# source-dpdk.c

DPDK模式下，和其它模式一样，注册了两个独有的模块。

|  |
| --- |
| TmModuleReceiveDpdkRegister(); |
| TmModuleDecodeDpdkRegister(); |

TmModuleReceiveDpdkRegister()

{

（1）ReceiveDpdkInit函数是线程初始化函数，为数据结构DpdkThreadVars申请了内存空间并对一些数据成员赋值，并将参数\*\*data指向这个空间。

（2）ReceiveDpdkLoop函数是循环获取数据包并将其转换成suricata能够处理的格式和结构。

struct rte\_mbuf \*bufs[16];存放数据包的相关信息和数据，dpdk传给上层应用程序的就是rte\_mbuf的地址。

const uint16\_t nb\_rx = rte\_eth\_rx\_burst(ptv->portid, ptv->queueid, bufs, 8);rte\_eth\_rx\_burst会调用eth\_em\_recv\_pkts或其它具体的驱动收包函。

rte\_pktmbuf\_mtod获得指向data的指针。

rte\_prefetch0报文处理时读取报文内容时添加了指令预取命令，提高性能。

DpdkProcessPacket包处理程序，程序将完善Packet结构，在函数中调用了rte\_mbuf\_to\_priv接口，功能是：Return the starting address of the private data area embedded in the given mbuf。

调用TmThreadsSlotProcessPkt，让其它slot处理包。

这两个函数会对包进行处理检查，并写日志。

}

TmModuleDecodeDpdkRegister()

{

（1）DecodeDpdkThreadInit线程初始化函数。

（2）DecodeDpdk调用解码函数DecodeEthernet。

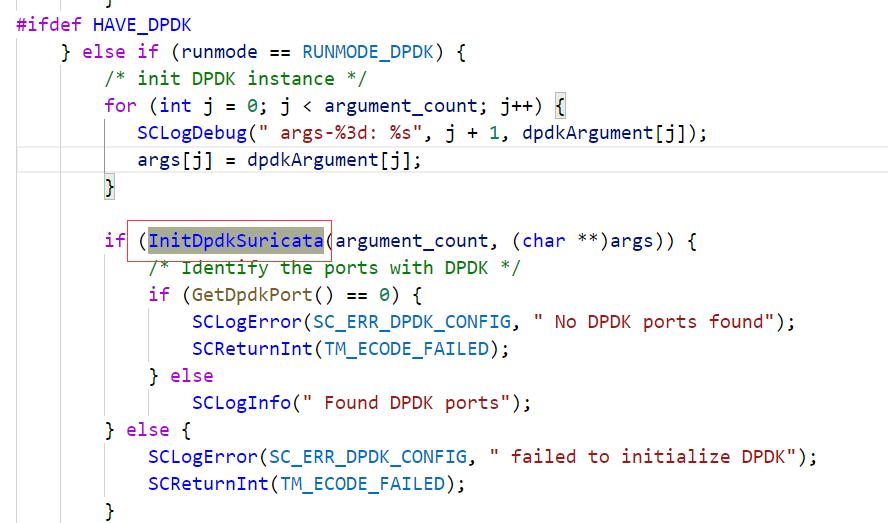
}

DpdkFowardPacket()如果匹配到的规则是drop就不转发，如果不是就转发。使用转发函数rte\_eth\_tx\_burst()。

DpdkBufferFowardPacket()如果匹配到的规则是drop就不转发，如果不是就转发。使用转发函数rte\_eth\_tx\_buffer()。

DpdkProcessPacket()处理数据包，填充Packet数据结构，

# suricata.c



调用了rte\_eal\_init。

# runmode-dpdk.c

RSS，为网卡设置多个队列，将数据包负载均衡到这些队列中，由多个CPU并行处理。

hash分流算法

ACL是通配符匹配算法，LPM是最长匹配算法。（分流算法或是创建路由表）

RunModeDpdkGetDefaultMode()默认是workers模式。

sym\_hash\_enable()和sym\_hash\_set()配置RSS的hash函数类型。

SetupDdpdkPorts()函数

{

首先为mbuf申请存储内存；

rte\_eth\_dev\_adjust\_nb\_rx\_tx\_desc()为网卡设置接收/发送描述符数目；

rte\_eth\_dev\_configure()的主要工作是分配接收/发送队列数组；

rte\_eth\_dev\_rss\_hash\_conf\_get()获得rss多队列的hash配置信息，如hash func；

rte\_eth\_rx\_queue\_setup()网卡接受多队列设置；

获得并设置MTU；

rte\_eth\_dev\_get\_supported\_ptypes()获得设备所支持的数据包类型；

rte\_eth\_dev\_set\_ptypes()设置设备处理的数据包类型；

rte\_eth\_add\_rx\_callback()为某个port的某个队列设置回调函数；

}

GetDpdkPort()中调用了rte\_eth\_dev\_count\_avail()函数，获得可以使用的port数目。

ValidateDpdkConfig()

{

GetDpdkPort();

SetupDdpdkPorts();

IPS模式需要固定的配置一进一出的port对，通过rte\_eth\_link\_get()获得port的相关链接信息，判断两个port的速度是否相同;

}

CreateDpdkAcl()创建ipv4和ipv6的ACL通配符匹配的ctx。这里好像并没有用到acl。

ParseDpdkYaml()读取suricata.yaml中dpdk字段的信息。

DpdkConfigParser()创建接口config结构并初始化，为tx发送队列创建一个buffer来缓存数据包，并且init。

RunModeDpdkWorkers()

{

DpdkGetRxThreads()获得需要的收包线程数目。

DumpGlobalConfig()输出所有的配置信息。

对每一个port通过 rte\_eth\_dev\_start()函数开启，然后通过rte\_eth\_link\_get()获得状态链接信息。

一个个的开启收包线程(workers模式，一个线程中包含所有的处理模块)

{

TmThreadCreatePacketHandler()为包处理线程创建TV实例；

DpdkConfigParser();

TmThreadSetCPU()线程绑定CPU；

通过TmModuleGetByName()和TmSlotSetFuncAppend()函数将ReceiveDPDK、DecodeDPDK、FlowWorker、RespondReject四个模块都入槽，这些模块将会按照顺序在线程内一次运行。

TmThreadSpawn()创建线程，TmThreadsSlotPktAcqLoop()是线程函数，里面会一次运行模块初始化函数。

}

}

dpdk\_mbuf\_ptype\_fiter\_nonip()函数和dpdk\_sw\_fiter\_nonip()函数都是发包函数，前者是在有硬件过滤器的情况下调用的，后者是在使用软件过滤器的情况下调用的。