LWIP

Key differences between MAC Address and IP address

- IP addresses are used for routing packets over the internet while MAC addresses are used for local communication
  - ↳ local → gateway (router) → network

What is LwIP
- ↳ stands for Light weight Internet Protocol
- ↳ provides free ICP/IP stack
- ↳ uses less RAM, therefore more suitable for embedded systems
- ↳ supports Most protocols like IPv4, IPv6, TCP, DNS ... etc

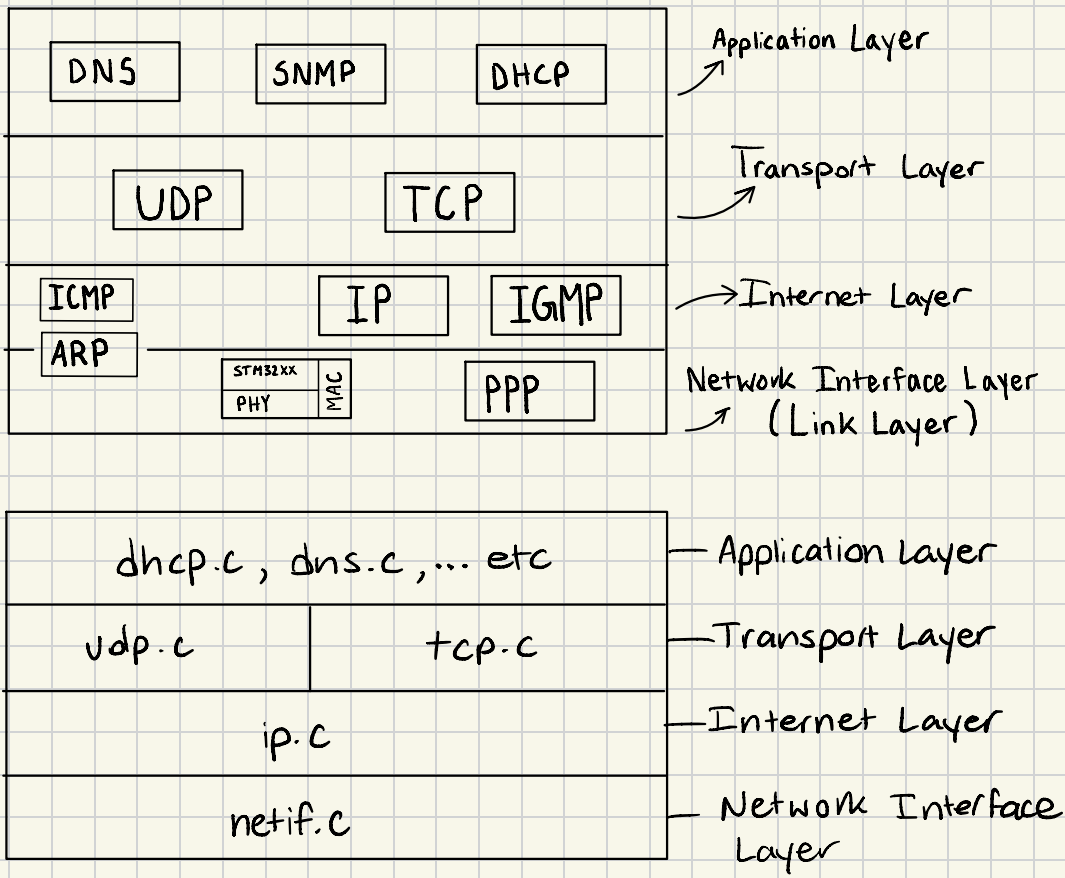
API Types for working with LWIP

RAW API
- ↳ This has the native API of LWIP
- ↳ Allows us to use event callbacks
- ↳ Has the best performance and code size

Netconn API
- ↳ Requires the use of an RTOS
- ↳ Allows the use of multi-threaded operations

Socket API
↳ developed on top of the Netconn API

LwIP Architecture:



Ethernet Controller Interface

↳ The official release of the LwIP does not provide any port to any microcontroller

↳ it comes with an ethernetif.c, that works as an interface between the stack and the ethernet controller

↳ This file is presented as a skeleton to be completed to support a specified architecture.

↳ ethernetif.c contains functions that ensure the transfer of the frames between the low-level driver (stm32_eth.c) and the LwIP stack

Configuration

↳ LwIP can be tuned to suit the applications requirements

↳ The default parameters of the stack can be found in the opt.h file

↳ This file is located under the LwIP directory at src/include/LwIP/

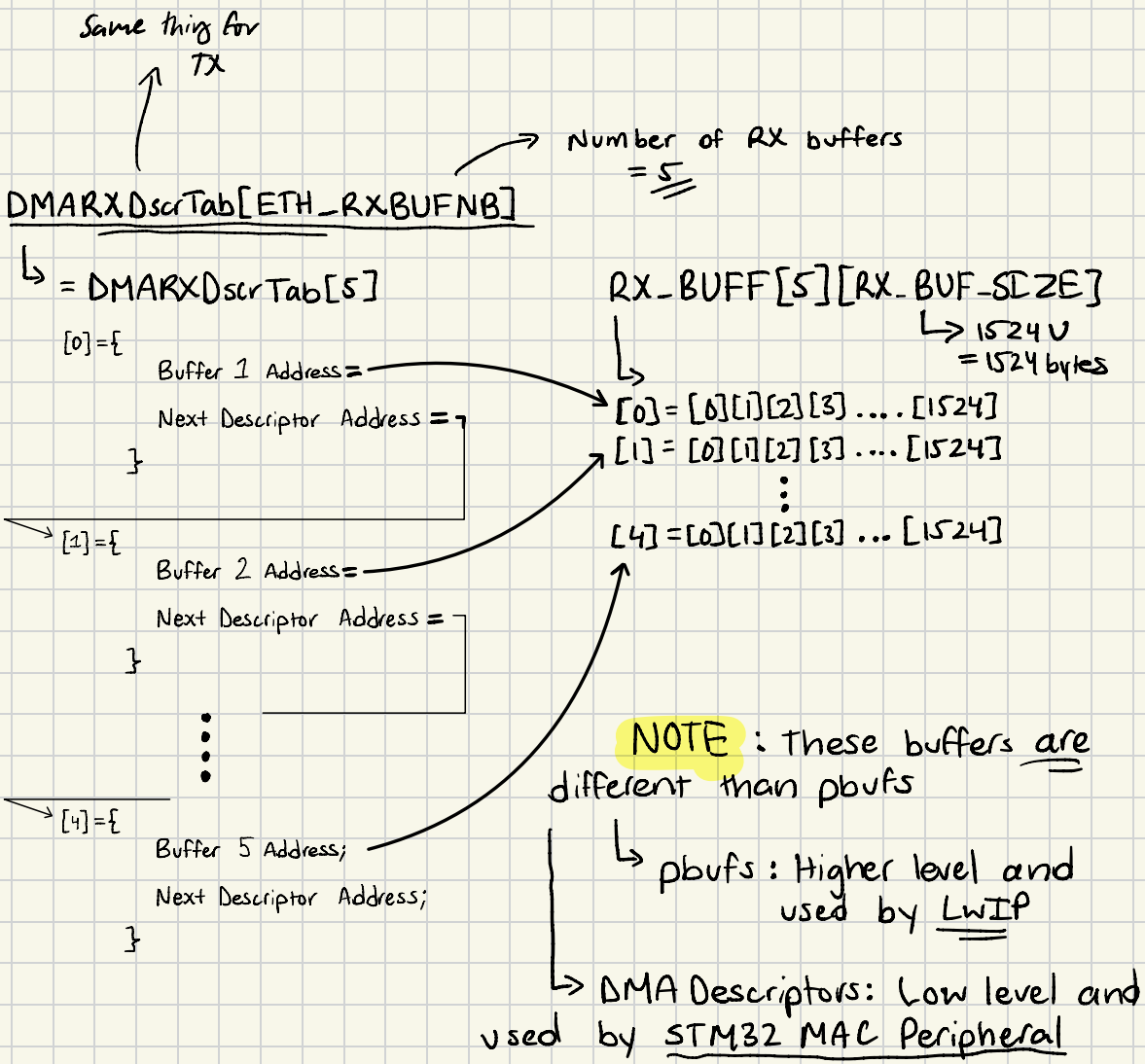
↳ To modify these settings, a new file is defined, lwipopts.h based on the opt.h file, and located under the LwIP directory at Target/

↳ some LwIP configuration Parameters:

- LWIP_DHCP: for enabling/disabling DHCP
- MEMP_NUM_TCP_PCB ; MEMP_NUM_UDP_PCB : For the maximum number of simultaneously active connections, for TCP and UDP connections respectively
- MEM_SIZE: For setting Heap size
- PBUF_POOL_SIZE ; PBUF_POOL_BUF_SIZE : Set the number of buffers and buffersize respectively

ETHERNET DESCRIPTORS & BUFFEERS LINKED LISTS

STM32 uses DMA descriptors to organize the Ethernet low-level buffers, this is done by a Linked List.



What is a PBUF?

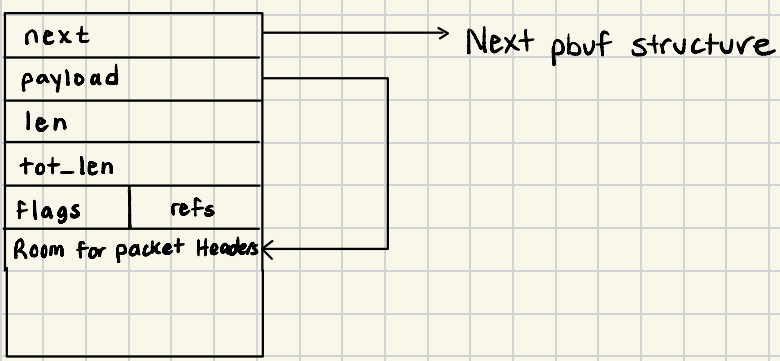
- It is a struct datatype from which we build packets

- It supports dynamic memory allocation for packet contents and can also reference externally managed packet contents both in RAM and ROM

- A packet may span over multiple pbufs, chained as a singly linked list. This is called a pbuf chain.

- Multiple packets may be queued also using this singly linked list. This is called a "packet queue"

Buffer Management (PBUF Struct)



next: pointer to next pbuf in pbuf chain

payload: pointer to packet data payload

len : length of the data content of the pbuf

tot_len: sum of the pbuf len plus all the len fields of the next pbufs in the chain

ref: (on 4 bits) reference count that indicates the number of pointers that reference the pbuf, A pbuf can be released from memory only when its reference count is zero.

flags: (on 4 bits) indicate the type of pbuf

Types of PBUF

PBUF_POOL:

- pbuf allocation is performed from a pool of statically pre-allocated pbufs that have a pre-defined size.

- Depending on the data size that needs to be allocated one, or multiple chained pbufs are allocated

PBUF_RAM:

- pbuf is dynamically allocated in memory (one continguous chunk of memory for the full pbuf)

PBUF_ROM:

- There is no allocation for memory space for user payload, the pbuf payload pointer points to data in the ROM memory, therefore this can only be used for sending constant data

API for managing pbufs

pbuf_alloc : Allocates a new pbuf

pbuf_realloc: Resizes a pbuf

pbuf_ref : Increments the reference count field of a pbuf

pbuf_free : Decrements the pbuf reference count. If the count reaches zero then the pbuf is deallocated.

pbuf_clen : Returns the count number of pbufs in a pbuf chain

pbuf_cat : chains two pbufs together but does not change the reference count of the tail pbuf chain.

pbuf_chain : Chains two pbufs together (tail chain reference count is incremented)

pbuf_dechain: Unchains the first pbuf from its succeeding pbufs in the chain

pbuf_copy_partial : Copies part of the contents of a packet buffer to an application supplied buffer.

pbuf_take : Copies application supplied data into a pbuf

pbuf_coalesce: Creates a single pbuf out of a queue of pbufs

FLOW CHART USING RAW API