Introduction of RomeDriver

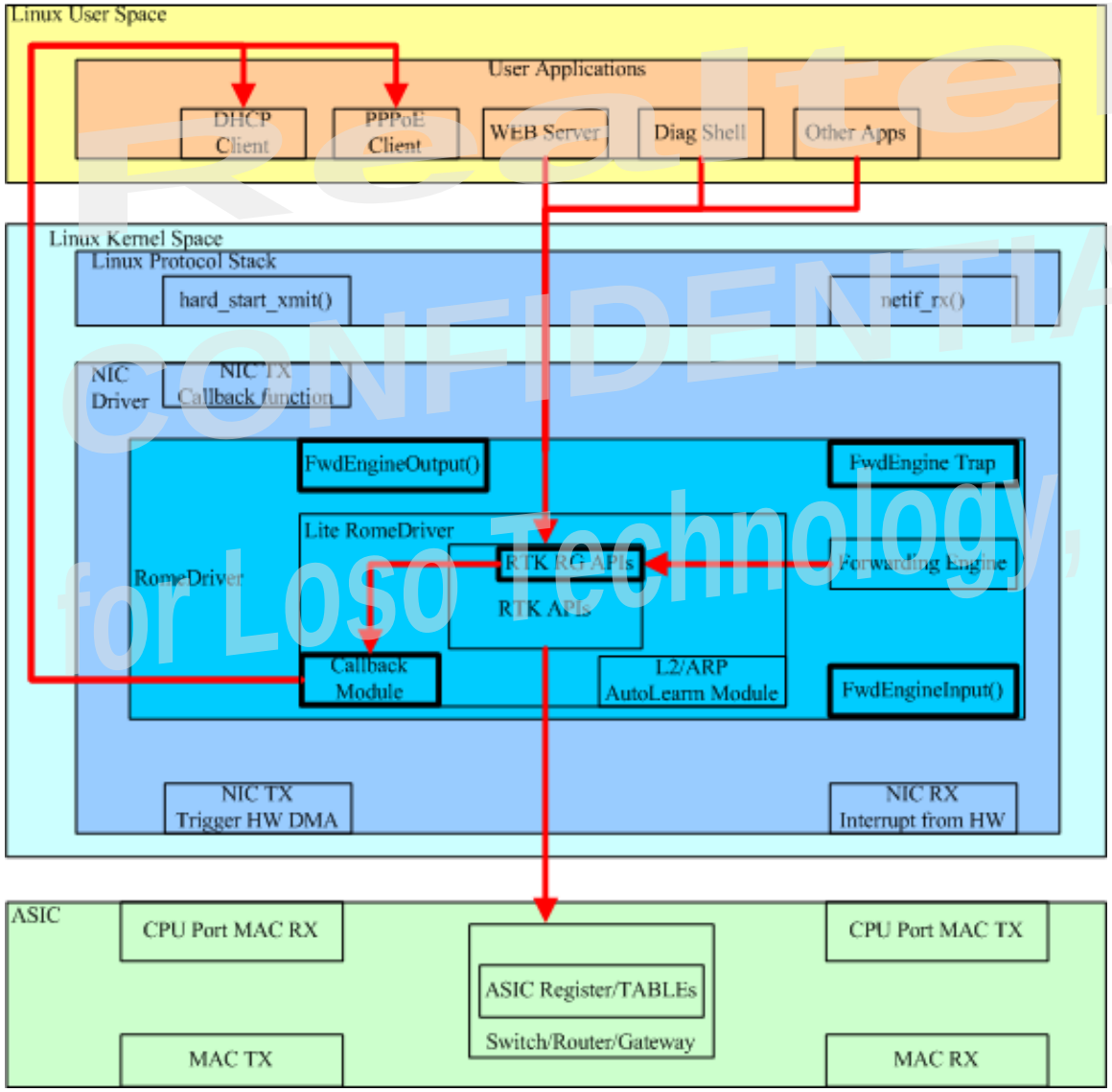
2014/01/09

Tsung-Ching Lin (Chuck)

Outline

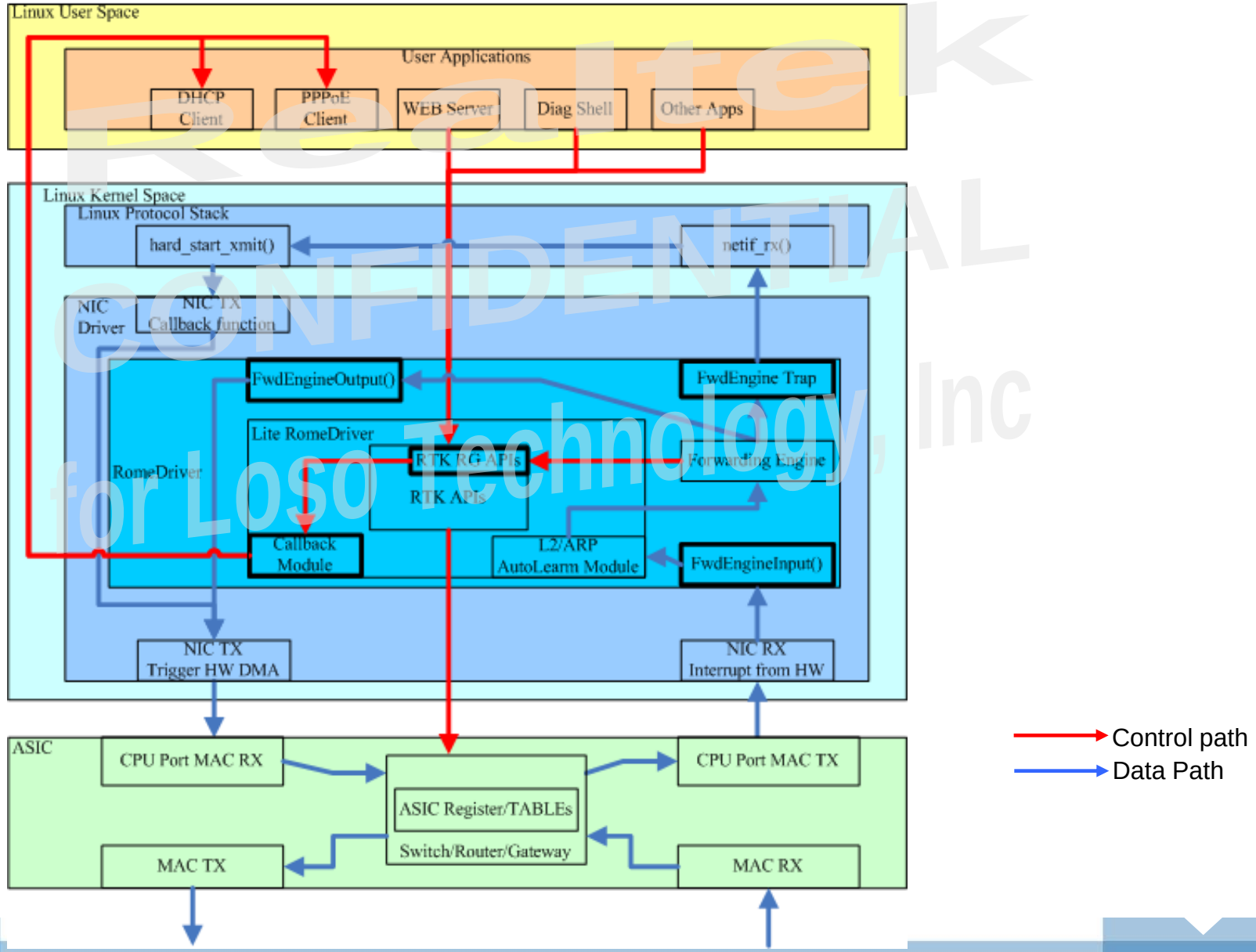
- System Architecture
- Callback & Diagshell
- Lan/Wan Interface Configuretion
- Debug Tools
- Forwarding Engine Packet Flow
- VLAN Configuretion
- ACL Configuretion
- Qos Configuretion
- Q&A

System Architecture



Control path

System Architecture



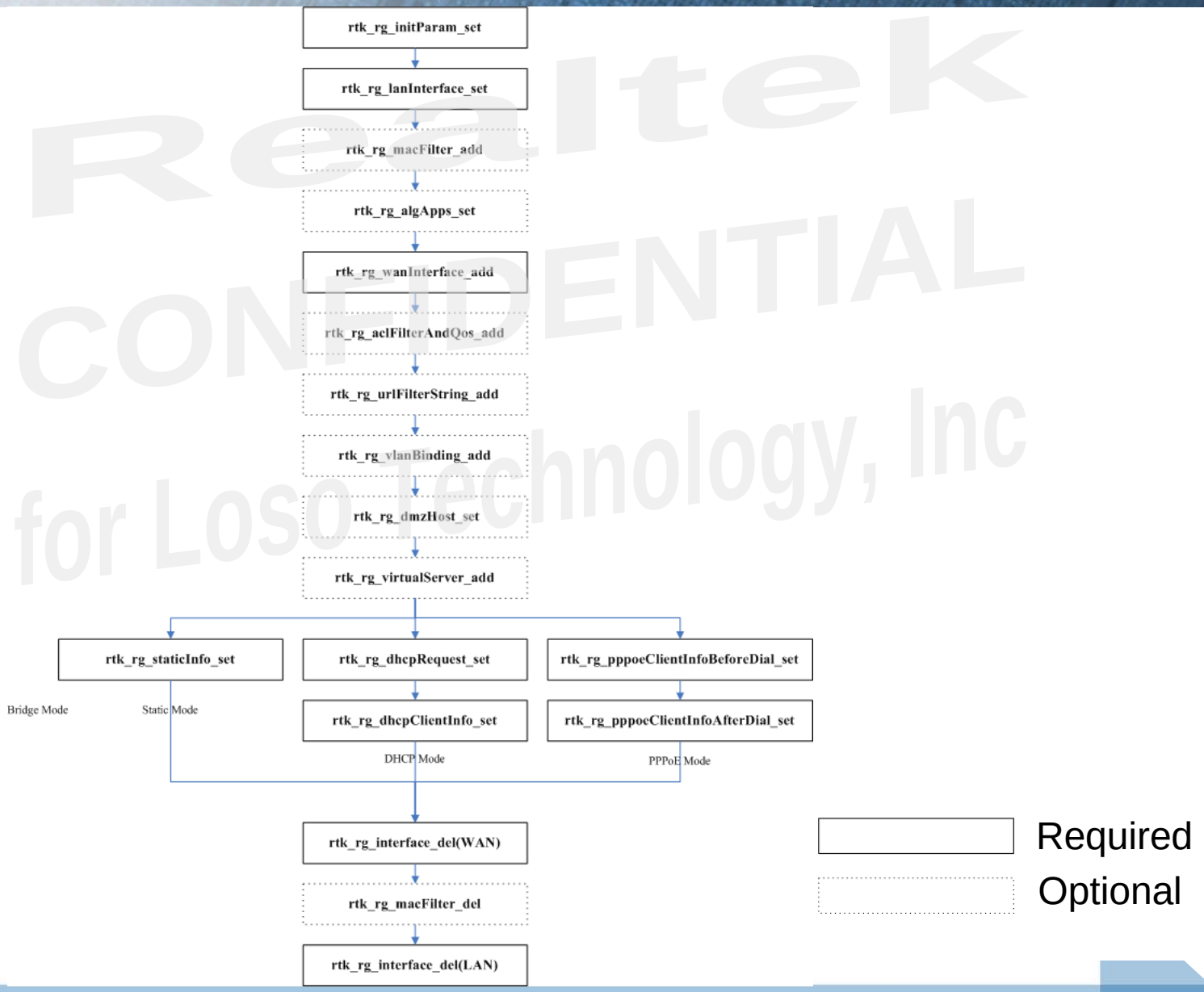
Callback

type	Hook point	Related API
p_initByHwCallBack	**initByHwCallBack**	rtk_rg_initParam_set
p_arpAddByHwCallBack	**arpAddByHwCallBack**	rtk_rg_arpEntry_add
p_arpDelByHwCallBack	**arpDelByHwCallBack**	rtk_rg_arpEntry_del
p_macAddByHwCallBack	**macAddByHwCallBack**	rtk_rg_macEntry_add
p_macDelByHwCallBack	**macDelByHwCallBack**	rtk_rg_macEntry_del
p_routingAddByHwCallBack	**routingAddByHwCallBack**	rtk_rg_lanInterface_add
p_routingDelByHwCallBack	**routingDelByHwCallBack**	rtk_rg_lanInterface_del
p_naptAddByHwCallBack	**naptAddByHwCallBack**	_rtk_rg_naptConnection_add
p_naptDelByHwCallBack	**naptDelByHwCallBack**	_rtk_rg_naptConnection_del
P_bindAddByHwCallBack	**bindingAddByHwCallBack**	rtk_rg_wanInterface_add
P_bindDelByHwCallBack	**bindingDelByHwCallBack**	rtk_rg_wanInterface_del
p_interfaceAddByHwCallBack	**interfaceAddByHwCallBack**	rtk_rg_lanInterface_add / rtk_rg_wanInterface_add
p_interfaceDelByHwCallBack	**interfaceDelByHwCallBack**	rtk_rg_interface_del
p_neighborAddByHwCallBack	**neighborAddByHwCallBack**	rtk_rg_neighborEntry_add
p_neighborDelByHwCallBack	**neighborDelByHwCallBack**	rtk_rg_neighborEntry_del
p_v6RoutingAddByHwCallBack	**v6RoutingAddByHwCallBack**	rtk_rg_lanInterface_add
p_v6RoutingDelByHwCallBack	**v6RoutingDelByHwCallBack**	rtk_rg_interface_del
p_pppoeBeforeDiagByHwCallBack	**pppoeBeforeDiagByHwCallBack**	rtk_rg_pppoeClientInfoBeforeDial_set
p_dhcpRequestByHwCallBack	**dhcpRequestByHwCallBack**;	rtk_rg_dhcpRequest_set

Diagshell

- A tool for access RG API directly
  - Enter "diag" to enter the application
  - rg {show | clear | set | add | del } API_KEYWORD PARAMETERS …

Lan/Wan Configuration



rtk_rg_initParam_set

- rtk_rg_initParam_set(rtk_rg_initParams_t *init_param)

//register user own callback

RTK.0> rg set callback interfaceAddByHwCallBack 0xFFFFFFFF

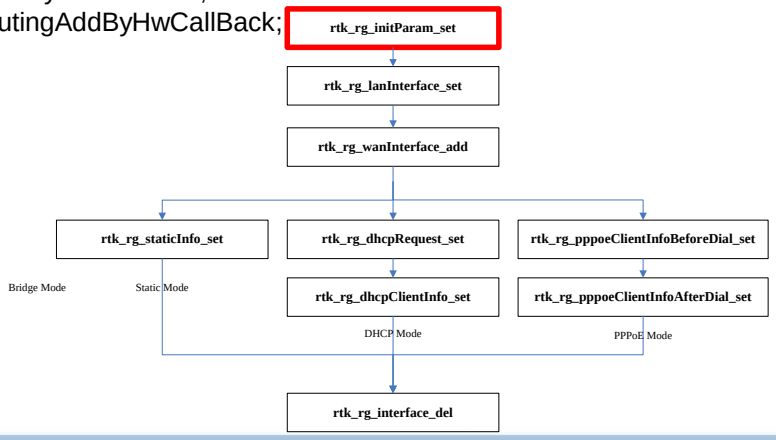
RTK.0> rg init callback igmpSnoopingEnable 1 macBasedTagDecision 1 wanPortGponMode 0

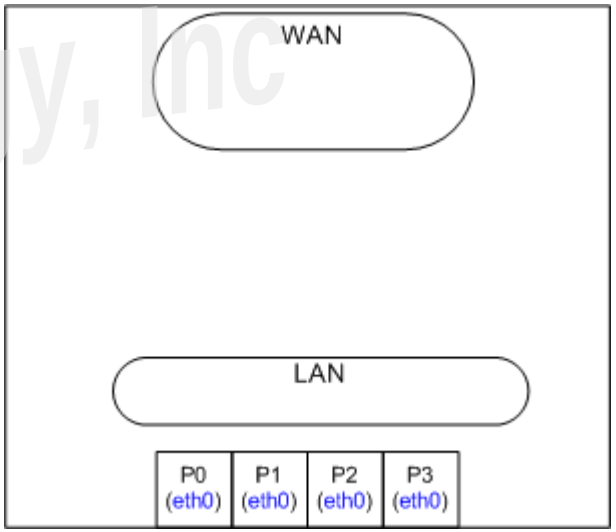
…

//register demo system callbacks

RTK.0> rg init callback default igmpSnoopingEnable 1 macBasedTagDecision 1 wanPortGponMode 0

```
typedef struct rtk_rg_initParams_s
{
   uint32 igmpSnoopingEnable:1;
   uint32 macBasedTagDecision:1;  //control DMAC2CVID per-port and forced state registers
   uint32 wanPortGponMode:1;      //control wan port is GPON mode or EPON/UTP mode

   p_initByHwCallBack initByHwCallBack;
   p_arpAddByHwCallBack arpAddByHwCallBack;
   p_arpDelByHwCallBack arpDelByHwCallBack;
   p_macAddByHwCallBack macAddByHwCallBack;
   p_macDelByHwCallBack macDelByHwCallBack;
   p_routingAddByHwCallBack routingAddByHwCallBack;
                  …
                  …
} rtk_rg_initParams_t;
```





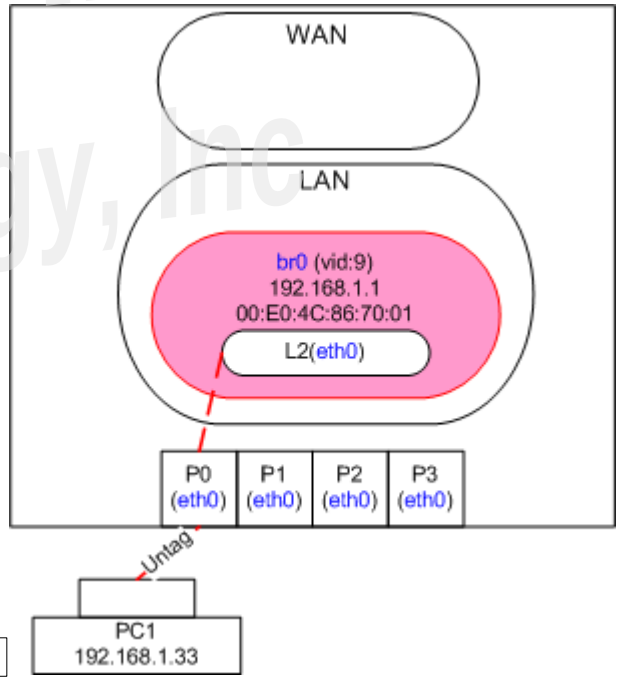
rtk_rg_lanInterface_add

- rtk_rg_lanInterface_add(rtk_rg_lanIntfConf_t *lan_info,int *intf_idx)

RTK.0> rg set lan-intf ip-version 0 gateway-mac 00:e0:4c:86:70:01 ip-addr 192.168.1.1 ip-mask 255.255.255.0 ipv6-addr 0::0 ipv6_network_mask_length 0 port-mask 0x4f untag-mask 0x4f intf-vlan_id 9 vlan-based-pri-enable disable mtu 1500 isIVL 0

RTK.0> rg add lan-intf entry

```
typedef struct rtk_rg_lanIntfConf_s
{
    rtk_rg_ip_version_t ip_version;    //0: ipv4, 1: ipv6, 2:both v4 & v6
    rtk_mac_t gmac;
    ipaddr_t ip_addr;
    ipaddr_t ip_network_mask;
    rtk_ipv6_addr_t ipv6_addr;
    int ipv6_network_mask_length;
    rtk_rg_portmask_t port_mask;
    rtk_rg_mac_portmask_t untag_mask;
    int intf_vlan_id;
    rtk_rg_enable_t vlan_based_pri_enable;
    int vlan_based_pri;
    int mtu;
    int isIVL;        //0: SVL, 1:IVL
} rtk_rg_lanIntfConf_t;
```

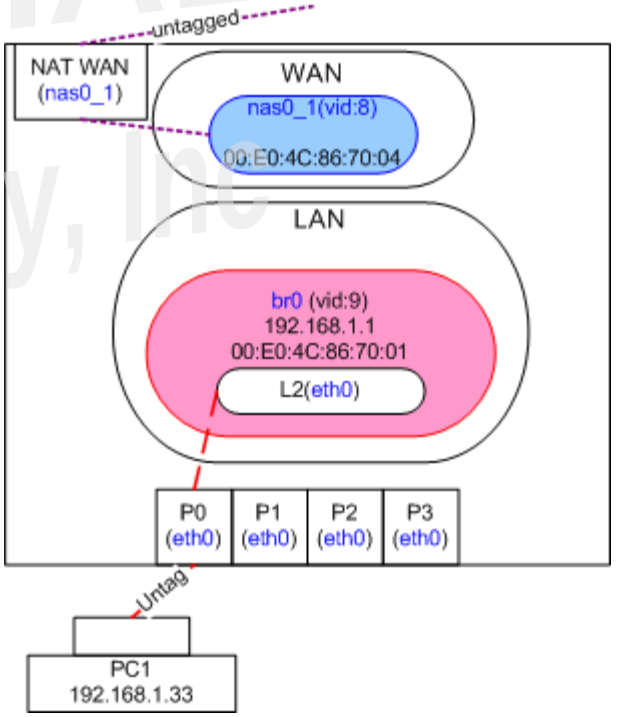


rtk_rg_wanInterface_add

- rtk_rg_wanInterface_add(rtk_rg_wanIntfConf_t *wanintf, int *wan_intf_idx)

RTK.0> rg set wan-intf **wan-type 0** gateway-mac 00:e0:4c:86:70:04 wan-port 4 port-binding-mask 0x0 egress-vlan-tag-on 0 egress-vlan-id 8 vlan-based-pri-enable disable isIVL 0
RTK.0> rg add wan-intf entry

```
typedef struct rtk_rg_wanIntfConf_s
{
    rtk_rg_wan_type_t wan_type; // 0:static  1:DHCP  2:PPPoE  3:Bridge
    rtk_mac_t gmac;
    rtk_rg_mac_port_idx_t wan_port_idx;
    rtk_rg_portmask_t port_binding_mask;
    int egress_vlan_tag_on;
    int egress_vlan_id;
    rtk_rg_enable_t vlan_based_pri_enable;
    int vlan_based_pri;
    int isIVL;        //0: SVL, 1:IVL
} rtk_rg_wanIntfConf_t;
```



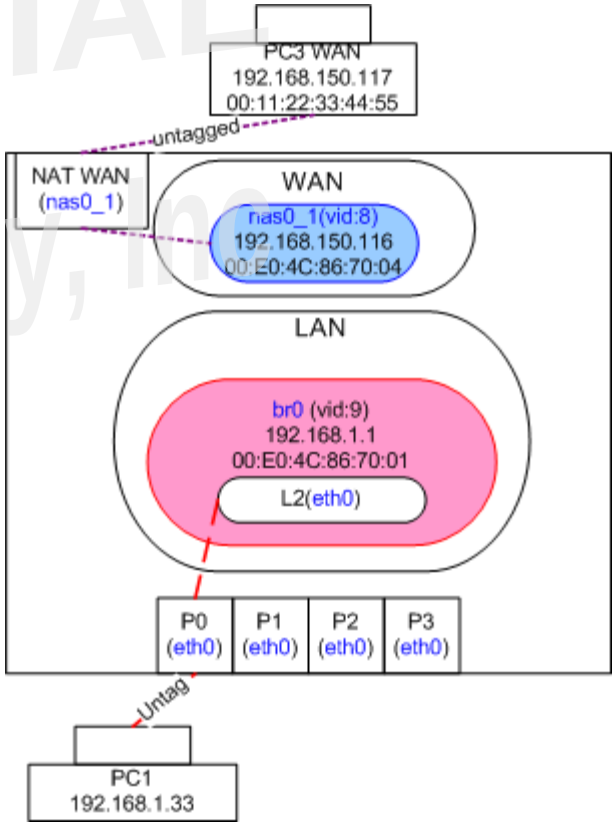
rtk_rg_staticInfo_set

- rtk_rg_staticInfo_set(int wan_intf_idx, rtk_rg_ipStaticInfo_t *static_info)

RTK.0> rg set wan-intf-static-info ip-version 0 napt_enable 1 ip_addr 192.168.150.116 ip_network_mask 255.255.255.0 ipv4_default_gateway_on 1 gateway_ipv4_addr 192.168.150.117 mtu 1500 gw_mac_auto_learn_for_ipv4 0 gateway_mac_addr_for_ipv4 00:11:22:33:44:55

RTK.0> rg add wan-intf-static-info intf-index 1

```
typedef struct rtk_rg_ipStaticInfo_s
{
    rtk_rg_ip_version_t ip_version;  //0: ipv4, 1: ipv6, 2:both v4 & v6
    int napt_enable;                 // L3 or L4
    ipaddr_t ip_addr;
    ipaddr_t ip_network_mask;
    int ipv4_default_gateway_on;  //1:should set default route, 0:otherwise
    ipaddr_t gateway_ipv4_addr;
    rtk_ipv6_addr_t ipv6_addr;
    int ipv6_mask_length;
    int ipv6_default_gateway_on;  //1:should set default route, 0:otherwise
    rtk_ipv6_addr_t gateway_ipv6_addr;
    int mtu;
    int gw_mac_auto_learn_for_ipv4;
    int gw_mac_auto_learn_for_ipv6;
    rtk_mac_t gateway_mac_addr_for_ipv4;
    rtk_mac_t gateway_mac_addr_for_ipv6;
} rtk_rg_ipStaticInfo_t;
```



Hwnat Tables

Network Interface Table

IDX	VALID	VLAND_ID	GATEWAY_MAC	MAC_MASK	EN_ROUTING	MTU
0	1	9(lan_gate)	00:e0:4c:86:70:01	NO_MASK(7)	1	1500
1	1	8(wan_gate)	00:e0:4c:86:70:04	NO_MASK(7)	1	1500

ARP Table

IDX	VALID	NH_L2IDX
33	1	150,0
372	1	353,0

L3 Routing Table

IDX	IP/MASK	VALID	TYPE(ARP) / TYPE(NH)	INT	DEST_NETIF_IDX / NH_ALGO	ARP_ADDR_START / NH_ADDR_START	ARP_ADDR_END / NH_NUM
0	192.168.1.0/24	1	ARP	1	0	0(0)	63(255)
1	192.168.150.0/24	1	ARP	1	1	64(256)	127(511)
7	0.0.0.0/0	1	NH	0	PER-SIP		

L2 Table

IDX	MAC	VLAN_ID	EGRESS_PORT
353,0	00-11-22-33-44-55	8	PON
150,0	00-11-22-33-44-00	9	PORT0

NextHop Table

IDX	TYPE	IFIDX	PPPIDX	NH_L2IDX
0	ETH	1	1	150,0

Internal External IP Table

IDX	INT_IP	EXT_IP	VALID	TYPE	NH_IDX	PRI_VALID	PRIORITY
0	0	192.168.150.116	1	NAPT	0	0	0

Create a Flow:
192.168.1.100:0x1234 =>10.0.0.1:0x80

NAPT Table

IDX	HASH_IDX	VALID	PRI_VALID	PRIORITY
1	468[192.168.1.100:0x1234, 10.0.0.1:0x80]	1	0	0

NAPTR Table

IDX	INT_IP	INT_PORT	REMOTE_HASH	EXT_IP_IDX	EXT_PORT_LSB	TCP	VALID	PRI_VALID	PRIORITY
468	192.168.1.100	0x1234	2689[10.0.0.1:0x80]	1	0x4321	1	2(NAPT)	0	0

Debug Tools

- **cat proc/dump/[TABLES]**
  - cat proc/dump/netif

```
# cat proc/dump/netif
>>ASIC Netif Table:

  [0]-vid[9] 00:e0:4c:86:70:01  L3/4 HW acc enabled
      7 MAC Addresses, MTU 1500 Bytes
                Untag member ports:0 1 2 3 6
                Active member ports:0 1 2 3 6

  [1]-vid[8] 00:e0:4c:86:70:04  L3/4 HW acc enabled
      7 MAC Addresses, MTU 1502 Bytes
                Untag member ports:0 1 2 3 4 5
                Active member ports:4 6
```

Lan Interface

Wan Interface

  - cat proc/dump/l3

```
# cat proc/dump/l3
>>L3 Routing Table:
[0] Valid 192.168.1.0/24 LAN RTLAN
            [ARP PROCESS]: NETIF(0) ARPSTA(0) ARPEND(63)

[1] Valid 192.168.150.0/24 WAN RTWAN
            [ARP PROCESS]: NETIF(1) ARPSTA(64) ARPEND(127)

[2]  (Invalid)
[3]  (Invalid)
[4]  (Invalid)
[5]  (Invalid)
[6] Valid 10.253.253.0/30 WAN RTLAN
            [CPU PROCESS]

[7] Valid 0.0.0.0/0 WAN RTWAN
            [NxtHop PROCESS]: NHSTA(0) NHNUM(1) NHNXT(0) NHALGO(PER-SIP) IPDOMAIN(6)
```

Lan Interface Route

Wan Interface Route

Default Route

Debug Tools

- **echo 0x4 > proc/rg/debug_level**

```
# echo 0x4 > proc/rg/debug_level
RomeDriver Debug level=0x4 [0x1:DEBUG=off][0x2:FIXME=off][0x4:CALLBACK=on][0x8:TRACE=off][0x10:ACL=off]
[0x20:WARN=off][0x40:TRACE_DUMP=off]
#
```

  - Example: List callback commands when add wan interface (p_interfaceAddByHwCallBack interfaceAddByHwCallBack;)

```
RTK.0> rg set wan-intf-static-info ip-version 0 napt_enable 1 ip_addr 192.168.150.116 ip_network_mask 255.255.255.0
ipv4_default_gateway_on 1 gateway_ipv4_addr 192.168.150.117 mtu 1500 gw_mac_auto_learn_for_ipv4 0 gateway_mac_addr_f
or_ipv4 00:e0:01:02:03:04
RTK.0>
RTK.0> rg add wan-intf-static-info intf-index 1
[rg callback]CMD:/bin/ash -c echo 4 nas0 > /proc/rtl8686gmac/dev_port_mapping port 4 assign to nas0
nas0 -> 0x10

[rg callback]CMD:/bin/ifconfig nas0 up
[rg callback]CMD:/bin/ethctl remsmux bridge nas0 nas0_1 smux_unregister_device remove smux dev nas0_1

[rg callback]CMD:/bin/ethctl addsmux ipoe nas0 nas0_1 napt <6>nas0_1 (): not using net_device_ops yet

[rg callback]CMD:/bin/ifconfig nas0_1 hw ether 00e04c867004
[rg callback]CMD:/bin/ifconfig nas0_1 up
[rg callback]CMD:/bin/ifconfig nas0_1 mtu 1500
[rg callback]CMD:/bin/route add default gw 192.168.150.117  [Exec Fialed],ret=256 @_rtk_rg_interfaceAddByHwCallBack,
line:1085
[rg callback]CMD:/bin/ash -c echo 1 > /proc/sys/net/ipv4/ip_forward
[rg callback]CMD:/bin/ifconfig nas0_1 192.168.150.116 netmask 255.255.255.0 broadcast 192.168.150.255
[rg callback]CMD:/bin/iptables -t nat -A POSTROUTING -o nas0_1 -j SNAT --to-source 192.168.150.116
intf[0] valid==1
add static info to interface[1] success.
RTK.0>
```

Debug Tools

- **cat /proc/rg/hwnat**
  - Show hwnat status

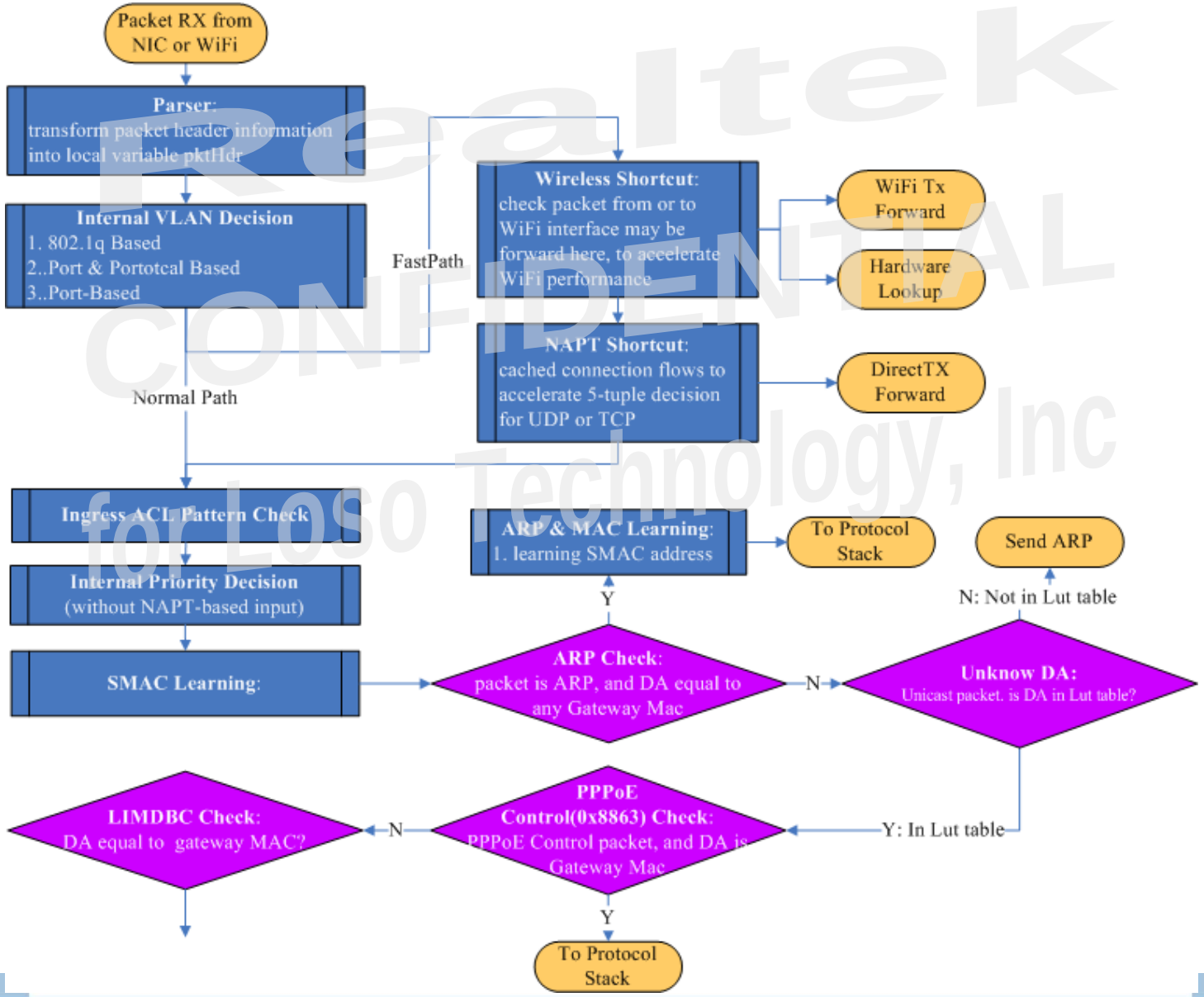
```
# cat proc/rg/hwnat
1:hwnat ENABLED, fwdEngine ENABLED
```

- **echo 0 > proc/rg/hwnat**
  - disable hwnat, packet will trap to fwdEngine directly.

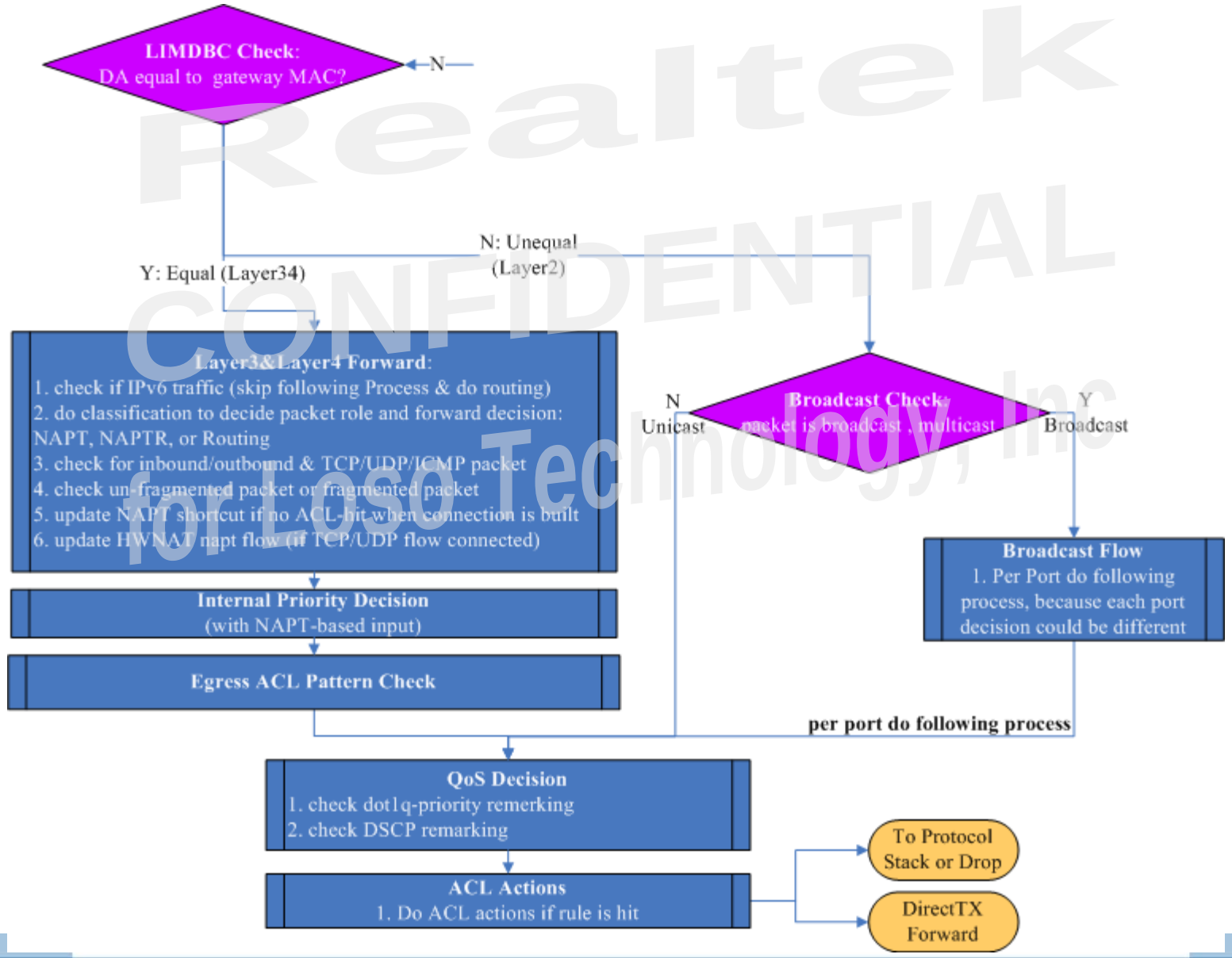
```
# echo 0 > proc/rg/hwnat
# cat proc/dump/acl
-------------- ACL TABLES ---------------
  --- ACL TABLE[0] ---
      valid:1
      active portmask:0x3f
      templateIdx:0
      action bits:[FWD][PRI]
      [FWD_ACTIDX:3(Trap to ACL Trap port)] portMask:0x0
      [PRI_ACTIDX:0(ACL Priority)] aclPri:0 dot1p:0 dscp:0 nexthop:0
```

- **echo 1 > proc/rg/hwnat**
  - enable hwnat
- **echo 2 > proc/rg/hwnat**
  - disable hwnat, and packet will trap to protocal-stack directly

Forwarding Engine Packet Flow(1)



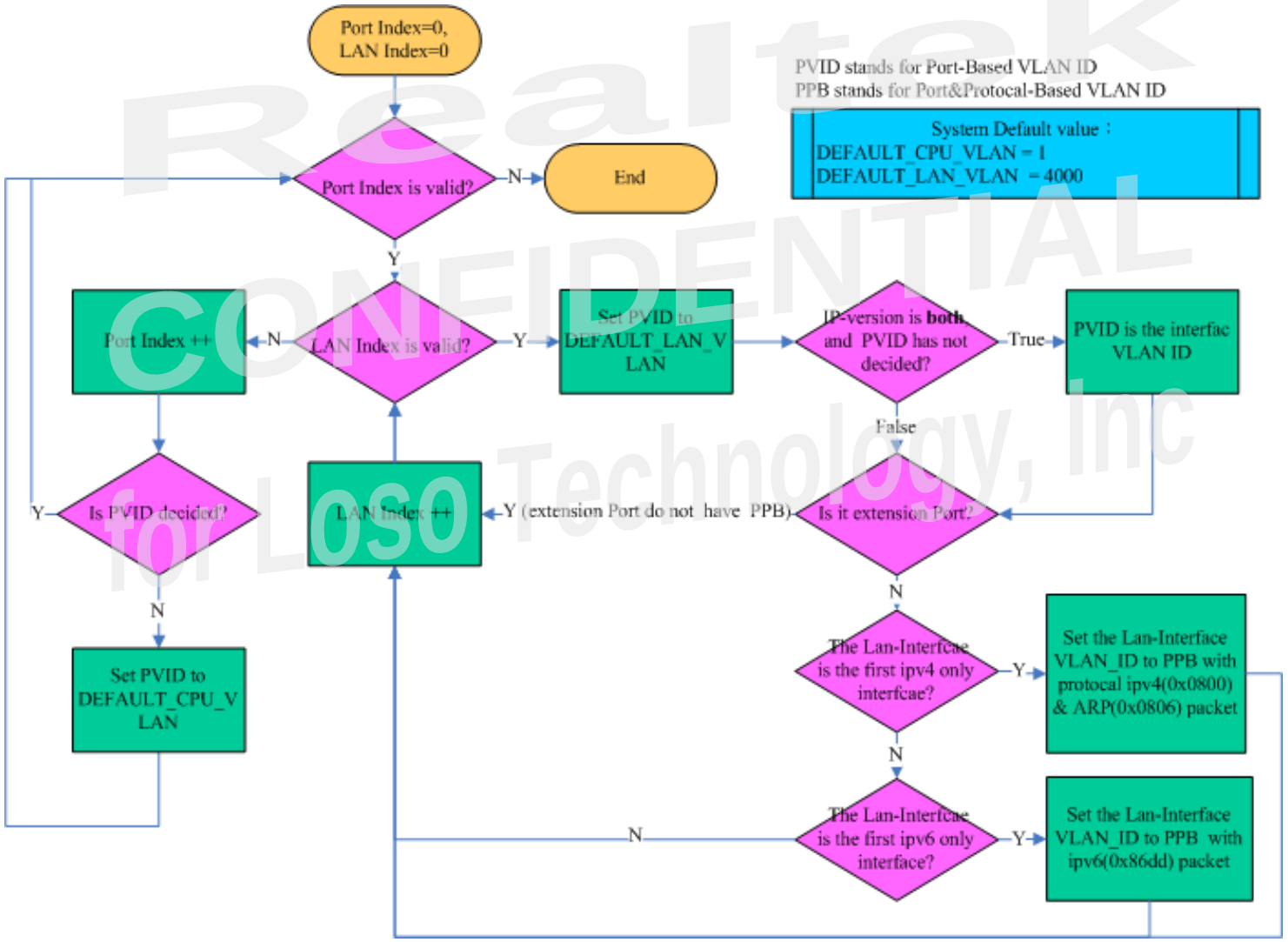
Forwarding Engine Packet Flow(2)



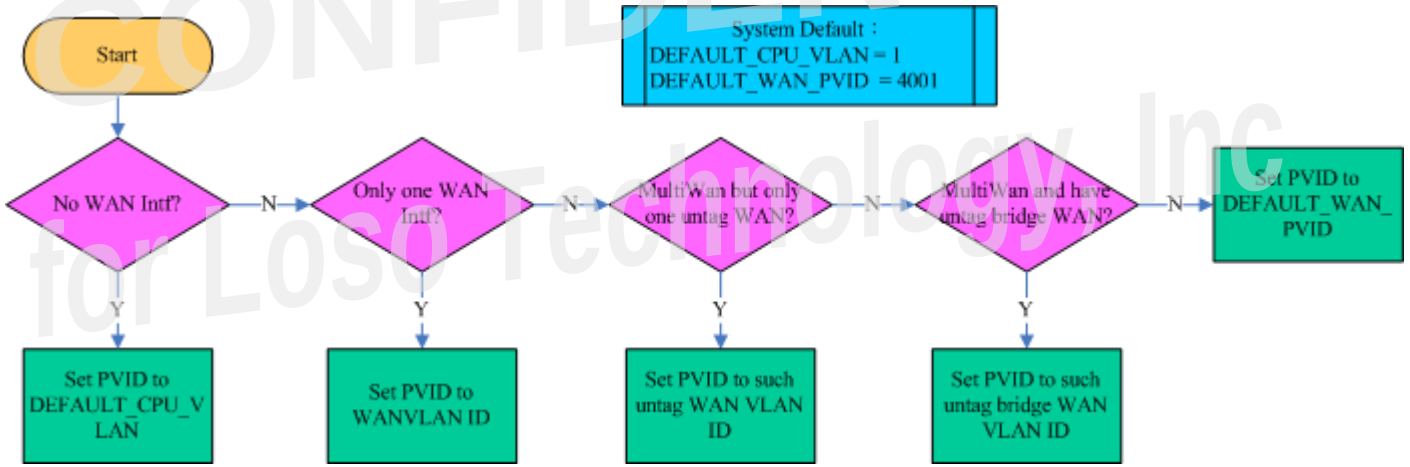
VLAN Configuration

- System Default VLAN
  - DEFAULT_CPU_VLAN(1): all mac port
  - DEFAULT_LAN_VLAN (4000): all LAN ports+CPU port+BridgeWan port
  - DEFAULT_WAN_PVID(4001): WAN port + CPU port
- Internal Vlan Decision
  - 802.1q Tag > Port&Portocal VLAN(PPBVID) >Port-Based VLAN(PVID)
- Per Port PVID & PPBVID Decision by RG
  - PVID can be set by API: rtk_rg_portBasedCVlanId_set

LAN Port PVID & PPBVID Decision



WAN Port PVID Decision



ACL Configuration

- RG ACL Configuration
  - API: rtk_rg_aclFilterAndQos_add

- Example for adding rule:

  [Type] Pure Ingress Packet Filter.

  [Patterns] IngressPort=port#0 & port#1,
  SIP=192.168.1.2,
  SMAC=00:11:22:33:44:55

  [Qos actions] dop1p-priority remarking to 2

- RG ⇔ ASIC Mapping

rtk_rg_aclFilterAndQos_add

- rtk_rg_aclFilterAndQos_add(rtk_rg_aclFilterAndQos_t *acl_filter, int *acl_filter_idx)

step1:
Choosing ACL type.

step2
Choosing ACL patterns.
& Set Patterns value.

step3:
Choosing ACL action type.

step4:
If ACL action type is QoS
Choosing qos actions
& set action value

```
typedef struct rtk_rg_aclFilterAndQos_s
{
  rtk_rg_acl_fwding_type_direction_t fwding_type_and_direction;
  unsigned int filter_fields;
  rtk_rg_portmask_t ingress_port_mask;
  int ingress_intf_idx;
  int egress_intf_idx;
  int ingress_ethertype
  rtk_mac_t ingress_smac;
  rtk_mac_t ingress_dmac;
  ipaddr_t ingress_src_ipv4_addr_start;
  ipaddr_t ingress_src_ipv4_addr_end;
  ipaddr_t ingress_dest_ipv4_addr_start;
  ipaddr_t ingress_dest_ipv4_addr_end;
  ...
  unsigned short int ingress_src_l4_port_start;
  unsigned short int ingress_src_l4_port_end;
  unsigned short int ingress_dest_l4_port_start;
  unsigned short int ingress_dest_l4_port_end;
  ...
  rtk_rg_acl_action_type_t action_type;
  rtk_rg_acl_qos_action_t qos_actions;

  unsigned char action_dot1p_remarking_pri;
  unsigned char action_ip_precedence_remarking_pri;
  unsigned char action_dscp_remarking_pri;
  unsigned char action_queue_id;
  unsigned char action_share_meter;
  unsigned char action_stream_id_or_llid;
  unsigned char action_acl_priority;
  rtk_rg_cvlan_tag_action_t action_acl_cvlan;
  rtk_rg_svlan_tag_action_t action_acl_svlan;
  unsigned char action_policy_route_wan;
} rtk_rg_aclFilterAndQos_t;
```

step1

```
typedef enum rtk_rg_acl_fwding_type_direction_e{
  ACL_FWD_TYPE_DIR_INGRESS_ALL_PACKET=0,
  ACL_FWD_TYPE_DIR_INGRESS_OR_EGRESS_L34_UP_DROP,
  ACL_FWD_TYPE_DIR_INGRESS_OR_EGRESS_L34_DOWN_DROP,
  ACL_FWD_TYPE_DIR_INGRESS_OR_EGRESS_L34_UP_STREAMID,
  ACL_FWD_TYPE_DIR_INGRESS_OR_EGRESS_L34_UP_STREAMID_CVLAN_SVLAN,
  ACL_FWD_TYPE_DIR_INGRESS_OR_EGRESS_L34_DOWN_CVLAN_SVLAN,
}rtk_rg_acl_fwding_type_direction_t;
```

step2

```
typedef enum rtk_rg_acl_filter_fields_e
{
  INGRESS_PORT_BIT=0x1,
  INGRESS_INTF_BIT=0x2,
  EGRESS_INTF_BIT=0x4,
  INGRESS_ETHERTYPE_BIT=0x8,
  ...
} rtk_rg_acl_filter_fields_t;
```

step3

```
typedef enum rtk_rg_acl_action_type_e
{
  ACL_ACTION_TYPE_DROP=0,
  ACL_ACTION_TYPE_PERMIT,
  ACL_ACTION_TYPE_TRAP,
  ACL_ACTION_TYPE_QOS, /* qos_actions valid*/
  ACL_ACTION_TYPE_TRAP_TO_PS,
  ACL_ACTION_TYPE_POLICY_ROUTE,
  ACL_ACTION_TYPE_END
} rtk_rg_acl_action_type_t;
```

step4

```
typedef enum rtk_rg_acl_filter_and_qos_action_e
{
  ACL_ACTION_NOP_BIT=(1<<0),
  ACL_ACTION_1P_REMARKING_BIT=(1<<1),
  ACL_ACTION_IP_PRECEDENCE_REMARKING_BIT=(1<<2),
  ACL_ACTION_DSCP_REMARKING_BIT=(1<<3),
  ACL_ACTION_QUEUE_ID_BIT=(1<<4),
  ACL_ACTION_SHARE_METER_BIT=(1<<5),
  ACL_ACTION_STREAM_ID_OR_LLID_BIT=(1<<6),
  ACL_ACTION_ACL_PRIORITY_BIT=(1<<7),
  ACL_ACTION_ACL_CVLANTAG_BIT=(1<<8),
  ACL_ACTION_ACL_SVLANTAG_BIT=(1<<9),
  ACL_ACTION_END=(1<<10),
} rtk_rg_acl_qos_action_t;
```

ACL Qos Example(Diagshell)

- Set ACL

```
#diag
RTK.0>rg set acl-filter fwding_type_and_direction 0
RTK.0>rg set acl-filter pattern ingress_port_mask 0x3
RTK.0>rg set acl-filter pattern ingress_src_ipv4_addr_start 192.168.1.2
ingress_src_ipv4_addr_end 192.168.1.2
RTK.0>rg set acl-filter pattern ingress_smac 00:11:22:33:44:55
RTK.0>rg set acl-filter action action_type 3
RTK.0>rg set acl-filter action qos action_dot1p_remarking_pri 2
RTK.0>rg add acl-filter entry
add acl-filter entry[0] success!
```

- Get ACL

```
# cat proc/dump/acl_rg
```

```
# cat proc/dump/acl_rg
=======================RG_ACL[0]==========================
[hw_acl_start:1(continue:2) hw_cf_start:0(continue:0)]
[Using range tables]:
[Patterns]:
filter_fields:0x2041
ingress_port_mask:0x3
ingress_smac: 00:11:22:33:44:55
ingress_sip_low_bound: 192.168.1.2  ingress_sip_up_bound:  192.168.1.2
l4-protocal: not care
[Actions]:
action type: ACL_ACTION_TYPE_QOS
qos_actions_bits: 0x2
dot1p_remarking: 2
#
```

ACL Qos Example(Source Code)

```
rtk_rg_aclFilterAndQos_t acl_filter;
int acl_filter_idx;
memset(&acl_filter,0,sizeof(rtk_rg_aclFilterAndQos_t));

acl_filter.fwding_type_and_direction=ACL_FWD_TYPE_DIR_INGRESS_ALL_PACKET;
acl_filter.filter_fields|=(INGRESS_PORT_BIT|INGRESS_IPV4_SIP_RANGE_BIT|INGRESS_SMAC_BIT);
//Ingress Port = port#0~port#1
acl_filter.ingress_port_mask.portmask=((1<<RTK_RG_MAC_PORT0)|(1<<RTK_RG_MAC_PORT1));
//SMAC = 00:11:22:33:44:55
acl_filter.ingress_smac.octet[0]=0x00;
acl_filter.ingress_smac.octet[1]=0x11;
acl_filter.ingress_smac.octet[2]=0x22;
acl_filter.ingress_smac.octet[3]=0x33;
acl_filter.ingress_smac.octet[4]=0x44;
acl_filter.ingress_smac.octet[5]=0x55;
//Src IP=192.168.1.2
acl_filter.ingress_src_ipv4_addr_start=0xc0a80102;
acl_filter.ingress_src_ipv4_addr_end=0xc0a80102;

acl_filter.action_type=ACL_ACTION_TYPE_QOS;
acl_filter.qos_actions|=ACL_ACTION_1P_REMARKING_BIT;
acl_filter.action_dot1p_remarking_pri = 2;

if(rtk_rg_aclFilterAndQos_add(&acl_filter, &acl_filter_idx)) return -1;
```

RG ACL to ASIC

ACL Templates

IDX	Pattern0	Pattern1	Pattern2	Pattern3	Pattern4	Pattern5	Pattern6	Pattern7
0	DA[15:0]	DA[31:16]	DA[47:32]	EXTPMSK	SA[15:0]	SA[31:16]	SA[47:32]	ETHERTYPE
1	L4_DPORT	SIP[15:0]	SIP[31:16]	L4_SPORT	IP_PROTO	PORTRANGE	DIP[15:0]	DIP[31:16]
2	CTAG	GEM/LLID	STAG	SIPv6[127:112]	DSCP	SIPv6[31:16]	SIPv6[15:0]	SessionID
3	DIPv6[127:112]	DIPv6[111:96]	DIPv6[95:80]	DIPv6[79:64]	DIPv6[63:48]	DIPv6[47:32]	DIPv6[31:16]	DIPv6[15:0]

ACL Rules

IDX	Pattern0	Pattern1	Pattern2	Pattern3	Pattern4	Pattern5	Pattern6	Pattern7	Portmask	CareTags	Template	Actions
0	0	0	0	0	0	0	0	0	0	0	0	I F L P S C
1	0	0	0	0	0x4455	0x2233	0x1100	0	0x3	0x8	0	0 0 0 0 0 1
2	0	0x0102	0xc0a8	0	0	0	0	0	0x3	0x8	1	0 0 0 0 0 0
⋮												
63	0	0	0	0	0	0	0	0	0	0	0	I F L P S C

```
rg set acl-filter fwding_type_and_direction 0
rg set acl-filter pattern ingress_port_mask 0x3
rg set acl-filter pattern ingress_src_ipv4_addr_start 192.168.1.2 ingress_src_ipv4_addr_end 192.168.1.2
rg set acl-filter pattern ingress_smac 00:11:22:33:44:55
rg set acl-filter action action_type 3
rg set acl-filter action qos action_dot1p_remarking_pri 2
rg add acl-filter entry
```

```
# cat proc/dump/acl
-------------- ACL TABLES ----------------
  --- ACL TABLE[1] ---
        valid:1
        field[4]:0x4455  mask[4]:0xffff {0x04:SMAC0[15:0]}
        field[5]:0x2233  mask[5]:0xffff {0x05:SMAC1[31:16]}
        field[6]:0x0011  mask[6]:0xffff {0x06:SMAC2[47:32]}
        active portmask:0x3
        tag_care:[IPv4]
        templateIdx:0
        action bits:[CVLAN]
        [CVLAN_ACTIDX:5(1P remark)] cvid:0 dot1p:2
  --- ACL TABLE[2] ---
        valid:1
        field[1]:0x0102  mask[1]:0xffff {0x10:IP4SIP[15:0]}
        field[2]:0xc0a8  mask[2]:0xffff {0x11:IP4SIP[31:16]}
        active portmask:0x3
        tag_care:[IPv4]
        templateIdx:1
        action bits:
```

Qos Configuration

- SPQ & WTQ setting
  - SPQ > WTQ
  - Per port have 8 Egress Queue
  - Internal-Priority 1-to-1 mapping to Queue (7 is reserved for dual wifi)
  - rtk_rg_qosInternalPriDecisionByWeight_set to give weight
- 802.1q Priority & DSCP Remarking
  - Internal-Priority Decision
  - RG API for Remarking

rtk_rg_qosStrictPriorityOrWeightFairQueue_set

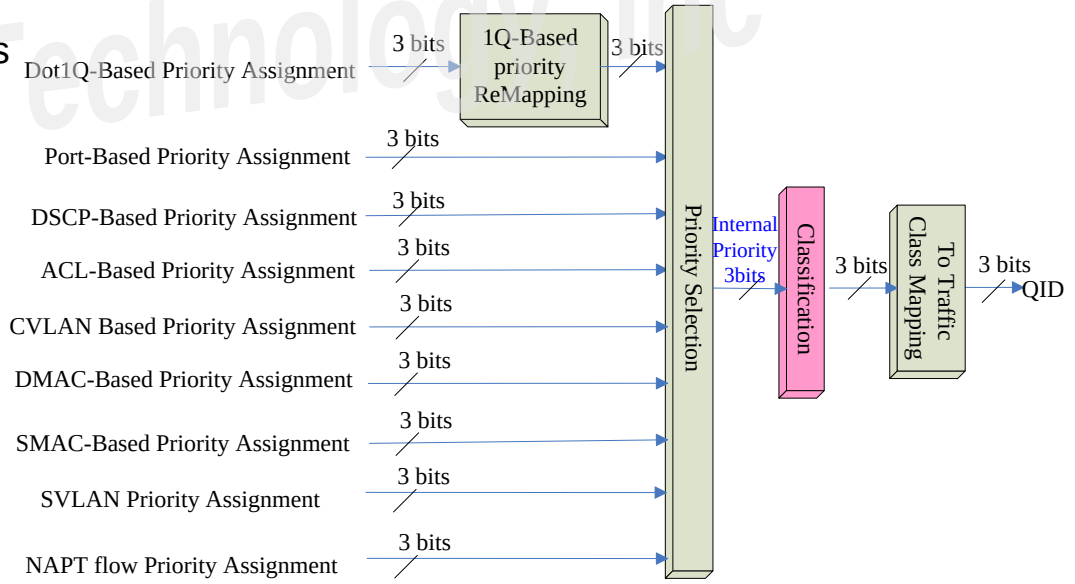
```
rtk_rg_aclFilterAndQos_t aclRule;
int aclIdx,i;

/*Set SPQ & WFQ for each port*/
rtk_rg_qos_queue_weights_t q_weight;
q_weight.weights[7]=0; //Queue4~7:Strict Priority
q_weight.weights[6]=0;
q_weight.weights[5]=0;
q_weight.weights[4]=0;
q_weight.weights[3]=4; //Queue3~0 4:3:2:1
q_weight.weights[2]=3;
q_weight.weights[1]=2;
q_weight.weights[0]=1;
for(i=0;i<7;i++) //per port egress queue weight setting
{
  rtk_rg_qosStrictPriorityOrWeightFairQueue_set (i,&q_weight);
}

/*Set ACL for all TCP packets to Queue[4]*/
memset(&aclRule,0,sizeof(rtk_rg_aclFilterAndQos_t));
aclRule.fwding_type_and_direction=ACL_FWD_TYPE_DIR_INGRESS_ALL_PACKET;
aclRule.filter_fields=INGRESS_L4_TCP_BIT;
aclRule.action_type=ACL_ACTION_TYPE_QOS;
aclRule.qos_actions=ACL_ACTION_QUEUE_ID_BIT;
aclRule.action_queue_id=0x4;
if(rtk_rg_aclFilterAndQos_add(&aclRule,&aclIdx)) return -1;
```

rtk_rg_qosInternalPriDecisionByWeight_set

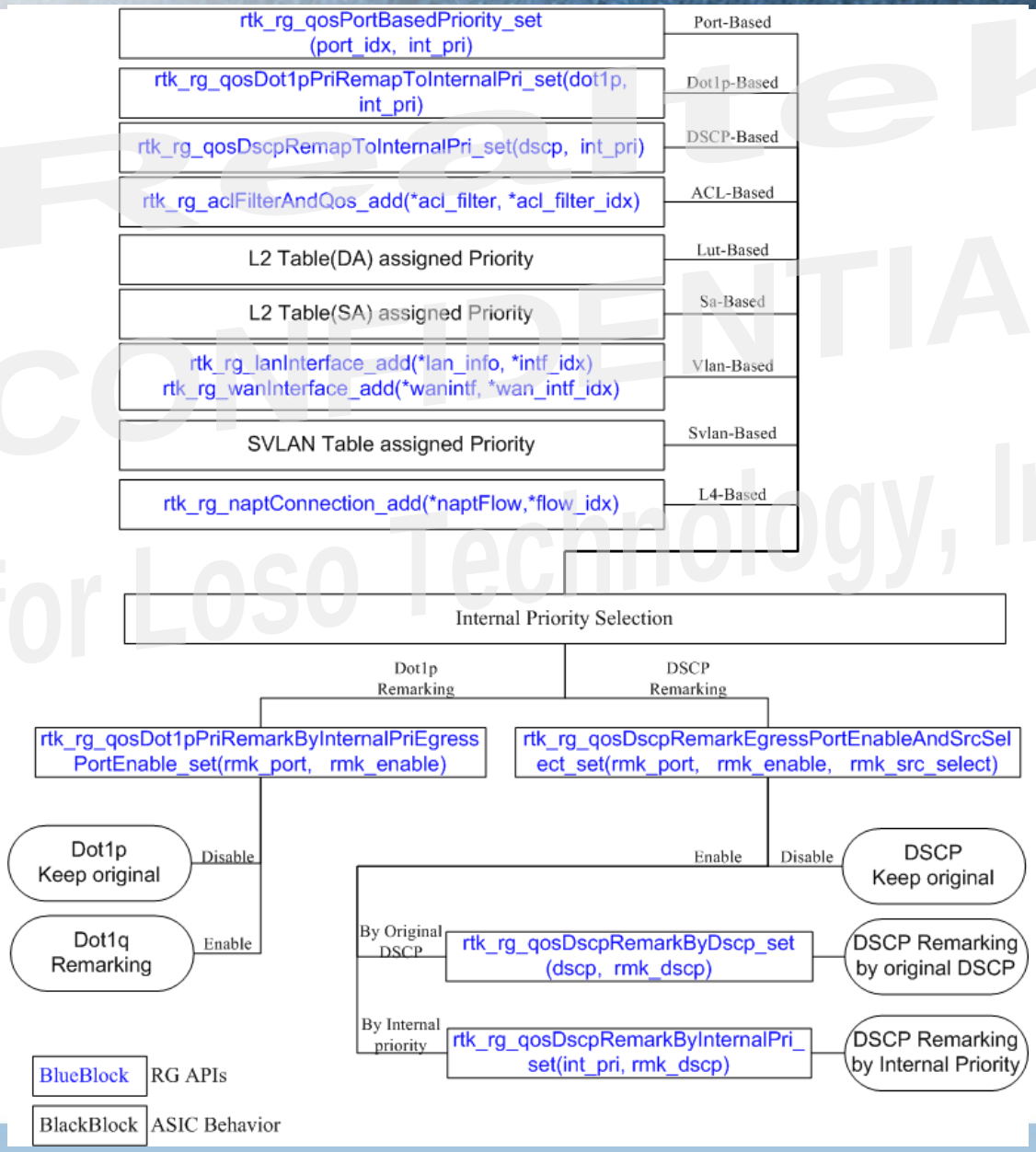
- rtk_rg_qosInternalPriDecisionByWeight_set(rtk_rg_qos_priSelWeight_t weightOfPriSel)

RTK.0> rg add qosInternalPriDecisionByWeight weight_of_portBased 1 weight_of_dot1q 2 weight_of_dscp 0 **weight_of_acl 15 weight_of_lutFwd 13** weight_of_saBaed 0 weight_of_vlanBased 10 weight_of_svlanBased 9 weight_of_l4Based 11

```
typedef struct rtk_rg_qos_priSelWeight_s
{
    uint32 weight_of_portBased;
    uint32 weight_of_dot1q;
    uint32 weight_of_dscp;
    uint32 weight_of_acl;
    uint32 weight_of_lutFwd;
    uint32 weight_of_saBaed;
    uint32 weight_of_vlanBased;
    uint32 weight_of_svlanBased;
    uint32 weight_of_l4Based;
}rtk_rg_qos_priSelWeight_t;
```

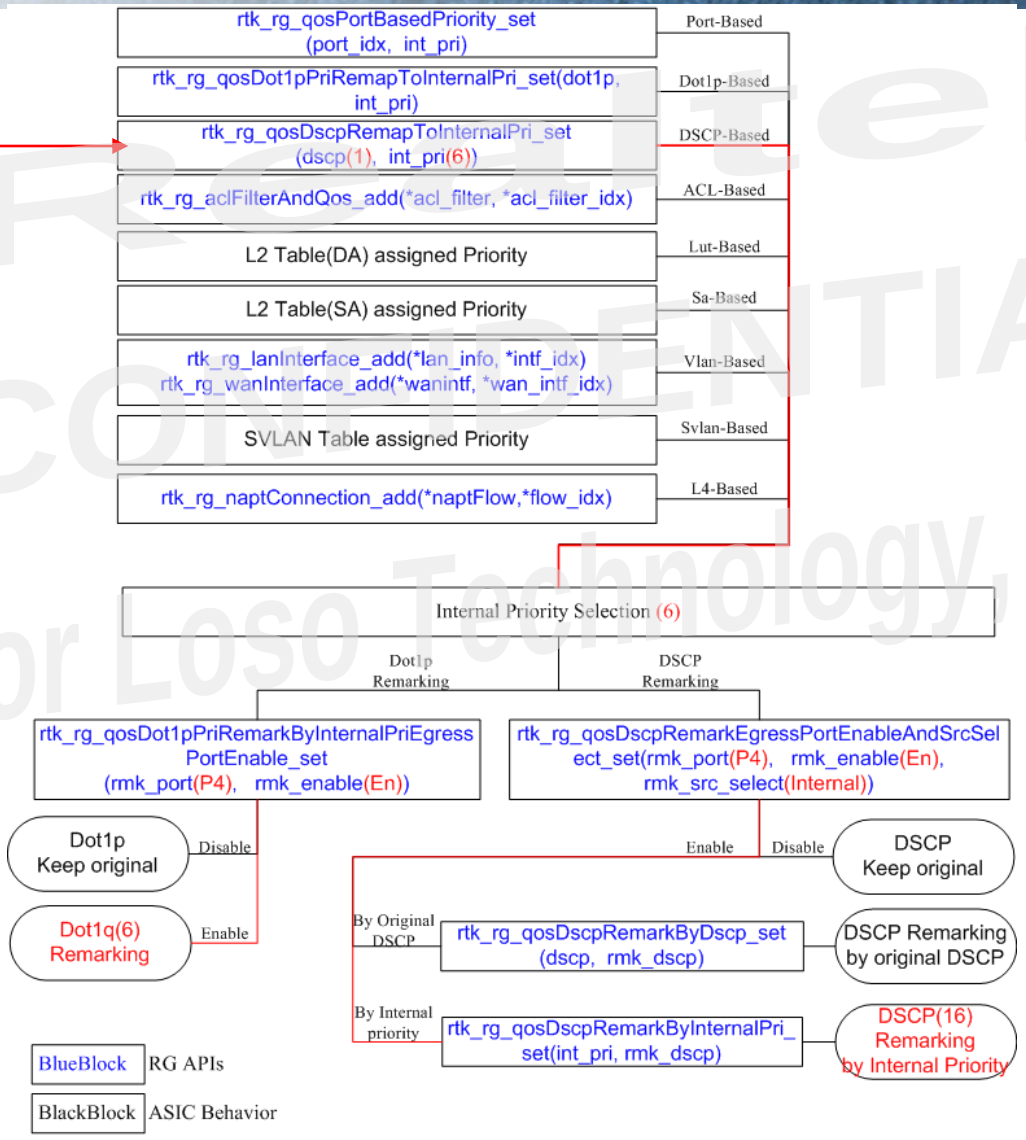


Qos Decision Flow



Qos Decision Flow

Ingress packet with DSCP=1




```
RTK.0> rg add qosDscpRemapToInternalPri dscp 1 intPri 6
RTK.0> rg add qosDot1pPriRemarkByInternalPriEgressPortEnable port 4 enabled 1
RTK.0> rg add qosDscpRemarkEgressPortEnableAndSrcSelect port 4 enabled 1 source_select 1
RTK.0> rg add qosDscpRemarkByInternalPri intPri 6 remark_dscp 16
```

Thank you.