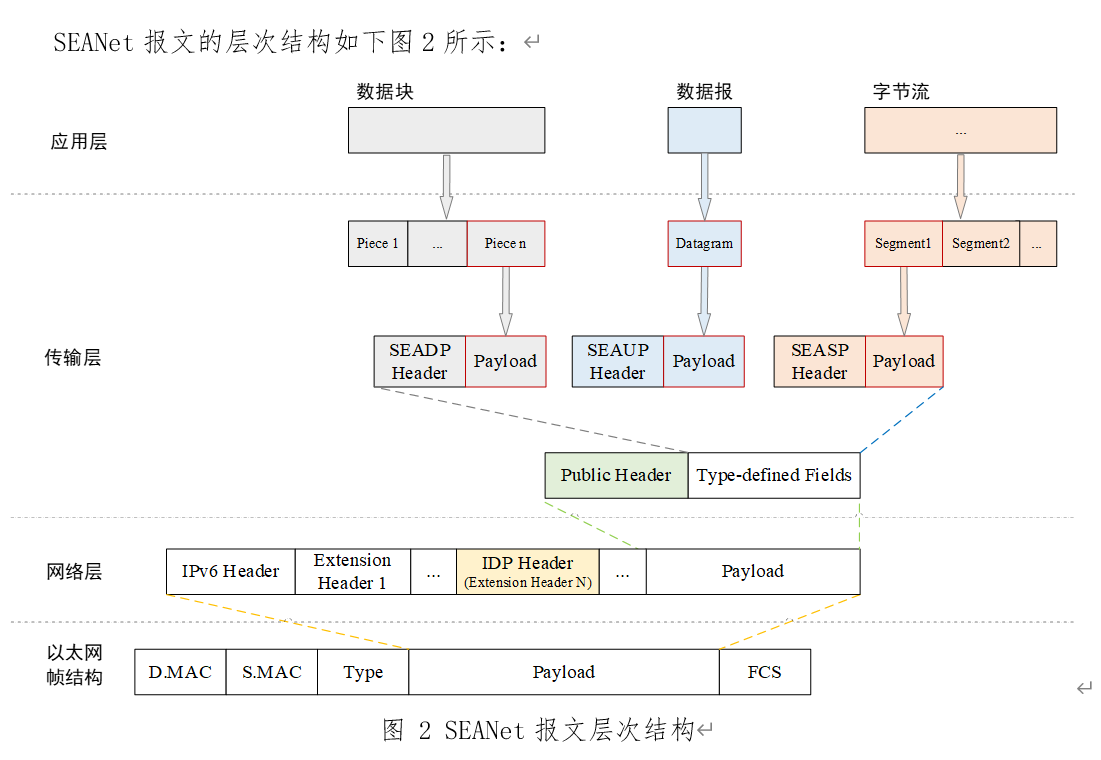
**任务相关的知识和概念的辨析与理解：**

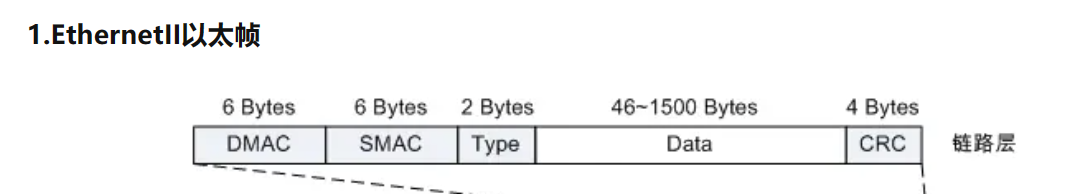
1. **数据报文基本结构的理解：**

在数据链路的不同节点之间传递的是以太网帧，对以太网帧进行解析得到网络层的IPV6报文，IPV6的报文由报头和PayLoad组成，经过解析去掉报头，IPV6的PayLoad是IDP的报文，同样，这样层层封装，我们care的最高层是传输层，更高层的我们并不care，将更高层的报文视作传输层的PayLoad，并不关注其中的细节。



* 1. **以太网帧结构**

wireshark捕捉到的是以太网帧，并且使用的是Ethernet II标准。



在IP数据包之外，包裹着以太网报头。这个报头有14个字节，IPV6报文的第一个字节是第15个字节。

DMAC为目的地MAC地址

SMAC为源MAC地址

Type/Length：占据2个字节，指示了数据包中包含的协议类型或者数据的长度。当值大于等于0x0600（1536）时，它表示协议类型；当值小于0x0600时，它表示数据长度。

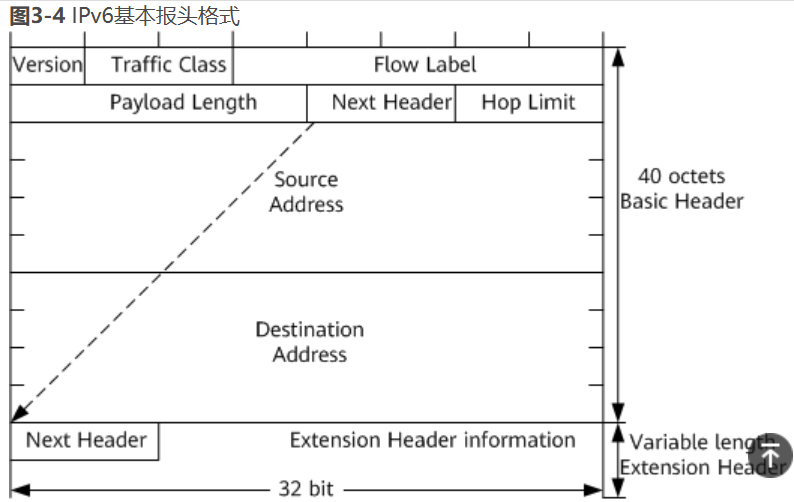
数据（Data）：数据字段中包含了上层协议的数据，例如IP数据包。数据字段的长度可以根据类型/长度字段的值来确定。

帧校验序列（Frame Check Sequence，FCS/CRC）：占据4个字节，用于错误检测，通常使用CRC-32算法生成。

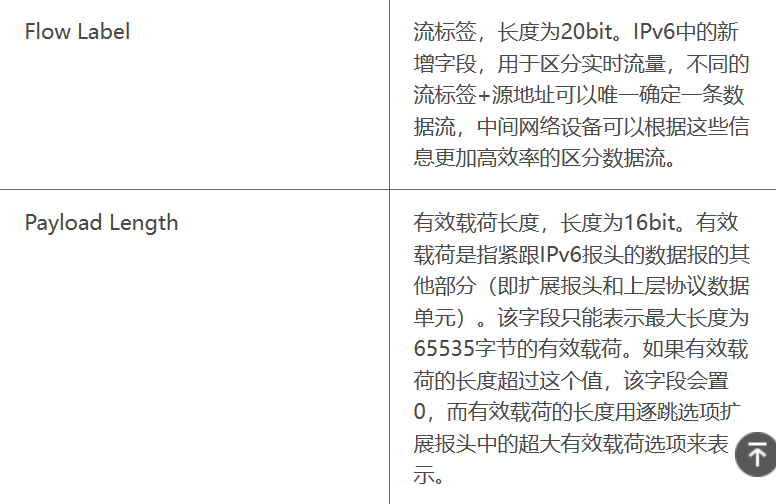
* 1. **IPV6报文结构**

因为我们的协议是附加在IPV6协议中的，因此要想读懂接受和发出的数据包，必须要先对IPV6有清楚的了解。

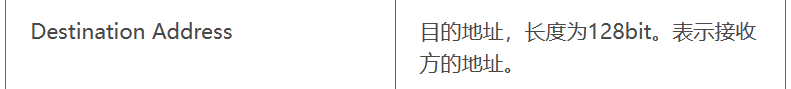
IPV6报文有40字节（Byte）即320bit的固定报头长度，后面会跟IPV6的Payload即第二个next header和拓展的协议数据单元











扩展报头格式如下：



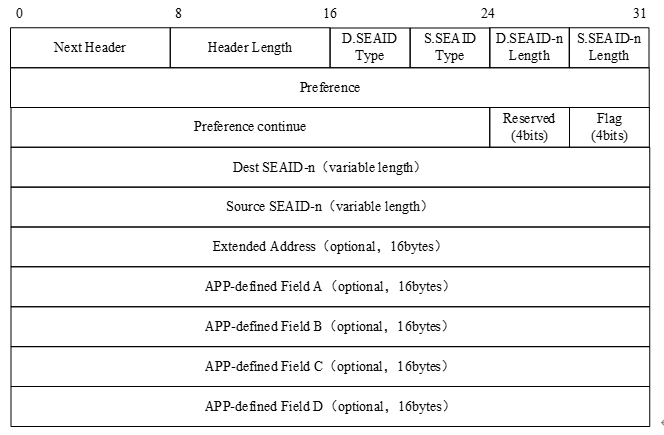
* 1. **IDP报文结构：**

首先对IDP（ID Protocol）首部进行分析（这是一个在网络层的附加性协议（我认为））负责SEAID的地址变换（SEAID用来标识对应的网络服务）（具体的的含义可以到SEANet报文设计中查阅），这里只介绍与任务相关的内容。Preference表示IDP处理的一些定制选项，

刚开始有6bit的业务标识。支持64种业务，对应64中拓扑或64种数据包路由的选路

其中第166～24bit中是IrA Flag（即刻冲寻址标志量0～7）其中比特5对应的是缓存标识，后面接着是对应的8个即刻重寻址参数（IrA Para）对应着8个flag量。从高比特往低比特对应，即第三个IrA Para对应就是缓存的参数（里面存储的是缓存数据包的跳数）

IDP报文首部格式如下：



上图中前三行是固定长度，后面都是可变长度，在图中为了美观对齐到32bit了，但是DestSEAID其实最大长度有160bit，远远大于32bit，这里要注意不要误解了。

* 1. **我们的报文设计所涉及到的IPV6结构与特点**

在数据包的构造上，SEANET数据包**利用了IPV6报文构造中的next header的设计**（next header是ipv6报文中ipv6首部之后紧跟的下一个拓展首部或上一层协议的类型。在ipv6首部之后，可以有0个或者多个拓展首部，最后一个拓展首部是上层（传输层）协议的数据）使用next header字段把SEANET协议系统的网络层协议IDP的首部引入IPV6的报文，这样子就能使用IPV6数据包作为载体。在ipv6报文中加入自己带有定制功能的拓展协议。在最后一个next header中填入SEANET协议系统的传输层协议的标识符，直接在传输层中应用自己的协议（具有增强传输功能）。

IPV6的next header是需要重点理解的。我们所定义的3.5层、3.6层还是什么，其实本质上都是IP层的扩展协议，从形式上就能理解（都服从IPV6定义的next header这个规则。）通过扩展报头，使得IP层可以定义更多的协议和中间层，从而实现更多定制化的功能。但是next header的引入也引入了数据包内外层之间的通信问题。

* 1. **需要理解的疑点**

**1.5.1 报文嵌套封装的结构问题**

要明白一个问题，IPV6的报头之后跟的payload其实是IDP报文，IDP报头之后跟的数据包其实是传输层报文，也就是说外层封装的协议的报头是连着的，我们只关心到传输层，也就是说我们把更高层的数据包的报头和payload都看成数据。也就是都看成传输层的payload。

**1.5.2 高可靠的理解问题**

SEANET做的高可靠其实是在UDP的基础上提高可靠性，尽量降低丢包率。并不是真正意义上的高可靠。高可靠增强传输其实是在传输层中做的，其中涉及到两对地址，分别在四个位置上填写：（IPV6报头里面的source IP和Destanation IP、 IDP报头中的extended address以及传输层协议报头中Net defined field中的RS IP字段。 增强传输在物理意义上需要知道端到端和每一跳的点到点地址。这四个地址和四个报文中的位置的对应关系是：RS IP填写点对点传输的源地址，IPV6的Destination地址写点到点目的地址；端到端的源地址填写在IPV6的source IP，端到端目的地址填写在IDP报头的extended address中）

1. **SEANET协议落地实施的方法及其相关概念：**

SEANET协议整体上涉及到3.5～4层（网络层和传输层）所谓协议之所以能生效，主要是因为对数据包进行了特殊的构造，而且有专门可以处理这种数据包的网络设备在，这样子应用了这种协议的数据包才得以在网络上进行转发。而处理这种类型数据包的网络设备则是通过很多流表串连而成的数据包处理流水线来实现处理功能的。

**2.1 数据包处理流水线：**

Dimond其实是SEANET数据包处理流水线的一个版本。



所有功能都是用上图的流水线做的，实际结构上是每张流表串连在一条线上，根据action里面的行为（go to 哪张表）来实现逻辑上的不同功能下流水线不同的结构。（上图所示是缓存包的处理流水线）

所谓MAC是指在数据传输路径上其他结点的端口（插光纤的口），Host是指当前自己结点中的主机（PCIE口）。二者的区别就在于数据包是在链路上继续跳还是说自己这个结点拿到数据包之后不继续跳，而是把数据包提交给自己结点内上层的主机。

AE是access Equipment的意思，其实就是主机，overlay是抽象到更高层的含义。

**2.2 流表相关基本概念：**

流表（flow table）产生背景是SDN把传统的网络硬件设备与网络解耦，将网络控制逻辑集中到控制器中使用软件对功能进行定义。流表是控制器中用来管理和控制网络流量的关键数据结构（控制器给设备下流表）。这种数据结构用于定义如何处理网络中的数据流（即该使用什么样的规则和操作）每个流表中都会定义一系列匹配条件和对应满足条件后去执行的操作。匹配到对应的条件之后，交换机会根据规则中定义的操作来处理数据包，有可能是转发、丢弃或修改等操作。流表中的匹配条件可以包括多个字段，如源IP地址、目标IP地址、源端口、目标端口、协议类型等。操作则可以包括转发数据包到指定的端口、修改数据包头部、应用QoS（Quality of Service）策略等。

需要注意的是，流表中匹配的字段有可能不是数据包中有的，比如dispatch表中匹配的local（1）其实就是通过比对数据包的IP来判断是否是要到本地的数据包的一个1位标志量。这个标志量与数据包相绑定。Inport type也是这种情况。

具体的做法可能是对数据包的字段进行解析之后哈希到一个地址（哈希表），在对应的地址中存储着指令或者操作，将其取出来进行操作。

1. **流水线对数据包处理流程与相关疑难点**
   1. **流水线对数据包处理的流程（以缓存包跳数为0的情况为例）**

输入数据包：



输出数据包：



对路由来的数据包，先进入可靠传输接受侧流水线，在这个流水线中对数据包进行解析（解包），解包之后使用local流表先对其目的地址进行范围匹配。在对应地址范围之内的话，对数据包中的高可靠使能字段和NACK标志字段进行判断。对于缓存数据包，在流表中匹配的结果是把数据包送入dispatch流表中，在流水线结束之后，会将数据包进行逆解析，把数据包再封上，送入IDP流水线之中，对IDP报文进行解析。Dispatch流表会对解析后的报文的local等6个字段进行匹配，并且使用掩码来表明在哪些情况下对哪些字段don’t care。，在不同的表项中定义了不同字段情况条件下对应的action的代码和下一个流表的代码，当缓存跳数为0（ira parameter3/cache hop=0），在这种情况下，只关心三个字段：Cache Hop（0）即缓存跳数为0；SEAID.TYPE（01）即目的SEAID是数据，源SEAID是传输服务；IRA（20）即仅对应缓存服务。因此在流表里对应的action是把输出端口设置为主机host,并且送往AE。然后这条流水线的处理就结束了，于是把数据包逆解析（封装好）送到IP转发流水线中，

这个线（Fib即forwarding）三张表的工作主要是根据IP来确定向哪个Mac端口转发。首先进入FIB\_Host，根据IP地址范围给出Adjcent表中的偏移量（把不同范围的IP地址分配到不同的组里，Adjcent表根据这些组分配对应的Mac地址）（需要注意的是，我们所说的IP范围，其实在流表里面是通过一条一条表项来对应实现的。Fib Host表里面每一条表项对应一个精确的IP地址。也就是说这张表会对精确的IP地址进行匹配）如果包的目的IP没有命中这些范围中的精确IP地址，则将数据包送往FIB\_LPM，这个表用来保底，Fib Host表对精确IP进行匹配，而Fib\_LPM使用最长前缀匹配（更为模糊的匹配方法，而/32这些指的是掩码，只关心前32位。），使用模糊匹配之后还是把数据包往Adjcent表里送，如果实在命中不了，就是处理不了，就packge in，Adjcent也是一样，如果处理不了就packge in，如果能正常处理，就按之前分的偏移量（即分的小组）设置转发的Mac地址等参数。至于Fib\_LPM里面单独设置了一个特定的IP地址来送往Adjcent的4偏移量处，则是为了测试这个功能，故意写一个IP，到时候测试的时候构造一个这个IP的包，看看是不是到4那边。

* 1. **流表相关疑难点理解：**

其实在上面描述的过程中就已经解释了部分疑难点。接下来解释其他的

Local使用IPV6的DestinationIP作为匹配项，有这几层原因：１.如果是不带高可靠服务的IPV6包，IPV6的DestIP就是正常的端到端目的IP地址，如果是带高可靠服务的，那么IPV6报头中的DestIP字段会填写上点到点的下一跳地址。因此通过IP地址的不同可以初步区分在我这个节点的本地，接收到的这个数据包是不是需要高可靠服务的，也就是说高可靠服务中在DestIP字段中填写的下一跳地址是有一定特点的（这个特点由控制器决定，流表都是控制器计算之后下达的。）如果符合高可靠这个宽松条件，就把数据包给到高可靠流表，使用更严格的条件区分不同情况。如果不符合，那么就是正常IPV６包，走IDP解析即可。

FibHost表也是一样，使用IP来作为匹配项确实是因为IP中带有可以决定下一步行为的信息。比如FibHost和Fib\_LPM和Adjecent表共同决定的行为是向不同Mac地址发包这种。（FibHost先把Ip地址分成小的类别，Fib\_LPM保底，然后Adjcent把这些小类别对应到不同的Mａｃ端口）FibHost根据不同的IP范围，把这些IP地址分到不同的小类别中（用ｂａｓｅ　ａｄｄｒｅｓｓ来区分不同小类别）根据这些小类别（不同的ｂａｓｅ　ａｄｄｒｅｓｓ），Adjcent表可以对应到不同的Mac地址上。对于控制器来说：根据这些Mac地址，控制器可以倒推（计算出）对应每种情况的IP，计算出流表之后下达给交换机即可。也就是说其实流表的匹配项都是控制器根据对应的行为去倒推计算出来的。（如果是高可靠，那么其IP的特点是在某个区域内，所以我把匹配项设成IP，然后IP满足这些范围就是高可靠，像这样子）

（需要注意的是，在逻辑上我们解释说local表是在匹配一个范围，但其实在实现上是一个一个具体表项，每一个具体精确的IP地址对应一个表项action）

流表中的流状态是什么？

**原始资料（分析问题解决问题的思路）**

**对IDP缓存（随路缓存）的理解：**

服务于文件数据和媒体数据，在传输的链路上的节点可以把数据包缓存在节点的本地，如果后续有其他节点需要同样的数据包，并且传输链路经过先前缓存数据包的那个节点，那么那个节点就可以直接提供服务（感觉有CDN和P2P传输的思想在其中）把集中的数据分散到网络中。缓存是通过ira字段标识该数据包是seanet缓存包。做的是overlay的缓存。对缓存的请求和响应包分开做了处理。在支持seanet的路由器节点中，对被标识的IDP缓存报文主要做四种处理：1.对来自主机（host）的缓存请求，通过OFIB发出，对来自MAC的缓存请求，转给AE（通过host），（上两条是对缓存请求包的处理）3.对跳数不为0的缓存数据，跳数减一并通过OFIB发出，对跳数为0的数据，转给AE(通过host)

我的第一个任务就是看对跳数为0的缓存数据响应包进行处理

对缓存请求和缓存响应数据包的初步理解：请求就是向某个节点请求其先前缓存的数据，响应就是该节点将被请求的数据包发送给请求者

一些需要补充的概念：1.流表（flow table）产生背景是SDN把传统的网络硬件设备与网络解耦，将网络控制逻辑集中到控制器中使用软件对功能进行定义。流表是控制器中用来管理和控制网络流量的关键数据结构。这种数据结构用于定义如何处理网络中的数据流（即该使用什么样的规则和操作）每个流表中都会定义一系列匹配条件和对应满足条件后去执行的操作。匹配到对应的条件之后，交换机会根据规则中定义的操作来处理数据包，有可能是转发、丢弃或修改等操作。流表中的匹配条件可以包括多个字段，如源IP地址、目标IP地址、源端口、目标端口、协议类型等。操作则可以包括转发数据包到指定的端口、修改数据包头部、应用QoS（Quality of Service）策略等。

需要注意的是，流表中匹配的字段有可能不是数据包中有的，比如dispatch表中匹配的local（1）其实就是通过比对数据包的IP来判断是否是要到本地的数据包的一个1位标志量。这个标志量与数据包相绑定。Inport type也是这种情况。

具体的做法可能是对数据包的字段进行解析之后哈希到一个地址（哈希表），在对应的地址中存储着指令或者操作，将其取出来进行操作。

Dimond其实是SEANET数据包处理流水线的一个版本

所以要先理解什么是SEANET协议，SEANET协议整体上涉及到3.5～4层（网络层和传输层）所谓协议之所以能生效，主要是因为对数据包进行了特殊的构造，而且有专门可以处理这种数据包的网络设备在，这样子应用了这种协议的数据包才得以在网络上进行转发。而处理这种类型数据包的网络设备则是通过流表来实现处理功能的。

在数据包的构造上，SEANET数据包利用了IPV6报文构造中的next header的设计（next header是ipv6报文中ipv6首部之后紧跟的下一个拓展首部或上一层协议的类型。在ipv6首部之后，可以有0个或者多个拓展首部，最后一个拓展首部是上层（传输层）协议的数据）使用next header字段把SEANET协议系统的网络层协议IDP的首部引入IPV6的报文，这样子就能使用IPV6数据包作为载体，解决了应用的问题。在ipv6报文中加入自己带有定制功能的协议。在最后一个next header中填入SEANET协议系统的传输层协议SEA首部，直接在传输层中应用自己的协议（具有增强传输功能）代替掉TCP这些

IPV6的next header是需要重点理解的。我们所定义的3.5层、3.6层还是什么，其实本质上都是IP层的扩展协议，从形式上就能理解（都服从IPV6定义的next header这个规则。）通过扩展报头，使得IP层可以定义更多的协议和中间层，从而实现更多定制化的功能。但是next header的引入也引入了数据包内外层之间的通信问题。

要明白一个问题，IPV6的报头之后跟的payload其实是IDP报文，IDP报头之后跟的数据包其实是传输层报文，也就是说外层封装的协议的报头是连着的，我们只关心到传输层，也就是说我们把更高层的数据包的报头和payload都看成数据。也就是都看成传输层的payload。

SEANET做的高可靠其实是在UDP的基础上提高可靠性，尽量降低丢包率。并不是真正意义上的高可靠。高可靠增强传输其实是在传输层中做的，其中涉及到两队地址，分别在四个位置上填写：（IPV6报头里面的source IP和Destanation IP、 IDP报头中的extended address以及传输层协议报头中Net defined field中的RS IP字段。 增强传输在物理意义上需要知道端到端和每一跳的点到点地址。这四个地址和四个报文中的位置的对应关系是：RS IP填写点对点传输的源地址，IPV6的Destination地址写点到点目的地址；端到端的源地址填写在IPV6的source IP，端到端目的地址填写在IDP报头的extended address中）

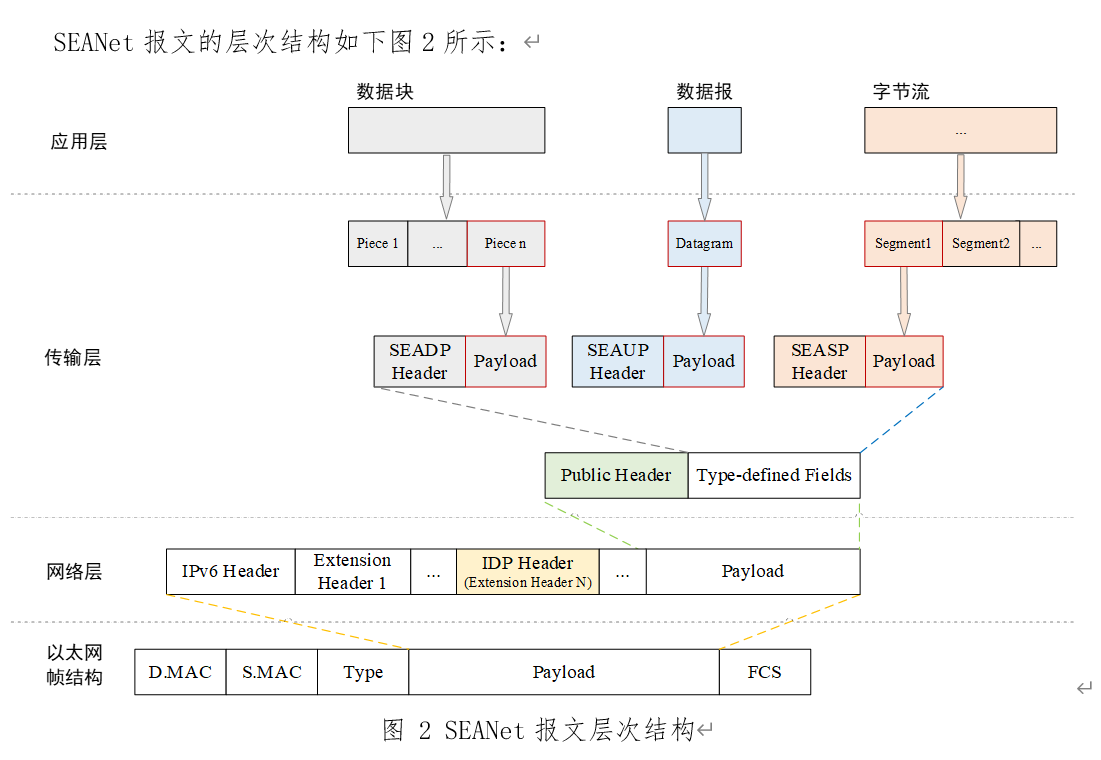
有一些不是很容易理解的点：

Local使用IPV6的DestinationIP作为匹配项，有这几层原因：１.如果是不带高可靠服务的IPV6包，IPV6的DestIP就是正常的端到端目的IP地址，如果是带高可靠服务的，那么IPV6报头中的DestIP字段会填写上点到点的下一跳地址。因此通过IP地址的不同可以初步区分在我这个节点的本地，接收到的这个数据包是不是需要高可靠服务的，也就是说高可靠服务中在DestIP字段中填写的下一跳地址是有一定特点的（这个特点由控制器决定，流表都是控制器计算之后下达的。）如果符合高可靠这个宽松条件，就把数据包给到高可靠流表，使用更严格的条件区分不同情况。如果不符合，那么就是正常IPV６包，走IDP解析即可。

FibHost表也是一样，使用IP来作为匹配项确实是因为IP中带有可以决定下一步行为的信息。比如FibHost和Adjecent表共同决定的行为是向不同Mac地址发包这种。（FibHost先把Ip地址分成小的类别，然后Adjcent把这些小类别对应到不同的Mａｃ端口，也就是分了两层。）FibHost根据不同的IP范围，把这些IP地址分到不同的小类别中（用ｂａｓｅ　ａｄｄｒｅｓｓ来区分不同小类别）根据这些小类别（不同的ｂａｓｅ　ａｄｄｒｅｓｓ），Adjcent表可以对应到不同的Mac地址上。对于控制器来说：根据这些Mac地址，控制器可以倒推（计算出）对应每种情况的IP，计算出流表之后下达给交换机即可。也就是说其实流表的匹配项都是控制器根据对应的行为去倒推计算出来的。（如果是高可靠，那么其IP的特点是在某个区域内，所以我把匹配项设成IP，然后IP满足这些范围就是高可靠，像这样子）

（需要注意的是，在逻辑上我们解释说local表是在匹配一个范围，但其实在实现上是一个一个具体表项，每一个具体精确的IP地址对应一个表项action）

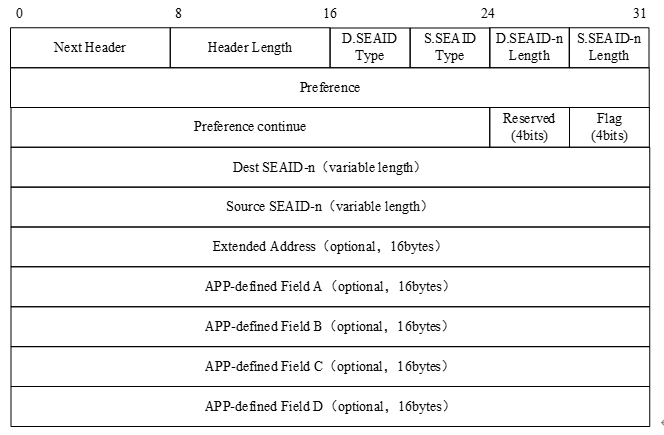
AE是access Equipment的意思，其实就是主机



首先对IDP（ID Protocol）首部进行分析（这是一个在网络层的附加性协议（我认为））负责SEAID的地址变换（SEAID用来标识对应的网络服务）（具体的的含义可以到SEANet报文设计中查阅），这里只介绍与任务相关的内容。Preference表示IDP处理的一些定制选项，

刚开始有6bit的业务标识。支持64种业务，对应64中拓扑或64种数据包路由的选路

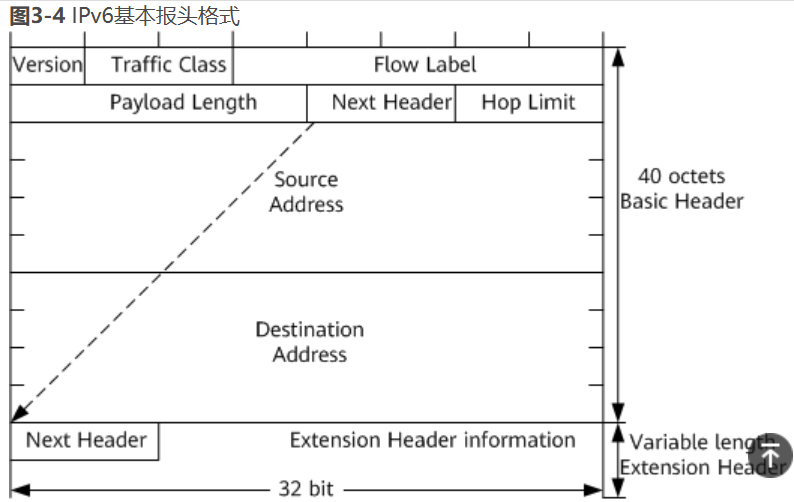
其中第166～24bit中是IrA Flag（即刻冲寻址标志量0～7）其中比特5对应的是缓存标识，后面接着是对应的8个即刻重寻址参数（IrA Para）对应着8个flag量。从高比特往低比特对应，即第三个IrA Para对应就是缓存的参数（里面存储的是缓存数据包的跳数）



IDP首部格式

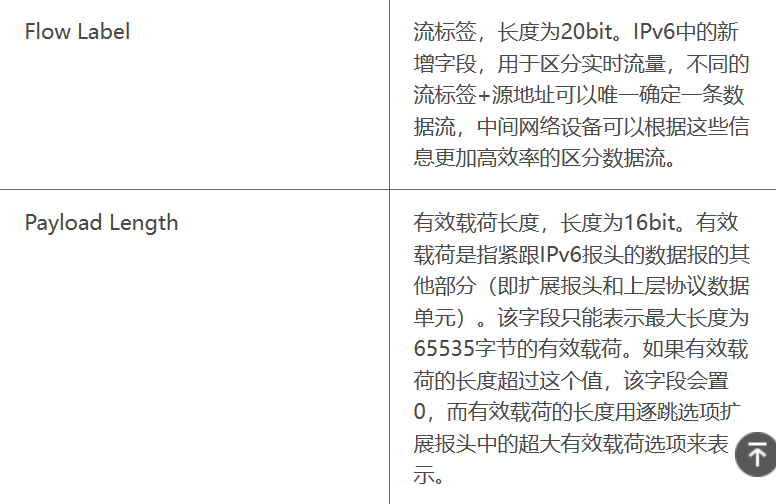
上图中前三行是固定长度，后面都是可变长度，在图中为了美观对齐到32bit了，但是DestSEAID其实最大长度有160bit，远远大于32bit，这里要注意不要误解了。

因为我们的协议是附加在IPV6协议中的，因此要想读懂接受和发出的数据包，必须要先对IPV6有清楚的了解。

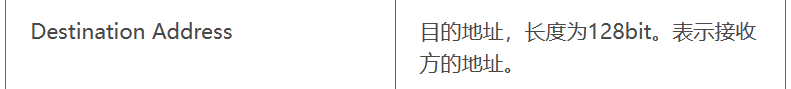


IPV6报文有40字节（Byte）即320bit的固定报头长度，后面会跟IPV6的Payload即第二个next header和拓展的协议数据单元





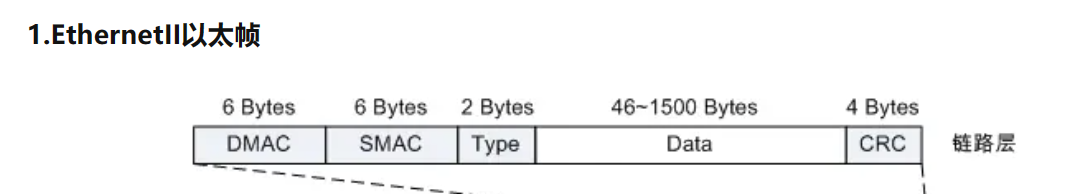




下面是扩展报头的格式：



因为wireshark捕捉到的是以太网帧，并且使用的是Ethernet II标准，因此要读懂数据包还需要对以太网帧格式有清楚的了解。



在IP数据包之外，包裹着以太网报头。这个报头有14个字节，IPV6报文的第一个字节是第15个字节。

DMAC为目的地MAC地址

SMAC为源MAC地址

（Type/Length）：占据2个字节，指示了数据包中包含的协议类型或者数据的长度。当值大于等于0x0600（1536）时，它表示协议类型；当值小于0x0600时，它表示数据长度。

数据（Data）：数据字段中包含了上层协议的数据，例如IP数据包。数据字段的长度可以根据类型/长度字段的值来确定。

帧校验序列（Frame Check Sequence，FCS）：占据4个字节，用于错误检测，通常使用CRC-32算法生成。

对其中输入的数据包进行格式分析，可以得到：第一个包含在ipv6固定长度报头中的next header的值是146（h92）（应该指向IDP扩展头）IDP扩展部分的next header指向147（h93）（应该是SEANET扩展头）



所有功能都是用上图的流水线做的，实际结构上是每张流表串连在一条线上，根据action里面的行为（go to 哪张表）来实现逻辑上的上图所示的结构。

所谓MAC是指在数据传输路径上其他结点的端口（插光纤的口），Host是指当前自己结点中的主机（PCIE口）。二者的区别就在于数据包是在链路上继续跳还是说自己这个结点拿到数据包之后不继续跳，而是把数据包提交给自己结点内上层的主机。

缓存包的处理过程是：

1. 对路由来的数据包，先进入可靠传输接受侧流水线，在这个流水线中对数据包进行解析（解包），解包之后使用local流表先对其目的地址进行范围匹配。在对应地址范围之内的话，对数据包中的高可靠使能字段和NACK标志字段进行判断。对于缓存数据包，在流表中匹配的结果是把数据包送入dispatch流表中，在流水线结束之后，会将数据包进行逆解析，把数据包再封上，送入IDP流水线之中，对IDP报文进行解析。Dispatch流表会对解析后的报文的local等6个字段进行匹配，并且使用掩码来表明在哪些情况下对哪些字段don’t care。，在不同的表项中定义了不同字段情况条件下对应的action的代码和下一个流表的代码，当缓存跳数为0（ira parameter3/cache hop=0），在这种情况下，只关心三个字段：Cache Hop（0）即缓存跳数为0；SEAID.TYPE（01）即目的SEAID是数据，源SEAID是传输服务；IRA（20）即仅对应缓存服务。因此在流表里对应的action是把输出端口设置为主机host,并且送往AE。然后这条流水线的处理就结束了，于是把数据包逆解析（封装好）送到IP转发流水线中，

这个线（Fib即forwarding）三张表的工作主要是根据IP来确定向哪个Mac端口转发。首先进入FIB\_Host，根据IP地址范围给出Adjcent表中的偏移量（把不同范围的IP地址分配到不同的组里，Adjcent表根据这些组分配对应的Mac地址）（需要注意的是，我们所说的IP范围，其实在流表里面是通过一条一条表项来对应实现的。Fib Host表里面每一条表项对应一个精确的IP地址。也就是说这张表会对精确的IP地址进行匹配）如果包的目的IP没有命中这些范围中的精确IP地址，则将数据包送往FIB\_LPM，这个表用来保底，Fib Host表对精确IP进行匹配，而Fib\_LPM使用最长前缀匹配（更为模糊的匹配方法，而/32这些指的是掩码，只关心前32位。），使用模糊匹配之后还是把数据包往Adjcent表里送，如果实在命中不了，就是处理不了，就packge in，Adjcent也是一样，如果处理不了就packge in，如果能正常处理，就按之前分的偏移量（即分的小组）设置转发的Mac地址等参数。至于Fib\_LPM里面单独设置了一个特定的IP地址来送往Adjcent的4偏移量处，则是为了测试这个功能，故意写一个IP，到时候测试的时候构造一个这个IP的包，看看是不是到4那边。