关于OSPF所需API的回复

2016.03.02

您好！十分感谢您的耐心回复，但是可能由于我的表述问题还有一些方面没有搞清楚，所以希望能再沟通一下。为方便阅读，我会将我有疑问的部分标黄，并使用**蓝色加粗**字体来提出我的问题。感谢您的耐心解答！

## 数据发送

OSPF信令全部通过IP包传输，所以只要一个可以发送IP包的函数就可以。

/\*

Unit:芯片号，此处为0

Tx\_pkt：要发送的报文结构体

Cookie：如果发送报文需要回调函数，那么cookie 保存的回调函数的参数数据，一般为NULL

\*/

**请问此函数返回的int值的意义是什么？另外下述结构体中那么多值都需要设置么？还是只需要设置标红部分？如果是部分设置，那别的部分是初始化为0么？**

**int bcm\_tx(int unit, bcm\_pkt\_t \*tx\_pkt, void \*cookie)**

相关数据结构：

typedef struct bcm\_pkt\_s bcm\_pkt\_t;

typedef struct bcm\_pkt\_blk\_s {

uint8 \*data;

int len;

} bcm\_pkt\_blk\_t;

struct bcm\_pkt\_s {

bcm\_pkt\_blk\_t \*pkt\_data; /\* Pointer to array of data blocks. \*/

uint8 blk\_count; /\* Number of blocks in data array. \*/

uint8 unit; /\* Unit number. \*/

uint8 cos; /\* The local COS queue to use. \*/

uint8 prio\_int; /\* Internal priority of the packet. \*/

bcm\_vlan\_t vlan; /\* 802.1q VID or VSI or VPN. \*/

uint8 vlan\_pri; /\* Vlan tag priority . \*/

uint8 vlan\_cfi; /\* Vlan tag CFI bit. \*/

bcm\_vlan\_t inner\_vlan; /\* Inner VID or VSI or VPN. \*/

uint8 inner\_vlan\_pri; /\* Inner vlan tag priority . \*/

uint8 inner\_vlan\_cfi; /\* Inner vlan tag CFI bit. \*/

bcm\_color\_t color; /\* Packet color. \*/

int8 src\_port; /\* Source port used in header/tag. \*/

bcm\_trunk\_t src\_trunk; /\* Source trunk group ID used in

header/tag, -1 if src\_port set . \*/

uint16 src\_mod; /\* Source module ID used in header/tag. \*/

uint8 dest\_port; /\* Destination port used in header/tag. \*/

uint16 dest\_mod; /\* Destination module ID used in

header/tag. \*/

uint8 opcode; /\* BCM\_HG\_OPCODE\_xxx. \*/

bcm\_gport\_t dst\_gport; /\* Destination virtual port \*/

bcm\_gport\_t src\_gport; /\* Source virtual port \*/

bcm\_multicast\_t multicast\_group; /\* Destination multicast group. \*/

uint32 stk\_flags; /\* Stacking header flags. \*/

bcm\_pkt\_stk\_forward\_t stk\_forward; /\* Stacking header forwarding opcode. \*/

uint32 stk\_classification\_tag; /\* Stacking header classification tag. \*/

uint32 stk\_pkt\_prio; /\* Stacking header new packet priority. \*/

uint32 stk\_dscp; /\* Stacking header new DSCP. \*/

uint32 stk\_load\_balancing\_number; /\* Stacking header load balancing

number. \*/

bcm\_if\_t stk\_encap\_id; /\* Stacking header encapsulation ID for

remote packet replication. \*/

uint16 pkt\_len; /\* Packet length according to flags. \*/

uint16 tot\_len; /\* Packet length as transmitted or

received. \*/

bcm\_pbmp\_t tx\_pbmp; /\* Target ports. Bcm\_pbmp 其实就是一个int 类型 \*/

bcm\_pbmp\_t tx\_upbmp; /\* Untagged target ports. \*/

bcm\_pbmp\_t tx\_l3pbmp; /\* L3 ports. \*/

bcm\_port\_t pkt\_trace\_src\_port; /\* pkt\_trace\_src\_port \*/

uint8 pfm; /\* BCM\_PORT\_PFM\_xxx flags. \*/

uint32 rx\_reason; /\* Opcode from packet. \*/

bcm\_rx\_reasons\_t rx\_reasons; /\* Set of packet "reasons". \*/

uint32 rx\_path; /\* Rx path of packet. \*/

uint8 rx\_unit; /\* Local rx unit. \*/

uint8 rx\_port; /\* Local rx port; not in HG hdr. \*/

uint8 rx\_cpu\_cos; /\* CPU may get pkt on diff cos. \*/

uint8 rx\_untagged; /\* The packet was untagged on ingress. \*/

uint32 rx\_classification\_tag; /\* Classification tag. \*/

uint32 rx\_matched; /\* Field processor matched rule. \*/

bcm\_if\_t rx\_l3\_intf; /\* L3 egress object interface ID. \*/

bcm\_vlan\_action\_t rx\_outer\_tag\_action; /\* Outer-tag action applied to packet. \*/

bcm\_vlan\_action\_t rx\_inner\_tag\_action; /\* Inner-tag action applied to packet. \*/

uint32 rx\_timestamp; /\* Time stamp of time sync protocol

packets. \*/

uint32 rx\_timestamp\_upper; /\* Upper 32-bit of 64-bit timestamp of

OAM DM. \*/

uint32 timestamp\_flags; /\* Timestamp flags. \*/

void \*cookie; /\* User data for callback. \*/

void \*cookie2; /\* Additional user data for callback. \*/

bcm\_pkt\_cb\_f call\_back; /\* Callback function. \*/

uint32 flags; /\* BCM\_PKT\_F\_xxx flags. \*/

void \*next; /\* When linked into lists. \*/

int8 dma\_channel; /\* DMA channel used; may be -1. \*/

bcm\_pkt\_blk\_t \_pkt\_data; /\* For single block packets (internal). \*/

bcm\_pkt\_t \*\_last\_pkt; /\* To link to end of linked lists

(internal). \*/

void \*\_dv; /\* DV controlling this packet

(internal). \*/

int8 \_idx; /\* Packet's index in the DV for RX

(internal). \*/

bcm\_pkt\_t \*\_next; /\* For BCM layer linked lists

(internal). \*/

void \*alloc\_ptr; /\* Pointer for deallocation (internal). \*/

void \*trans\_ptr; /\* Transport pointer associated with

packet (internal). \*/

uint8 \_higig[16]; /\* HiGig header value (network byte

order). \*/

uint8 \_pb\_hdr[12]; /\* Pipe Bypass Header (network byte

order). \*/

uint8 \_sltag[4]; /\* SL tag value (network byte order). \*/

uint8 \_vtag[4]; /\* VLAN tag if not in packet (network

byte order). \*/

uint8 \_sbx\_rh[16]; /\* SBX Route Header & Shim. \*/

uint8 \_sbx\_hdr\_len; /\* Length of SBX header(s). \*/

uint8 \_dpp\_hdr[10]; /\* DPP header contents \*/

int \_dpp\_hdr\_type; /\* DPP header and extensions type \*/

int flow\_id; /\* Internal flow id. \*/

uint32 filter\_enable; /\* filters to be enabled. \*/

bcm\_pbmp\_t \_dv\_tx\_pbmp; /\* Actual pbmp assigned to tx

descriptor(internal). \*/

bcm\_pbmp\_t \_dv\_tx\_upbmp; /\* Actual upbmp assigned to tx

descriptor(internal). \*/

uint32 flags2; /\* BCM\_PKT\_F2\_xxx flags. \*/

uint8 oam\_replacement\_type; /\* OAM replacement type used in SOBMH

header. \*/

uint8 oam\_replacement\_offset; /\* OAM replacement offset used in SOBMH

header. \*/

uint16 oam\_lm\_counter\_index; /\* OAM LM counter index used in SOBMH

header. \*/

uint32 rx\_trap\_data; /\* Additional trap information \*/

void \*\_dcb; /\* Pointer for dcb. \*/

uint16 oam\_lm\_counter\_index\_2; /\* OAM Second LM counter index used in

SOBMH header. \*/

uint16 ma\_ptr; /\* OAM MA Pointer value. For BCM5645x,

this corresponds to endpoint group

index \*/

bcm\_pkt\_timestamp\_mode\_t timestamp\_mode; /\* OAM DM timestamp mode \*/

bcm\_pkt\_oam\_lm\_counter\_mode\_t counter\_mode\_1; /\* OAM LM counter-1 mode \*/

bcm\_pkt\_oam\_lm\_counter\_mode\_t counter\_mode\_2; /\* OAM LM counter-2 mode \*/

uint8 timestamp\_offset; /\* Offset to place the timestamp in the

packet. \*/

bcm\_rx\_decap\_tunnel\_t rx\_decap\_tunnel; /\* Tunnel Decap during packet rx \*/

bcm\_gport\_t src\_vport; /\* Source VPort (In-LIF) \*/

bcm\_gport\_t dst\_vport; /\* Destination VPort (Out-LIF) \*/

uint32 fwd\_hdr\_offset; /\* Distance (in bytes) to forwarding

header from start of packet \*/

int snoop\_cmnd; /\* Snoop command \*/

bcm\_gport\_t stk\_dst\_gport; /\* Stacking destination port \*/

uint32 stk\_route\_tm\_domains; /\* Bitmap of the traversed TM domains \*/

uint32 oam\_hdr\_offset; /\* Distance (in bytes) to OAM header

from start of packet \*/

uint8 oam\_lm\_replacement\_offset; /\* Replacement offset for LM counter in

Bytes \*/

bcm\_pkt\_rx\_oam\_type\_t rx\_oam\_pkt\_type; /\* OAM Pkt Type. \*/

bcm\_pkt\_oam\_counter\_t oam\_counter[BCM\_PKT\_OAM\_COUNTER\_MAX]; /\* OAM counter array. \*/

uint32 oam\_counter\_size; /\* Size of Counter array \*/

uint8 \_olp\_hdr[20]; /\* OLP Header (network byte order). \*/

bcm\_pkt\_dnx\_t dnx\_header\_stack[BCM\_PKT\_NOF\_DNX\_HEADERS]; /\* DNX Header stack \*/

uint8 dnx\_header\_count; /\* Number of DNX headers \*/

};

OSPF协议内的调用方式应该是什么（是否是在需要发送数据包时直接调用）？

【】OSPF 协议收发包有没有使用socket 发包？

**目前没有使用，现在理解的是直接调用您提供的接口送出接口。**

## 数据接收

同理，需要一个可以接收IP包的函数。

/\*

Unit:芯片号，此处为0

Name：注册函数标识名称

Callback：接收回调函数

Priority：标识该回调函数的优先级

Cookie：保存的回调函数的参数数据

Flags：中断和cos 相关的标记符

\*/

int bcm\_rx\_register( int unit, const char \*name, bcm\_rx\_cb\_f callback, uint8 priority, void \*cookie, uint32 flags)

使用上述函数注册接收函数.

typedef bcm\_rx\_t (\*bcm\_rx\_cb\_f)(int unit, bcm\_pkt\_t \*pkt, void \*cookie);

/\*接收中使用的pkt 结构体同发送\*/

## 参数设置

目前针对该环境编写的OSPF默认为点到点模式、单AREA、单AS，且未添加对组播、QoS等功能的扩展。

可以支持修改的参数有：Hello包发送间隔、报文重发间隔、邻居失效间隔（默认为4\*Hello包间隔）、链路开销等。

另外，OSPF需要获取本节点（路由器）各个端口的IP地址和子网掩码。

/\*

该函数是通过ip 地址和掩码获取路由的其他信息的，应该不满足当前需求。

这个对应应该是软件维护的，switch 芯片做的是对应哪些ip 从哪个端口出去，而自己端口本身没有ip地址。所以如果想知道从哪个端口出去能到达哪些ip的话，这边可以提供驱动接口，否则提供不了。

\*/

**这里我想我之前没有描述清楚。我们需要获取的是当前节点（路由器）自己的每个端口的IP地址和子网掩码，不需要知道端口出去后对端的信息，而前者应该不可能是路由协议来指定的（如果是由软件来维护，也应该由某地址分配协议在初始化路由协议前来指定）。**

**另外能否提供输入ip地址返回端口号以及通过端口号查找ip地址（以及子网掩码等信息，如果有的话）的接口函数。**

Int bcm\_esw\_l3\_route\_get(int unit, bcm\_l3\_route\_t \*info)

## 路由表更新

本OSPF协议中使用一个函数来直接更新存储于其依附的节点实例中的路由表对象。具体方法是：在协议初始化时将OSPF实例中的路由表指针（ForwardingTable\* forwardingTable）指向其所依附节点中的路由表对象（ForwardingTable forwardingTable）——这实际上是使OSPF实例和节点共用一张路由表（一个路由表实例）——然后在需要更新路由表时直接调用该实例的更新路由表方法（forwardingTable-> updateForwardingTable()），这将直接修改位于节点中的路由表对象。同时，IP层中也使用同样的方式来查找位于节点中的路由表，以此来达到多协议之间路由表内容的统一。

/\*

在switch 芯片内部有这样两种L3 table，一种是Host 表项，一般来说是和端口相连的是单个ip 的设备比如说pc，另一种是route ，一般来说是一个网段的也就是有子网掩码的比如说另一台交换机。所以对应的有两种更新分别为bcm\_l3\_host\_add 和bcm\_l3\_route\_add

\*/

int bcm\_l3\_host\_add(int unit, bcm\_l3\_host\_t \*info);

int bcm\_l3\_route\_add(int unit, bcm\_l3\_route\_t \*info);

**请问是否是将路由表项放入info中，通过调用上述函数可以把路由表项写入路由表中？下述结构体中那么多值都需要设置么？还是只需要设置标红部分？如果是部分设置，那别的部分是初始化为0么？事实上目前看来标红部分的某些项协议中应该都无法提供。**

相关结构体：

typedef struct bcm\_l3\_host\_s {

uint32 l3a\_flags; /\* See BCM\_L3\_xxx flag definitions. 指定IPv4 还是Ipv6地址\*/

uint32 l3a\_flags2; /\* See BCM\_L3\_FLAGS2\_xxx flag definitions. \*/

bcm\_vrf\_t l3a\_vrf; /\* Virtual router instance. \*/

bcm\_ip\_t l3a\_ip\_addr; /\* Destination host IP address (IPv4). Ipv4地址\*/

bcm\_ip6\_t l3a\_ip6\_addr; /\* Destination host IP address (IPv6). Ipv6地址\*/

bcm\_cos\_t l3a\_pri; /\* New priority in packet. \*/

bcm\_if\_t l3a\_intf; /\* L3 intf associated with this address. 对应的L3 interface\*/

bcm\_mac\_t l3a\_nexthop\_mac; /\* Next hop MAC addr. 下一跳mac地址\*/

bcm\_module\_t l3a\_modid; /\* Module ID packet is switched to. \*/

bcm\_port\_t l3a\_port\_tgid; /\* Port/TGID packet is switched to. 对应switch 芯片的物理出端口\*/

bcm\_port\_t l3a\_stack\_port; /\* Used if modid not local (Strata Only). \*/

int l3a\_ipmc\_ptr; /\* Pointer to IPMC table. \*/

int l3a\_lookup\_class; /\* Classification lookup class ID. \*/

bcm\_if\_t encap\_id; /\* Encapsulation index. \*/

bcm\_if\_t native\_intf; /\* L3 native interface (source MAC). \*/

} bcm\_l3\_host\_t;

typedef struct bcm\_l3\_route\_s {

uint32 l3a\_flags; /\* See BCM\_L3\_xxx flag definitions. 指定IPv4 还是Ipv6地址\*/

uint32 l3a\_flags2; /\* See BCM\_L3\_FLAGS2\_xxx flag

definitions. \*/

uint32 l3a\_ipmc\_flags; /\* Used for multicast route entry. See

BCM\_IPMC\_xxx flag definitions. \*/

bcm\_vrf\_t l3a\_vrf; /\* Virtual router instance. \*/

bcm\_ip\_t l3a\_subnet; /\* IP subnet address (IPv4). \*/

bcm\_ip6\_t l3a\_ip6\_net; /\* IP subnet address (IPv6). \*/

bcm\_ip\_t l3a\_ip\_mask; /\* IP subnet mask (IPv4). \*/

bcm\_ip6\_t l3a\_ip6\_mask; /\* IP subnet mask (IPv6). \*/

bcm\_if\_t l3a\_intf; /\* L3 interface associated with route. 对应的L3 interface\*/

bcm\_ip\_t l3a\_nexthop\_ip; /\* Next hop IP address (XGS1/2, IPv4). \*/

bcm\_mac\_t l3a\_nexthop\_mac; /\* Next hop MAC address. 下一跳mac地址\*/

bcm\_module\_t l3a\_modid; /\* Module ID. \*/

bcm\_port\_t l3a\_port\_tgid; /\* Port or trunk group ID. 对应switch 芯片的物理出端口\*/

bcm\_port\_t l3a\_stack\_port; /\* Used if modid is not local (Strata

Only). \*/

bcm\_vlan\_t l3a\_vid; /\* BCM5695 only - for per-VLAN def

route. \*/

bcm\_cos\_t l3a\_pri; /\* Priority (COS). \*/

uint32 l3a\_tunnel\_option; /\* Tunnel option value. \*/

bcm\_mpls\_label\_t l3a\_mpls\_label; /\* MPLS label. \*/

int l3a\_lookup\_class; /\* Classification class ID. \*/

bcm\_if\_t l3a\_expected\_intf; /\* Expected L3 Interface used for

multicast RPF check \*/

int l3a\_rp; /\* Rendezvous point ID \*/

bcm\_multicast\_t l3a\_mc\_group; /\* L3 Multicast group index \*/

} bcm\_l3\_route\_t;