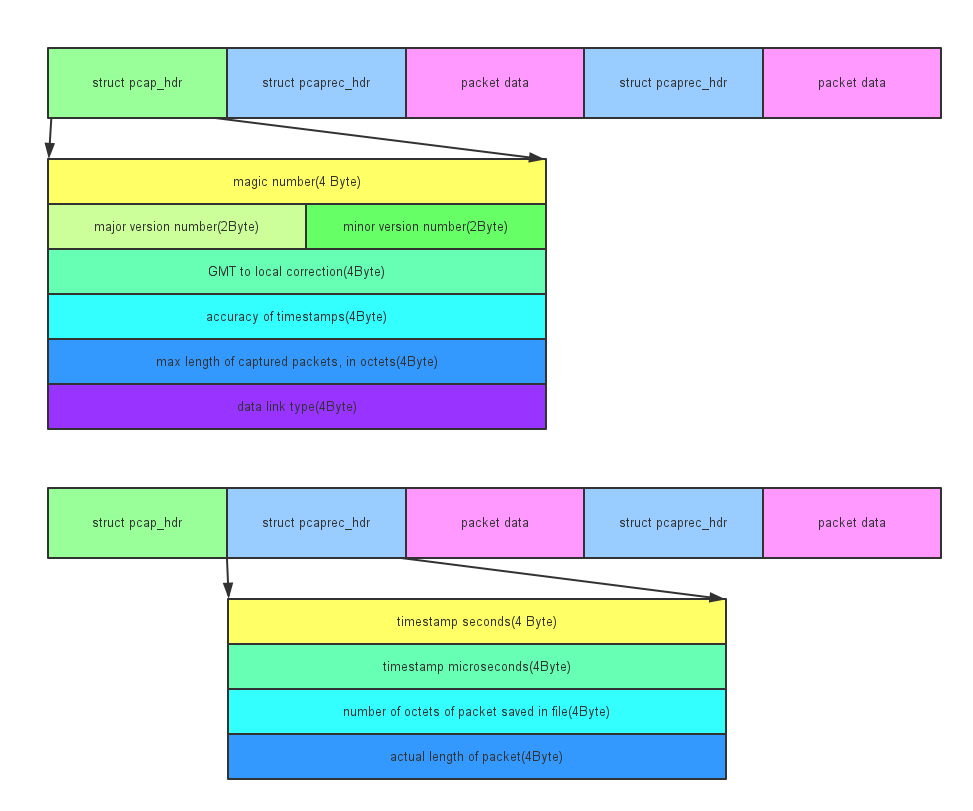
pcap文件格式解析

文章内容保存在

wireshark在截获数据报文以后，可以保存为pcap文件。pcap文件的格式按照一定的存储顺序来保存。以下就是根据wireshark写入pcap文件的顺序以及结构：

wireshark会调用libpcap\_write\_file\_header函数先将struct pcap\_hdr结构写入文件，然后再循环将struct pcaprec\_hdr与packet data的内容写入文件。

wireshark写入的顺序已经定了，那么我们读取的时候，只要按照这个顺序来读，那么就能够最终得到我们的网络包数据，网络包数据是从2层开始的。但是有一点值得说明的是，wireshark支持多种文件格式的读取，其中包括pcap文件，pcapng文件等等，文件不一样，那么文件格式多少是一些差别的，不能都使用上面的结构图来解析，好在面对众多的文件格式，wireshark给我们提供了解读文件的模块，wiretap，利用wiretap模块的api函数，即可解析文件。

其实文件格式分析到这里，如果以前就有解析网络数据包的框架，那么就可以不用看wireshark的处理流程了。但是我这里没有网络数据包的解析框架，那么只能利用wireshark本身的处理流程。

wireshark是使用epan模块对数据包进行解析。epan是协议分析模块(Wireshark Protocol Analyzer Library)。P表示protocol, an表示Analyzer，e这个缩写我一直没搞清楚，可能是ethernet的缩写，也可能是ethereal。epan下有一个dissector的目录，专门将数据包分段，然后按照协议依次进行解析处理。

wireshark通过函数process\_packet开始数据包处理。初始化dissector等相关数据后，会调用dissect\_packet，然后通过找到frame\_handle全局句柄，回调frame的处理函数dissect\_frame。dissect\_frame这里，会利用回调调用dissect\_eth\_withoutfcs处理mac层的数据，(对于http报文)之后依次调用dissect\_ip, dissect\_tcp, dissect\_http协议处理。

乍看一下，这个流程还是相当简单明了的。但是这块却是我花时间最多的地方，函数跳转来跳转去的，一直找不到逻辑关系。由于对ip以及tcp是最熟的，那么先看dissect\_ip是怎么调用到dissect\_tcp的，这个理顺了，其他也好理解了。

dissect\_ip通过解析ip协议，获取到4层protocol，然后在函数最后，调用dissector\_try\_uint来根据protocol匹配下一层协议，也就是要匹配tcp。在哪匹配呢？原来ip协议在初始化的时候，注册了一个名称为ip.proto的hash表，在tcp协议初始化时，tcp协议向ip.proto的hash表，添加了协议为6，且回调函数为dissect\_tcp的函数。那么只要在dissector\_try\_uint时通过协议匹配到6，那么就能找到dissect\_tcp进行下一层协议的处理。根据这样的原则，从dissect\_frame到dissect\_http依次传递，就能够完成整个http协议的处理。

简化一下dissect\_ip找到dissect\_tcp的流程图：

