# 简介

FD.io（the Fast Data Project），快速数据项目是旨在为开源软件提供高性能网络解决方案的的综合项目。

VPP( the Vector Packet Processing library)，矢量数据包处理库，是FD.io中的核心项目之一。 VPP是一种可扩展的框架，为通用CPU提供产品级质量的用户空间交换机/路由器功能。在虚拟化的路由交换中，DPDK+VPP，是一种能极大提升虚拟转发设备性能的技术方案。

VPP内部业务逻辑是通过一系列的node连接来实现的，这些node通常在初始化时定义好，比如二层以太处理ethernet-input，三层ip4-input等，通过初始化定义，将node连接成一个有序的向量图，来实现VPP的业务功能。

VPP可以通过Feature机制灵活地控制node之间的连接。每个feature是一个node，用户可以启用或停止某个或某些feature，来达到自己想要处理的逻辑。用户可以自己写插件，把自己定义的node（业务逻辑）加入的指定的位置。通过为接口指定feature可以在接口上启用对应的feature功能。

VPP中，将不同的feature按照类型分成了不同的组，每组feature称之为一个arc。arc中的feature按照代码指定的顺序串接起来。arc结构中，记录这组feature中的起始node和结束node。系统初始化时，会完成初步的排序，但并没有应用到对应的接口中。

本文以VPP 2020.2.27日主分支上的代码来分析。

# 初始化

VNET\_FEATURE\_ARC\_INIT宏用来注册vnet\_feature\_arc\_registration\_t 类型的arc，将其注册到vnet\_feature\_main\_t类型的全局变量feature\_main。每个arc包含一组feature。

arc类型如下：



每种arc都对应了不同的一组功能，常用的如ip4-unicast（处理IPv4报文单播）、ip4-local（处理目的为自己的IPv4报文）、ip4-punt（处理IPv4本地上送报文）等。

VNET\_FEATURE\_INIT宏用来注册vnet\_feature\_registration\_t 类型的feature，将其注册到vnet\_feature\_main\_t类型的全局变量feature\_main。一个feature等价于一个node。每个feature都可以通过外部命令行启用/停止。

# 关键数据结构

## vnet\_feature\_main\_t



该结构是所有feature数据结构的入口主结构体。具体的功能如注释，其中红色框内的在初始化时由vnet\_feature\_init对相关参数做对应的初始化。

## vnet\_feature\_config\_main\_t



该结构是所有feature相关的配置主结构体，每个arc有一个对应的该结构体。在为接口开启feature时，会创建对应的接口体，并按照接口索引设置config\_index\_by\_sw\_if\_index这个vec向量，保存配置的索引。

## vnet\_config\_main\_t



feature配置主结构体，里面的config\_pool是配置的内存池，不同的接口如果设置了不同的feature就会申请不同的vnet\_config\_t。

每创建一个vnet\_config\_t，都会先对其调用vnet\_config\_init进行初始化。config\_pool是真正存放vnet\_config\_t的池子，而下文描述的config\_string则保存在config\_string\_heap中。

## vnet\_config\_t



feature的配置结构体，里面保存了多个feature，当一个接口开启多个特性时，会在对应的feature中增加多个vnet\_config\_feature\_t。该结构体中还在config\_string中保存了feature之间的node关系。

接口开启feature时，将会调用vnet\_config\_add\_feature来添加feature，在首次添加时，会创建vnet\_config\_t这个结构体。

## vnet\_config\_feature\_t



config中的具体的feature，当为接口配置feature时，将会为该配置最终生成此数据结构，以区别与其他feature。

## config\_string



config\_string里保存了与feature相关的node之间的前后调度关系，主要是通过保存next node的索引而实现的，具体可见下面的例子。

## 数据关系图



上述四个结构体的关系图如上图所示，其结构层层包含，层级之间，采用vec变长数组。

# 关键函数

## vnet\_feature\_init

该函数负责feature的初始化，将注册的arc和feature做整合处理，具体处理见如下流程图：



其中，只有第5步做的事情较为复杂，其主要是通过vnet\_feature\_arc\_init函数对arc内的feature进行排序，而排序也只针对初始化时那些定义了runs\_before的进行排序处理，然后生成对应的feature\_index。

## vnet\_feature\_enable\_disable

该函数主要是将某个feature开启或关闭，内部生成对应的config配置数据，最重要的是config\_string，并将配置的各个feature按照顺序设置node矢量图（即node的next\_index），通过这个函数，可以将arc内使能的feature按照前后关系生成对应的node的矢量图。



生成config、对config内部的feature设置node矢量图等都是在find\_config\_with\_features中处理的。

## vnet\_feature\_arc\_start

该函数主要由arc内的start node调用，在start node中通过该函数，进入到已经组织好的feature的node的矢量图中。feature中的node处理完当前节点后，需要调用 vnet\_get\_config\_data 去设置下一个节点，保存在buffer的opaque信息里，以指导下一步进入哪个node。

比如在ipv4-unicast这个arc中，其start node之一是ip4-input，那么在ip4-input的处理函数中，将会调用此函数进入到feature的node矢量图中。

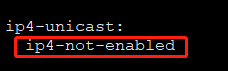
# 示例

通过show命令可以看到所有的arc和arc内的feature：

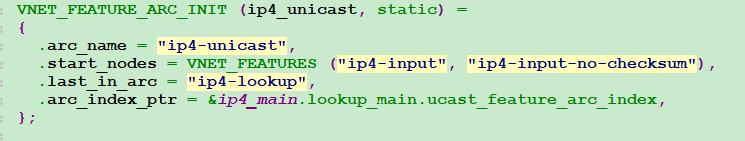
 通过set interface feature命令可以在接口上使能某个feature，通过show interface <ifname> features可以看到接口使能了哪些特性：

 下面以ip4-unicast这个arc为示例：

接口GigabitEthernet0/c/0未配置任何IP地址，并且状态为down，此时通过show命令看到ip4-unicast中默认启用了ip4-not-enabled



代码在注册时，起始节点为ip4-input和ip4-input-no-checksum，尾节点为ip4-lookup：



通过show命令看一下这三个节点：





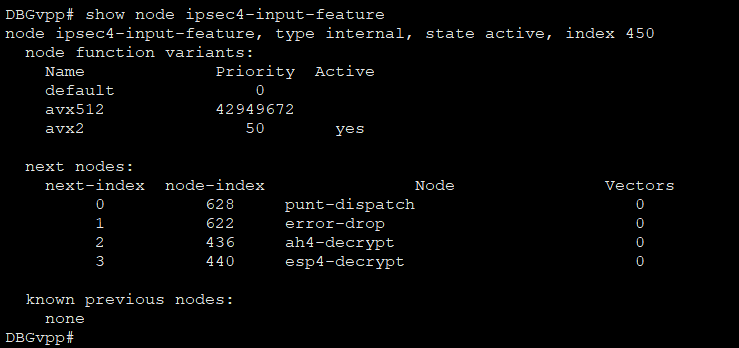
其中7这个值是对应于config\_string里存的feature在前一节点的next索引，在这里是ip4-not-enabled这个feature在start node里的next索引。

ip4-not-enabled这个特性是在接口创建时，就通过vnet\_feature\_enable\_disable 设置上去的。

下面通过CLI配置命令再增添一个，可以看到node关系对应的变化：

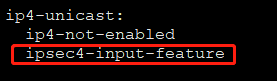
set interface feature GigabitEthernet0/c/0 ipsec4-input-feature arc ip4-unicast

在配置命令之前，先看一下ipsec4-input-feature这个节点：

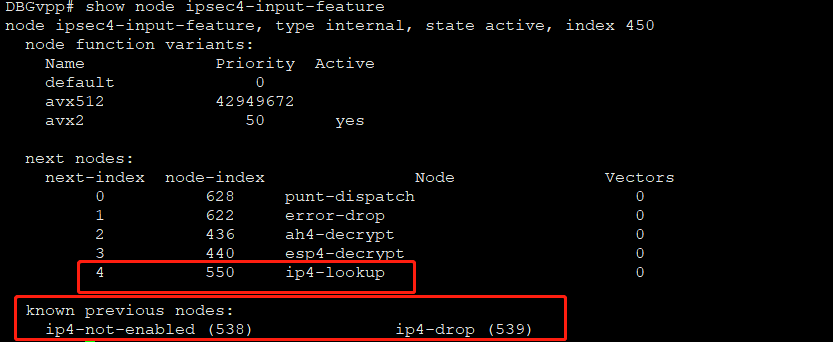


可以看到，目前其next-index有4个节点。

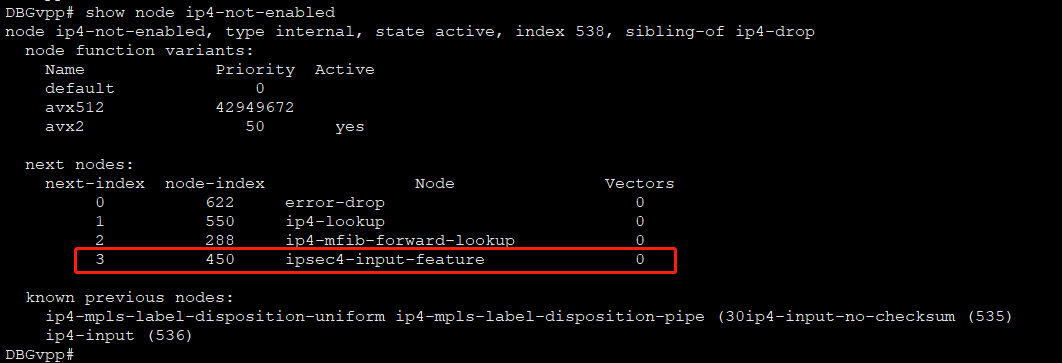
命令配置之后，看接口的特性已经增加了ipsec4-input-feature：



再重新看一下这个节点：



增加了一个next-index，指明下一个节点是ip4-lookup，同时把ip4-not-enabled置为了自己的前置节点。



对ip4-not-enabled这个节点，也增加了对应的next-index，指向了ipsec4-input-feature这个节点，这样就将特性的node传入到node处理图中。



config\_string中，指明了每个节点的顺序，当报文从ip4-input进来后，根据接口开启特性，在config\_string里查找对应的下一个node节点，理论上的node逻辑就应该如下：

 这里只是个逻辑示意，真正流程上，这样组合feature，报文在第二步就会丢弃掉。

# 总结

如上文所述，通过VPP的feature机制，可以在不改变VPP现有框架下，灵活地增加/删除业务功能。虚拟化网络设备相比于传统的硬件设备的一大优势就是灵活，而VPP的feature机制，更是适合虚拟化网络功能的开发，加上其性能优势，让VPP在虚拟化转发的诸多技术中占有一席之地。

# 参考

<https://blog.csdn.net/jqh9804/article/details/54772764>

<https://blog.csdn.net/sjin_1314/article/details/103022635>