX。CubeMX 里的 LwIP 配置页面对每个选项都做了详细解释，包括建议的最大/最小值。见下图：



## 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。