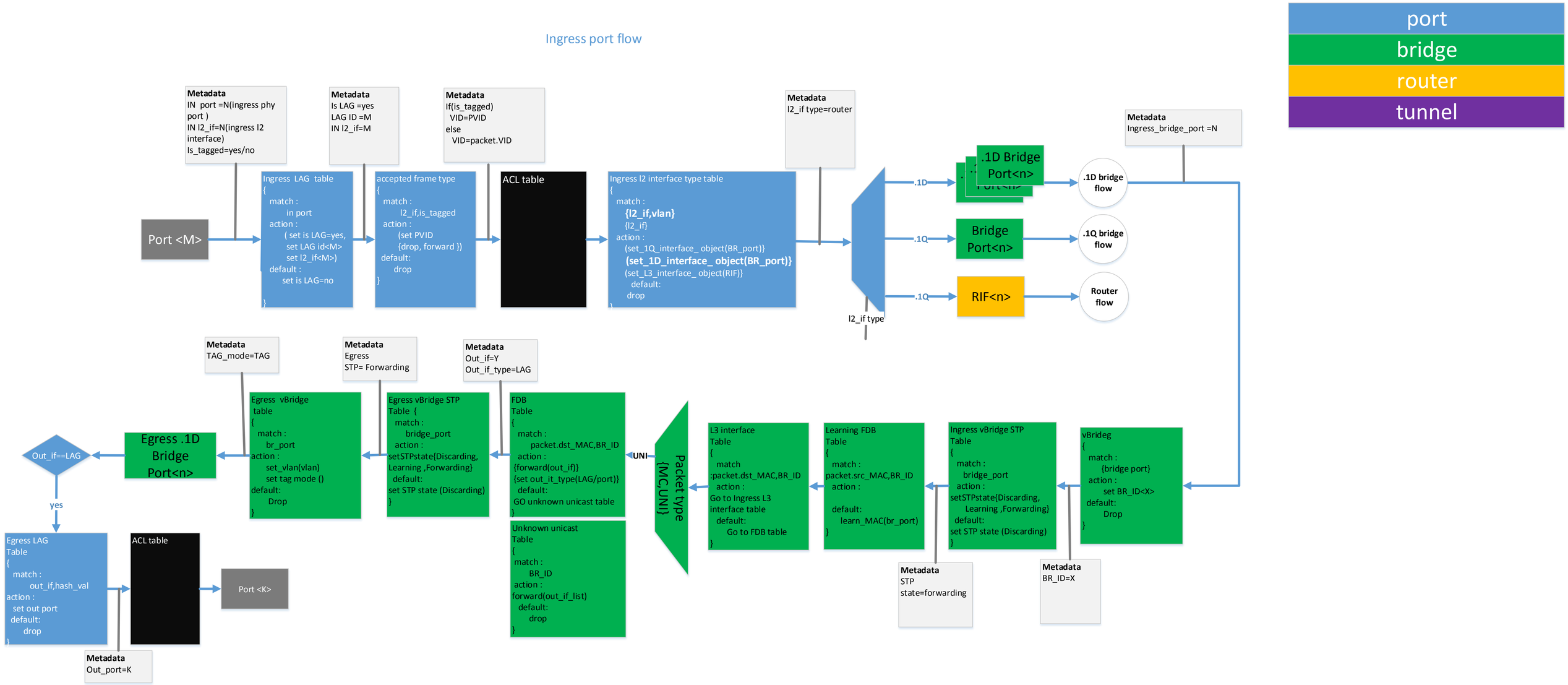
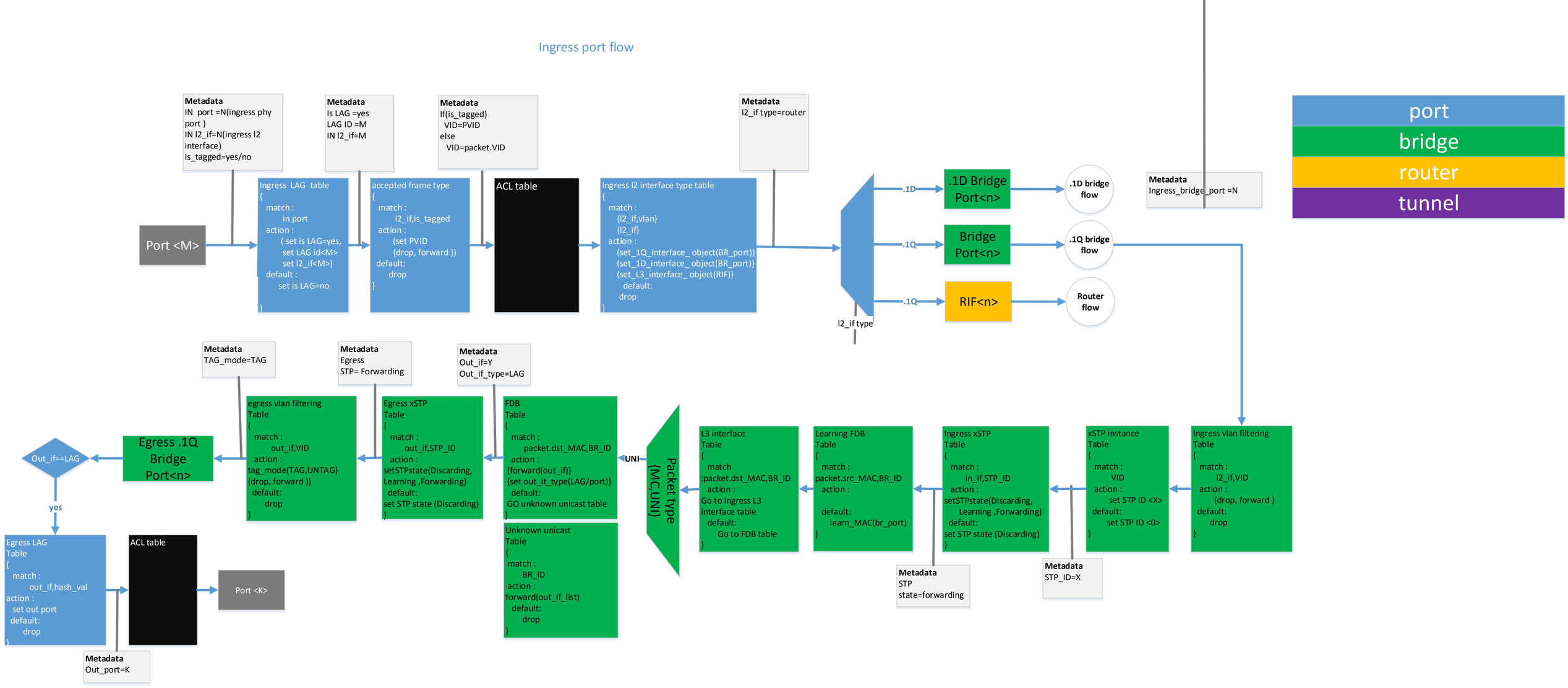
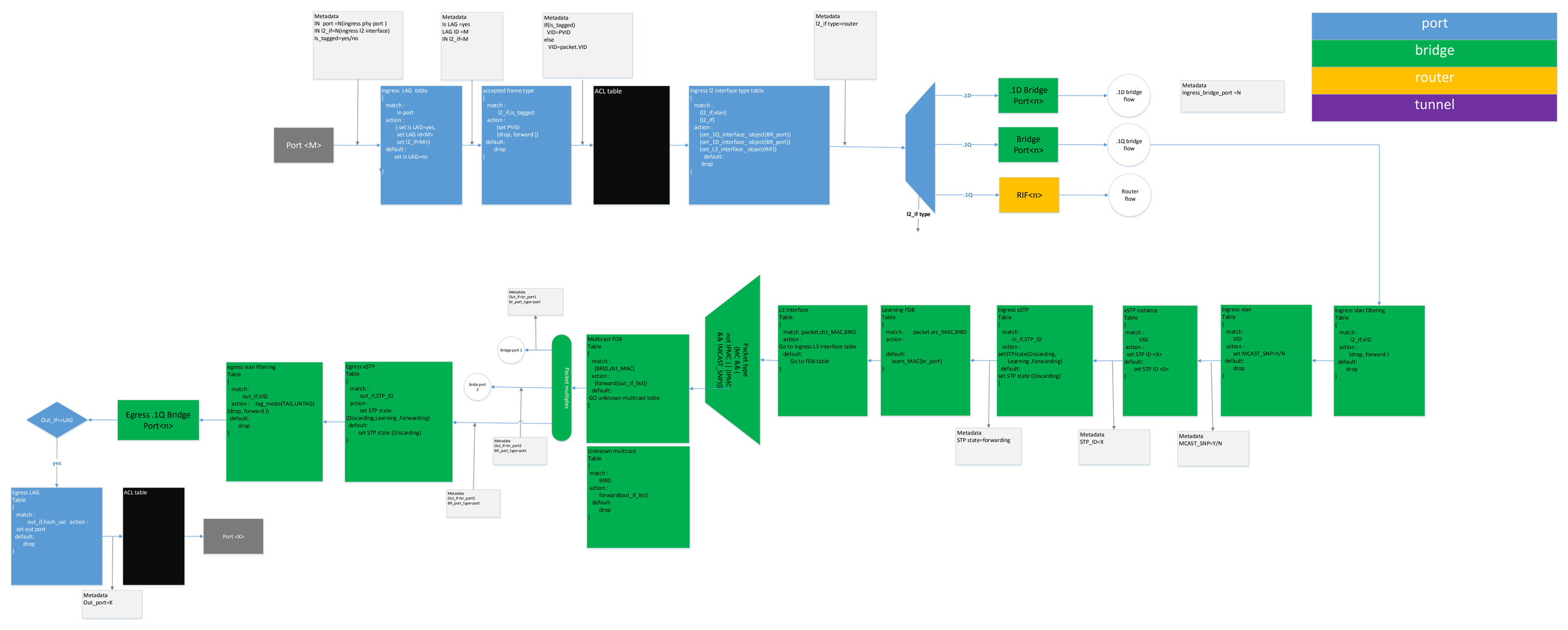
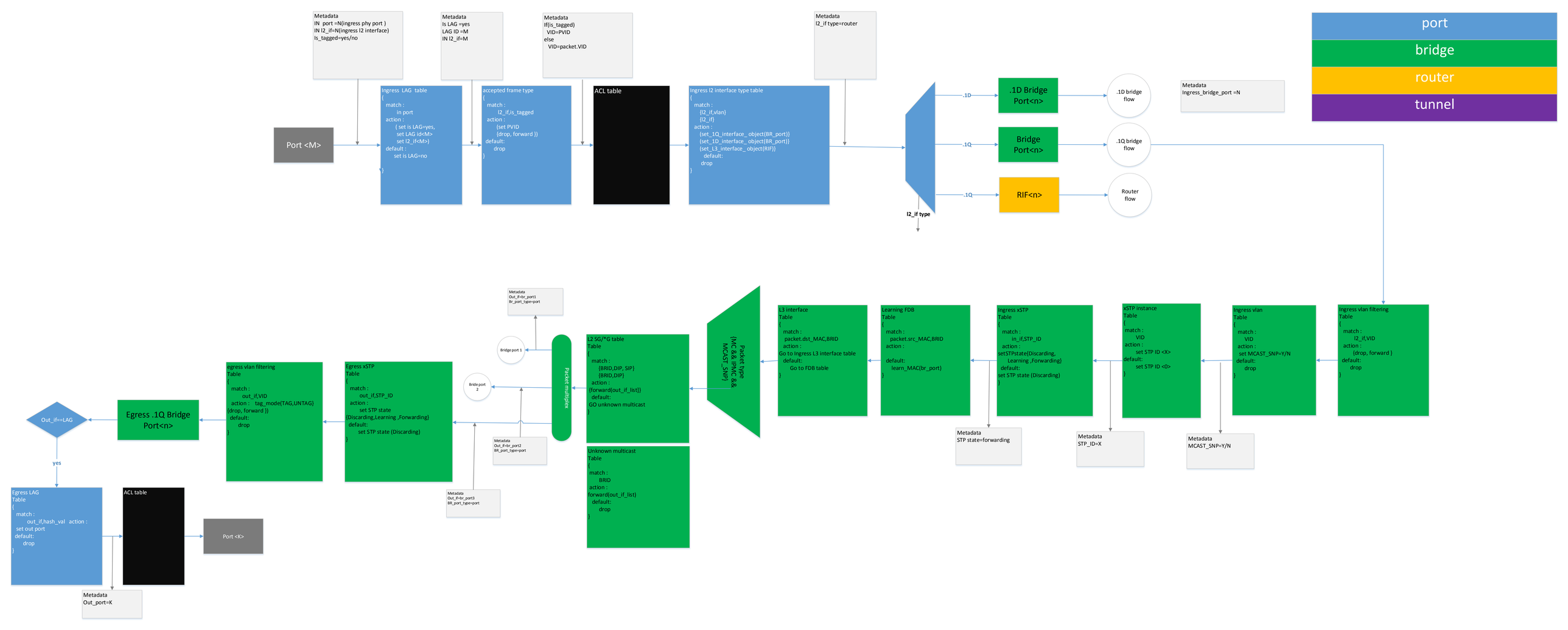
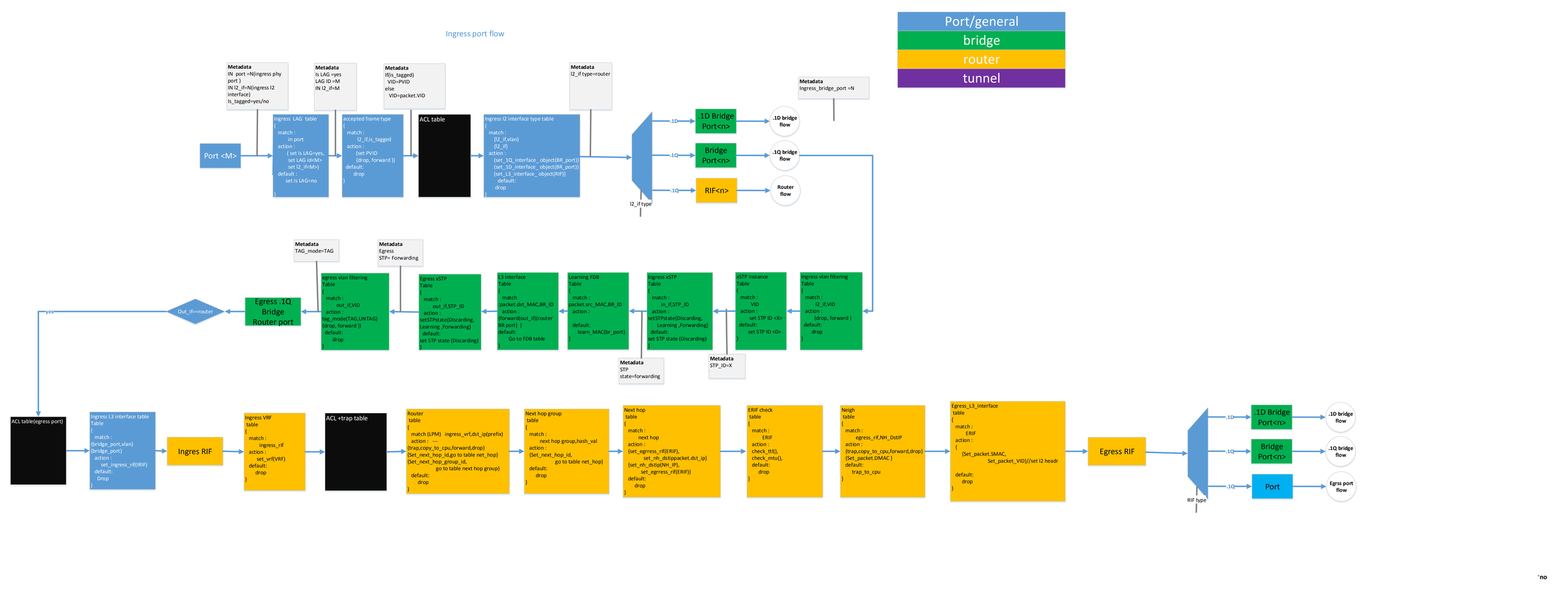

port
bridge
router
tunnel
Ingress port flow
Metadata
IN port =N(ingress phy port)
IN l2_if=N(ingress l2 interface)
Is_tagged=yes/no
Metadata
Is LAG =yes
LAG ID =M
IN l2_if=M
Metadata
If(is_tagged)
VID=PVID
else
VID=packet.VID
Metadata
l2_if type=router
Metadata
Ingress_bridge_port =N
Port <M>
Ingress LAG table
{
match :
in port
action :
(set is LAG=yes,
set LAG id<M>
set l2_if<M>)
default :
set is LAG=no
}
accepted frame type
{
match :
l2_if,is_tagged
action :
(set PVID
{drop, forward })
default:
drop
}
ACL table
Ingress l2 interface type table
{
match :
{l2_if,vlan}
{l2_if}
action :
(set_1Q_interface_ object(BR_port)}
(set_1D_interface_ object(BR_port)}
(set_L3_interface_ object(RIF)}
default:
drop
}
.1D
.1Q
.1Q
.1D Bridge Port<n>
Bridge Port<n>
RIF<n>
.1D bridge flow
.1Q bridge flow
Router flow
l2_if type
Metadata
TAG_mode=TAG
Metadata
Egress
STP= Forwarding
Metadata
Out_if=Y
Out_if_type=LAG
Egress vBridge table
{
match :
br_port
action :
set_vlan(vlan)
set tag mode ()
default:
Drop
}
Egress vBridge STP Table {
match :
bridge_port
action :
setSTPstate{Discarding, Learning ,Forwarding}
default:
set STP state (Discarding)
}
FDB Table
{
match :
packet.dst_MAC,BR_ID
action :
{forward(out_if)}
{set out_it_type(LAG/port)}
default:
GO unknown unicast table
}
Unknown unicast Table
{
match :
BR_ID
action :
forward(out_if_list)
default:
drop
}
UNI
Packet type {MC,UNI}
L3 interface Table
{
match :packet.dst_MAC,BR_ID
action :
Go to Ingress L3 interface table
default:
Go to FDB table
}
Learning FDB Table
{
match :
packet.src_MAC,BR_ID
action :
default:
learn_MAC(br_port)
}
Ingress vBridge STP Table
{
match :
bridge_port
action :
setSTPstate{Discarding, Learning ,Forwarding}
default:
set STP state (Discarding)
}
vBrideg
{
match :
{bridge port}
action :
set BR_ID<X>
default:
Drop
}
Metadata
STP
state=forwarding
Metadata
BR_ID=X
Out_if==LAG
Egress .1D Bridge Port<n>
yes
Egress LAG Table
{
match :
out_if,hash_val
action :
set out port
default:
drop
}
ACL table
Port <K>
Metadata
Out_port=K

Ingress port flow

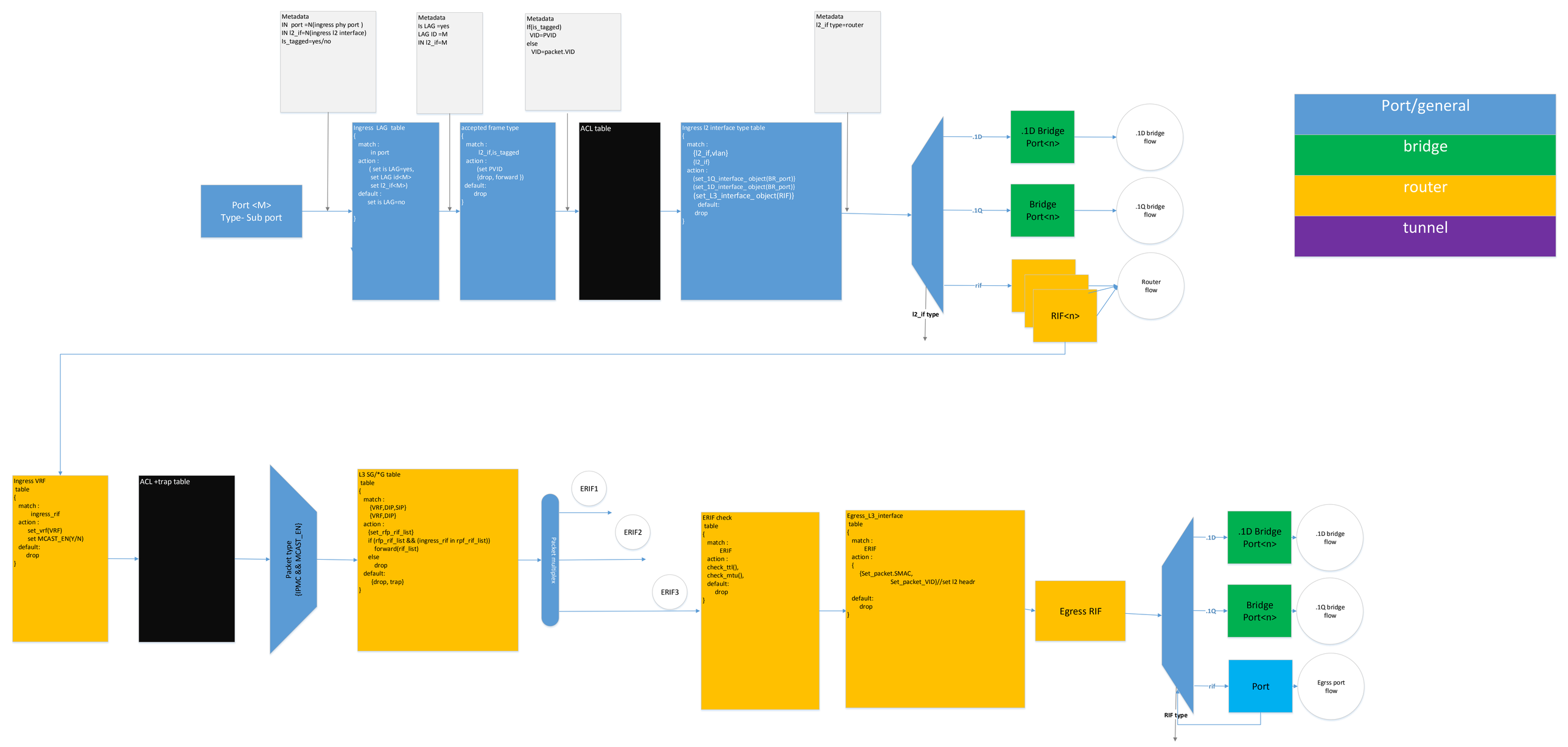


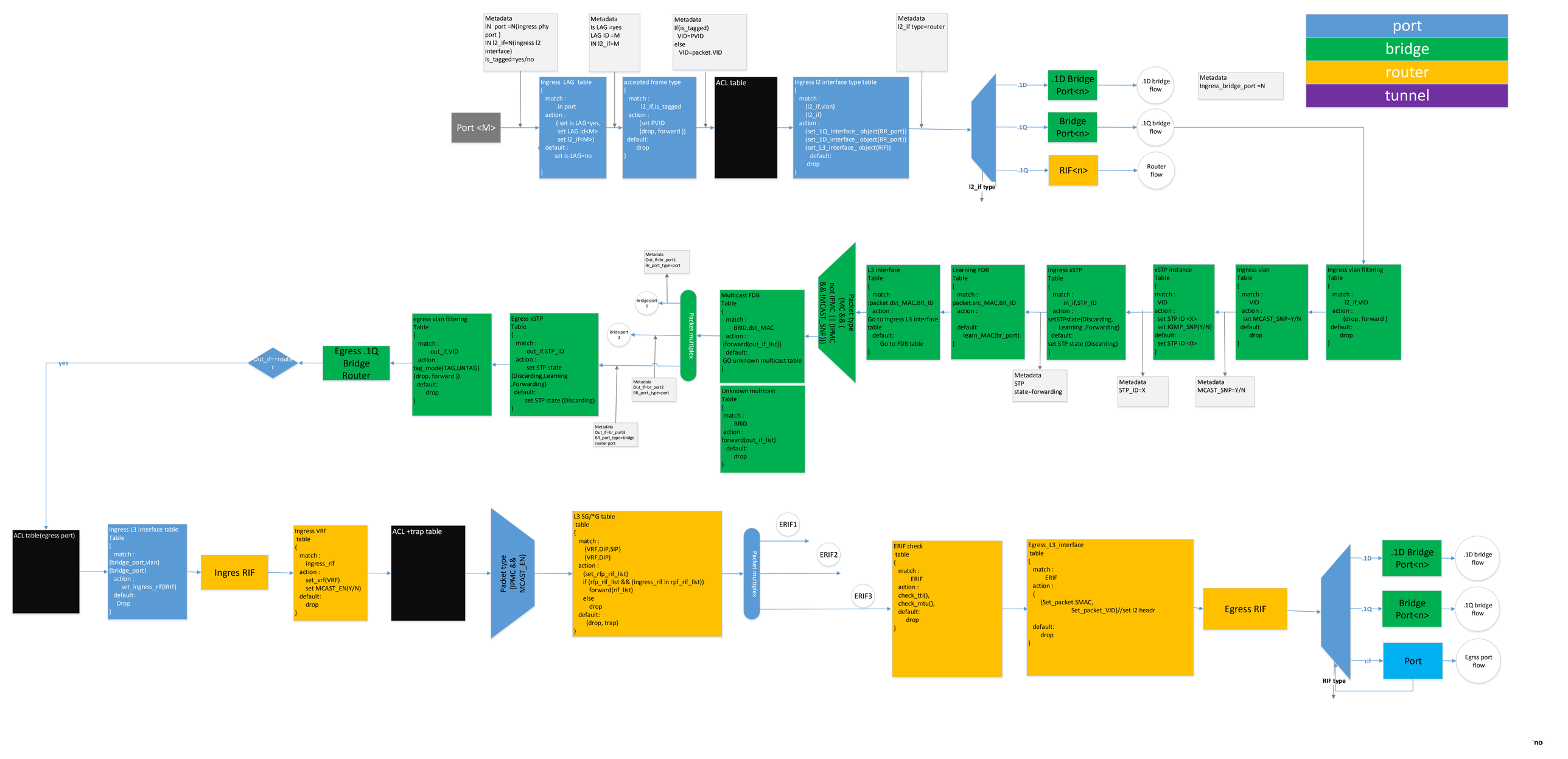

port
bridge
router
tunnel
Metadata
IN port =N(ingress phy port)
IN I2_if=N(ingress I2 interface)
Is_tagged=yes/no
Metadata
Is LAG =yes
LAG ID =M
IN I2_if=M
Metadata
If(is_tagged)
VID=PVID
else
VID=packet.VID
Metadata
I2_if type=router
Port <M>
Ingress LAG table
{
match :
in port
action :
{ set is LAG=yes,
set LAG id<M>
set I2_if<M>)
default :
set is LAG=no
}
accepted frame type
{
match :
I2_if,is_tagged
action :
{set PVID
{drop, forward }}
default:
drop
}
ACL table
Ingress I2 interface type table
{
match :
{I2_if,vlan}
{I2_if}
action :
{set_1Q_interface_ object(BR_port)}
{set_1D_interface_ object(BR_port)}
{set_L3_interface_ object(RIF)}
default:
drop
}
I2_if type
.1D
.1Q
.1Q
.1D Bridge Port<n>
Bridge Port<n>
RIF<n>
.1D bridge flow
.1Q bridge flow
Router flow
Metadata
Ingress_bridge_port =N
Ingress vlan filtering
Table
{
match :
I2_if,VID
action :
{drop, forward }
default:
drop
}
Ingress vlan
Table
{
match :
VID
action :
set MCAST_SNP=Y/N
default:
drop
}
Metadata
MCAST_SNP=Y/N
xSTP instance
Table
{
match :
VID
action :
set STP ID <X>
default:
set STP ID <0>
}
Metadata
STP_ID=X
Ingress xSTP
Table
{
match :
in_if,STP_ID
action :
setSTPstate(Discarding,
Learning ,Forwarding)
default:
set STP state (Discarding)
}
Metadata
STP state=forwarding
Learning FDB
Table
{
match : packet.src_MAC,BRID
action :
default:
learn_MAC(br_port)
}
L3 interface
Table
{
match :packet.dst_MAC,BRID
action :
Go to Ingress L3 interface table
default:
Go to FDB table
}
Packet type
{MC && (
not IPMC || (IPMC
&& !MCAST_SNP))}
Multicast FDB
Table
{
match :
{BRID,dst_MAC}
action :
{forward(out_if_list)}
default:
GO unknown multicast table
}
Unknown multicast
Table
{
match :
BRID
action :
forward(out_if_list)
default:
drop
}
Packet multiplex
Bridge port 1
Bride port 2
Metadata
Out_if=br_port1
Br_port_type=port
Metadata
Out_if=br_port2
BR_port_type=port
Metadata
Out_if=br_port3
BR_port_type=port
Egress xSTP
Table
{
match :
out_if,STP_ID
action :
set STP state
{Discarding,Learning ,Forwarding}
default:
set STP state (Discarding)
}
egress vlan filtering
Table
{
match :
out_if,VID
action : tag_mode{TAG,UNTAG}
{drop, forward }}
default:
drop
}
Egress .1Q Bridge Port<n>
Out_if==LAG
yes
Egress LAG
Table
{
match :
out_if,hash_val action :
set out port
default:
drop
}
ACL table
Port <K>
Metadata
Out_port=K


port
bridge
router
tunnel
Metadata
IN_port =N(ingress phy port)
IN I2_if=N(ingress I2 interface)
Is_tagged=yes/no
Metadata
Is LAG =yes
LAG ID =M
IN I2_if=M
Metadata
If(is_tagged)
VID=PVID
else
VID=packet.VID
Metadata
I2_if type=router
Port <M>
Ingress_LAG_table
{
match :
in port
action :
{ set is LAG=yes,
set LAG id<M>
set I2_if<M>)
default :
set is LAG=no
}
accepted frame type
{
match :
I2_if,is_tagged
action :
{set PVID
{drop, forward }}
default:
drop
}
ACL table
Ingress I2 interface type table
{
match :
{I2_if,vlan}
{I2_if}
action :
{set_1Q_interface_ object(BR_port)}
{set_1D_interface_ object(BR_port)}
{set_L3_interface_ object(RIF)}
default:
drop
}
I2_if type
.1D
.1Q
.1Q
.1D Bridge Port<n>
Bridge Port<n>
RIF<n>
.1D bridge flow
.1Q bridge flow
Router flow
Metadata
Ingress_bridge_port =N
Ingress vlan filtering
Table
{
match :
I2_if,VID
action :
{drop, forward }
default:
drop
}
Ingress vlan
Table
{
match :
VID
action :
set MCAST_SNP=Y/N
default:
drop
}
xSTP instance
Table
{
match :
VID
action :
set STP ID <X>
default:
set STP ID <0>
}
Ingress xSTP
Table
{
match :
in_if,STP_ID
action :
setSTPstate{Discarding,
Learning ,Forwarding}
default:
set STP state (Discarding)
}
Learning FDB
Table
{
match :
packet.src_MAC,BRID
action :
default:
learn_MAC(br_port)
}
L3 interface
Table
{
match :
packet.dst_MAC,BRID
action :
Go to Ingress L3 interface table
default:
Go to FDB table
}
Packet type
{MC && IPMC && MCAST_SNP}
L2 SG/*G table
Table
{
match :
{BRID,DIP, SIP}
{BRID,DIP}
action :
{forward(out_if_list)}
default:
GO unknown multicast
}
Unknown multicast
Table
{
match :
BRID
action :
forward(out_if_list}
default:
drop
}
Packet multiplex
Bridge port 1
Bride port 2
Metadata
Out_if=br_port1
Br_port_type=port
Metadata
Out_if=br_port2
BR_port_type=port
Metadata
Out_if=br_port3
BR_port_type=port
Metadata
STP state=forwarding
Metadata
STP_ID=X
Metadata
MCAST_SNP=Y/N
egress vlan filtering
Table
{
match :
out_if,VID
action : tag_mode{TAG,UNTAG}
{drop, forward }}
default:
drop
}
Egress xSTP
Table
{
match :
out_if,STP_ID
action :
set STP state
{Discarding,Learning ,Forwarding}
default:
set STP state (Discarding)
}
Egress .1Q Bridge Port<n>
Out_if==LAG
yes
Egress LAG
Table
{
match :
out_if,hash_val action :
set out port
default:
drop
}
ACL table
Port <K>
Metadata
Out_port=K

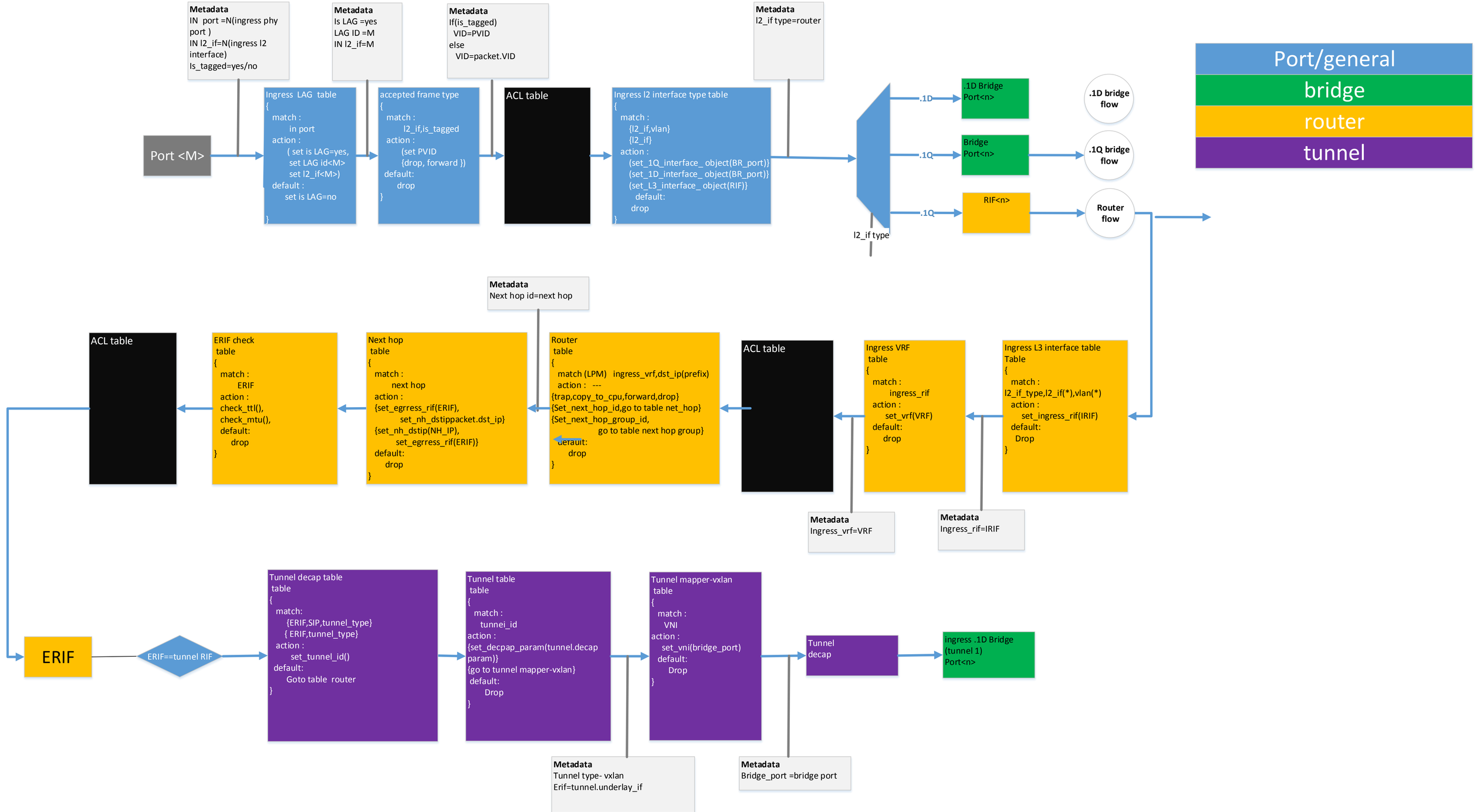

Port/general
bridge
router
tunnel
Ingress port flow
Metadata
IN_port =N(ingress phy port)
IN I2_if=N(ingress l2 interface)
Is_tagged=yes/no
Metadata
Is LAG =yes
LAG ID =M
IN I2_if=M
Metadata
If(is_tagged)
VID=PVID
else
VID=packet.VID
Metadata
I2_if type=router
Metadata
Ingress_bridge_port =N
Port <M>
Ingress_LAG_table
{
match :
in port
action :
{ set is LAG=yes,
set LAG id<M>
set I2_if<M>)
default :
set is LAG=no
}
accepted frame type
{
match :
I2_if,is_tagged
action :
{set PVID
(drop, forward)}
default:
drop
}
ACL table
Ingress I2 interface type table
{
match :
{I2_if,vlan}
{I2_if}
action :
{set_1Q_interface_object(BR_port)}
{set_1D_interface_object(BR_port)}
{set_L3_interface_object(RIF)}
default:
drop
}
.1D
.1Q
.1Q
.1D Bridge Port<n>
Bridge Port<n>
RIF<n>
.1D bridge flow
.1Q bridge flow
Router flow
I2_if type
Metadata
TAG_mode=TAG
Metadata
Egress
STP= Forwarding
egress vlan filtering
Table
{
match :
out_if,VID
action :
tag_mode(TAG,UNTAG)
{drop, forward }}
default:
drop
}
Egress xSTP
Table
{
match :
out_if,STP_ID
action :
setSTPstate(Discarding,
Learning ,Forwarding)
default:
set STP state (Discarding)
}
L3 interface
Table
{
match
packet.dst_MAC,BR_ID
action :
{forward(out_if)(router
BR port) }
default:
Go to FDB table
}
Learning FDB
Table
{
match :
packet.src_MAC,BR_ID
action :
default:
learn_MAC(br_port)
}
Ingress xSTP
Table
{
match :
in_if,STP_ID
action :
setSTPstate(Discarding,
Learning ,Forwarding)
default:
set STP state (Discarding)
}
xSTP instance
Table
{
match :
VID
action :
set STP ID <X>
default:
set STP ID <0>
}
Ingress vlan filtering
Table
{
match :
I2_if,VID
action :
{drop, forward }
default:
drop
}
Metadata
STP
state=forwarding
Metadata
STP_ID=X
yes
Out_if==router
Egress .1Q Bridge Router port
ACL table(egress port)
Ingress L3 interface table
Table
{
match :
{bridge_port,vlan}
{bridge_port}
action :
set_ingress_rif(IRIF)
default:
Drop
}
Ingres RIF
Ingress VRF
table
{
match :
ingress_rif
action :
set_vrf(VRF)
default:
drop
}
ACL +trap table
Router
table
{
match (LPM) ingress_vrf,dst_ip(prefix)
action : ---
{trap,copy_to_cpu,forward,drop}
{Set_next_hop_id,go to table net_hop}
{Set_next_hop_group_id,
go to table next hop group}
default:
drop
}
Next hop group
table
{
match :
next hop group,hash_val
action :
{Set_next_hop_id,
go to table net_hop}
default:
drop
}
Next hop
table
{
match :
next hop
action :
{set_egress_rif(ERIF),
set_nh_dstippacket.dst_ip}
{set_nh_dstip(NH_IP),
set_egress_rif(ERIF)}
default:
drop
}
ERIF check
table
{
match :
ERIF
action :
check_ttl(),
check_mtu(),
default:
drop
}
Neigh
table
{
match :
egress_rif,NH_DstIP
action :
{trap,copy_to_cpu,forward,drop}
{Set_packet.DMAC }
default:
trap_to_cpu
}
Egress_L3_interface
table
{
match :
ERIF
action :
{
{Set_packet.SMAC,
Set_packet_VID}//set l2 headr
default:
drop
}
Egress RIF
.1D
.1Q
.1Q
.1D Bridge Port<n>
Bridge Port<n>
Port
RIF type
.1D bridge flow
.1Q bridge flow
Egrss port flow

*no


Metadata
IN_port =N(ingress phy port)
IN l2_if=N(ingress l2 interface)
Is_tagged=yes/no
Metadata
Is LAG =yes
LAG ID =M
IN l2_if=M
Metadata
If(is_tagged)
VID=PVID
else
VID=packet.VID
Metadata
l2_if type=router
Port <M>
Type- Sub port
Ingress_LAG_table
{
match :
in port
action :
(set is LAG=yes,
set LAG id<M>
set l2_if<M>)
default :
set is LAG=no
}
accepted frame type
{
match :
l2_if,is_tagged
action :
{set PVID
{drop, forward }}
default:
drop
}
ACL table
Ingress l2 interface type table
{
match :
{l2_if,vlan}
{l2_if}
action :
{set_1Q_interface_ object(BR_port)}
{set_1D_interface_ object(BR_port)}
{set_L3_interface_ object(RIF)}
default:
drop
}
.1D
.1Q
rif
l2_if type
.1D Bridge Port<n>
Bridge Port<n>
RIF<n>
.1D bridge flow
.1Q bridge flow
Router flow
Port/general
bridge
router
tunnel
Ingress VRF
table
{
match :
ingress_rif
action :
set_vrf(VRF)
set MCAST_EN(Y/N)
default:
drop
}
ACL +trap table
Packet type
{IPMC && MCAST_EN}
L3 SG/*G table
table
{
match :
{VRF,DIP,SIP}
{VRF,DIP}
action :
{set_rfp_rif_list}
if (rfp_rif_list && (ingress_rif in rpf_rif_list))
forward(rif_list)
else
drop
default:
{drop, trap}
}
Packet multiplex
ERIF1
ERIF2
ERIF3
ERIF check
table
{
match :
ERIF
action :
check_ttl(),
check_mtu(),
default:
drop
}
Egress_L3_interface
table
{
match :
ERIF
action :
{
{Set_packet.SMAC,
Set_packet_VID}//set l2 headr
default:
drop
}
Egress RIF
.1D
.1Q
rif
RIF type
.1D Bridge Port<n>
Bridge Port<n>
Port
.1D bridge flow
.1Q bridge flow
Egrss port flow


Metadata
IN port =N(ingress phy port)
IN I2_if=N(ingress I2 interface)
Is_tagged=yes/no
Metadata
Is LAG =yes
LAG ID =M
IN I2_if=M
Metadata
If(is_tagged)
VID=PVID
else
VID=packet.VID
Metadata
I2_if type=router
port
bridge
router
tunnel
Port <M>
Ingress LAG table
{
match :
in port
action :
(set is LAG=yes,
set LAG id<M>
set I2_if<M>)
default :
set is LAG=no
}
accepted frame type
{
match :
I2_if,is_tagged
action :
(set PVID
{drop, forward })
default:
drop
}
ACL table
Ingress I2 interface type table
{
match :
{I2_if,vlan}
{I2_if}
action :
(set_1Q_interface_ object{BR_port})
(set_1D_interface_ object{BR_port})
(set_L3_interface_ object(RIF))
default:
drop
}
I2_if type
.1D
.1Q
.1Q
.1D Bridge Port<n>
Bridge Port<n>
RIF<n>
.1D bridge flow
.1Q bridge flow
Router flow
Metadata
Ingress_bridge_port =N
Ingress vlan filtering Table
{
match :
I2_if,VID
action :
{drop, forward }
default:
drop
}
Ingress vlan Table
{
match :
VID
action :
set MCAST_SNP=Y/N
default:
drop
}
xSTP instance Table
{
match :
VID
action :
set STP ID <X>
set IGMP_SNP(Y/N)
default:
set STP ID <0>
}
Ingress xSTP Table
{
match :
in_if,STP_ID
action :
setSTPstate{Discarding, Learning ,Forwarding}
default:
set STP state (Discarding)
}
Learning FDB Table
{
match :
packet.src_MAC,BR_ID
action :
default:
learn_MAC(br_port)
}
L3 interface Table
{
match
:packet.dst_MAC,BR_ID
action :
Go to Ingress L3 interface table
default:
Go to FDB table
}
Packet type
{MC && (
not IPMC || IPMC
&& IMCAST_SNF)}
Multicast FDB Table
{
match :
BRID,dst_MAC
action :
{forward(out_if_list)}
default:
GO unknown multicast table
}
Unknown multicast Table
{
match :
BRID
action :
forward(out_if_list)
default:
drop
}
Packet multiplex
Metadata
Out_if=br_port1
Br_port_type=port
Bridge port 1
Bride port 2
Metadata
Out_if=br_port2
BR_port_type=port
Metadata
Out_if=br_port3
BR_port_type=bridge router port
Egress xSTP Table
{
match :
out_if,STP_ID
action :
set STP state {Discarding,Learning ,Forwarding}
default:
set STP state (Discarding)
}
egress vlan filtering Table
{
match :
out_if,VID
action :
tag_mode{TAG,UNTAG}
{drop, forward }}
default:
drop
}
Out_if==router
yes
Egress .1Q Bridge Router
Metadata
STP state=forwarding
Metadata
STP_ID=X
Metadata
MCAST_SNP=Y/N
ACL table(egress port)
Ingress L3 interface table
Table
{
match :
{bridge_port,vlan}
{bridge_port}
action :
set_ingress_rif(IRIF)
default:
Drop
}
Ingres RIF
Ingress VRF
table
{
match :
ingress_rif
action :
set_vrf(VRF)
set MCAST_EN(Y/N)
default:
drop
}
ACL +trap table
Packet type
{IPMC &&
MCAST_EN}
L3 SG/*G table
table
{
match :
{VRF,DIP,SIP}
{VRF,DIP}
action :
{set_rfp_rif_list}
if (rfp_rif_list && (ingress_rif in rpf_rif_list))
forward(rif_list)
else
drop
default:
{drop, trap}
}
Packet multiplex
ERIF1
ERIF2
ERIF3
ERIF check
table
{
match :
ERIF
action :
check_ttl(),
check_mtu(),
default:
drop
}
Egress_L3_interface
table
{
match :
ERIF
action :
{
{Set_packet.SMAC,
Set_packet_VID}//set I2 headr
default:
drop
}
Egress RIF
.1D
.1Q
rif
.1D Bridge Port<n>
Bridge Port<n>
Port
.1D bridge flow
.1Q bridge flow
Egrss port flow
RIF type

`no


Port/general
bridge
router
tunnel
Metadata
IN port =N(ingress phy port)
IN l2_if=N(ingress l2 interface)
Is_tagged=yes/no
Metadata
Is LAG =yes
LAG ID =M
IN l2_if=M
Metadata
If(is_tagged)
VID=PVID
else
VID=packet.VID
Metadata
l2_if type=router
Port <M>
Ingress LAG table
{
match :
in port
action :
(set is LAG=yes,
set LAG id<M>
set l2_if<M>)
default :
set is LAG=no
}
accepted frame type
{
match :
l2_if,is_tagged
action :
(set PVID
{drop, forward })
default:
drop
}
ACL table
Ingress l2 interface type table
{
match :
{l2_if,vlan}
{l2_if}
action :
(set_1Q_interface_ object(BR_port)}
(set_1D_interface_ object(BR_port)}
(set_L3_interface_ object(RIF)}
default:
drop
}
l2_if type
.1D
.1Q
.1Q
.1D Bridge Port<n>
Bridge Port<n>
RIF<n>
.1D bridge flow
.1Q bridge flow
Router flow
Metadata
Next hop id=next hop
ACL table
ERIF check
table
{
match :
ERIF
action :
check_ttl(),
check_mtu(),
default:
drop
}
Next hop
table
{
match :
next hop
action :
{set_egrress_rif(ERIF),
set_nh_dstippacket.dst_ip}
{set_nh_dstip(NH_IP),
set_egrress_rif(ERIF)}
default:
drop
}
Router
table
{
match (LPM) ingress_vrf,dst_ip(prefix)
action : ---
{trap,copy_to_cpu,forward,drop}
{Set_next_hop_id,go to table net_hop}
{Set_next_hop_group_id,
go to table next hop group}
default:
drop
}
ACL table
Ingress VRF
table
{
match :
ingress_rif
action :
set_vrf(VRF)
default:
drop
}
Ingress L3 interface table
Table
{
match :
l2_if_type,l2_if(*),vlan(*)
action :
set_ingress_rif(IRIF)
default:
Drop
}
Metadata
Ingress_vrf=VRF
Metadata
Ingress_rif=IRIF
ERIF
ERIF==tunnel RIF
Tunnel decap table
table
{
match:
{ERIF,SIP,tunnel_type}
{ ERIF,tunnel_type}
action :
set_tunnel_id()
default:
Goto table router
}
Tunnel table
table
{
match :
tunnei_id
action :
{set_decpap_param(tunnel.decap param)}
{go to tunnel mapper-vxlan}
default:
Drop
}
Tunnel mapper-vxlan
table
{
match :
VNI
action :
set_vni(bridge_port)
default:
Drop
}
Tunnel decap
ingress .1D Bridge (tunnel 1) Port<n>
Metadata
Tunnel type- vxlan
Erif=tunnel.underlay_if
Metadata
Bridge_port =bridge port

Port/general

bridge

router

tunnel

Ingress port flow

Port <M>

Metadata
IN_port =N(ingress phy port)
IN l2_if=N(ingress l2 interface)
Is_tagged=yes/no

Ingress_LAG_table
{
match :
in port
action :
(set is LAG=yes,
set LAG id<M>
set l2_if<M>)
default :
set is LAG=no
}

Metadata
Is LAG =yes
LAG ID =M
IN l2_if=M

accepted frame type
{
match :
l2_if,is_tagged
action :
{set PVID
{drop, forward }}
default:
drop
}

Metadata
If(is_tagged)
VID=PVID
else
VID=packet.VID

ACL table

Ingress l2 interface type table
{
match :
{l2_if,vlan}
{l2_if}
action :
{set_1Q_interface_ object(BR_port)}
{set_1D_interface_ object(BR_port)}
{set_L3_interface_ object(RIF)}
default:
drop
}

Metadata
l2_if type=router

l2_if type

.1D

.1Q

.1Q

.1D Bridge Port<n>

Bridge Port<n>

RIF<n>

.1D bridge flow

.1Q bridge flow

Router flow

Metadata
Ingress_bridge_port =N

vBrideg
{
match :
{bridge port}
action :
set BR_ID<X>
default:
Drop
}

Metadata
BR_ID=X

Ingress vBridge STP Table
{
match :
bridge_port
action :
setSTPstate{Discarding, Learning ,Forwarding}
default:
set STP state (Discarding)
}

Metadata
STP state=forwarding

Learning FDB Table
{
match :
packet.src_MAC,BR_ID
action :
default:
learn_MAC(br_port)
}

L3 interface Table
{
match
:packet.dst_MAC,BR_ID
action :
Go to Ingress L3 interface table
default:
Go to FDB table
}

Packet type {MC,UNI}

UNI

FDB Table
{
match :
packet.dst_MAC,BR_ID
action :
{forward(out_if)}
{forward(tunnel_id,dst_ip)}
{set out_it_type(LAG/port)}
default:
GO unknown unicast table
}

Unknown unicast Table
{
match :
BR_ID
action :
forward(out_if_list+(ou_if,dst_ip)*N)
default:
drop
}

Packet multiplex

Egress port flow

Egress LAG flow

Metadata
Out_if=Y1
Out_if_type=port

Metadata
Out_if=Y2
Out_if_type=LAG

Metadata
Out_if_type=tunnel

Egress vBridge STP Table {
match :
bridge_port
action :
setSTPstate{Discarding, Learning ,Forwarding}
default:
set STP state (Discarding)
}

Egress vBridge table
{
match :
br_port
action :
set_vlan(vlan)
set tag mode ()
default:
Drop
}

Metadata
Egress
STP= Forwarding

Egress .1D Bridge Port<n>

Out_if==tunnel port

yes

Metadata
TAG_mode=TAG

Metadata
Out_if=Y
Out_if_type=tunnel port
Tunnel type- vxlan
bridge_port=tunnel.underlay_if
Dst_ip=dst_id
Src_ip=tunnel.encap.src_ip
Router_rif=tunnel.encap.oerlay_if

ACL table(egress port)

Tunnel table
table
{
match :
tunnel_id
action :
{set_encpap_param(tunnel.encap param)}
{go to tunnel mapper-vxlan}
default:
Drop
}

Tunnel mapper-vxlan
table
{
match :BR_ID

action :
set_vni(VNI)
default:
Drop
}

Metadata
VNI=VNI

Tunnel (enacap)

RIF<n>

Ingress VRF
table
{
match :
ingress_rif
action :
set_vrf(VRF)
default:
drop
}

ACL table

Router
table
{
match (LPM) ingress_vrf,dst_ip(prefix)
action : ---
{trap,copy_to_cpu,forward,drop}
{Set_next_hop_id,go to table net_hop}
{Set_next_hop_group_id,
go to table next hop group}
default:
drop
}

Next hop group
table
{
match :
next hop group,hash_val
action :
{Set_next_hop_id,
go to table net_hop}
default:
drop
}

Next hop
table
{
match :
next hop
action :
{set_egrress_rif(ERIF),
set_nh_dstippacket.dst_ip}
{set_nh_dstip(NH_IP),
set_egrress_rif(ERIF)}
default:
drop
}

Metadata
Next hop =NH

Metadata
Next hop group=GR

Metadata
Ingress_vrf=VRF

Metadata
Ingress_rif=IRIF

vbridge

Port/general

bridge

router

tunnel

Ingress port flow

Port <M>

Metadata
IN port =N(ingress phy port)
IN l2_if=N(ingress l2 interface)
Is_tagged=yes/no

Metadata
Is LAG =yes
LAG ID =M
IN l2_if=M

Metadata
If(is_tagged)
VID=PVID
else
VID=packet.VID

Metadata
l2_if type=router

Metadata
Ingress_bridge_port =N

```
Ingress LAG table
{
  match :
        in port
  action :
       ( set is LAG=yes,
        set LAG id<M>
        set l2_if<M>)
  default :
       set is LAG=no
}
```

```
accepted frame type
{
   match :
        l2_if,is_tagged
   action :
       (set PVID
       {drop, forward })
  default:
       drop
}
```

ACL table

```
Ingress l2 interface type table
{
  match :
     {l2_if,vlan}
     {l2_if}
  action :
     (set_1Q_interface_ object(BR_port)}
     (set_1D_interface_ object(BR_port)}
     (set_L3_interface_ object(RIF)}
       default:
     drop
}
```

.1D → 1D Bridge Port<n> → .1D bridge flow

.1Q → Bridge Port<n> → .1Q bridge flow

router → RIF<n> → Router flow

l2_if type

Metadata
Out_if=Y
Out_if_type=tunnel port
Tunnel type- vxlan
bridge_port=tunnel.underlay_if
Dst_ip=dst_id
Src_ip=tunnel.encap.src_ip
Router_rif=tunnel.encap.oerlay_if

Metadata
TAG_mode=TAG

Metadata
Egress
STP= Forwarding

```
Egress  vBridge
 table
{
   match :
      br_port
action :
      set_vlan(vlan)
      set tag mode ()
default:
       Drop
}
```

```
Egress vBridge STP
Table  {
   match :
       bridge_port
   action :
setSTPstate{Discarding,
Learning ,Forwarding}
  default:
set STP state (Discarding)
}
```

```
FDB
Table
{
   match :
       packet.dst_MAC,BR_ID
   action :
{forward(out_if)}
{forward(out_if,dst_ip)}
{set out_it_type(LAG/port)}
   default:
GO unknown unicast table
}
```

UNI

Packet type {MC,UNI}

```
L3 interface
Table
{
   match
:packet.dst_MAC,BR_ID
   action :
Go to Ingress L3
interface table
   default:
       Go to FDB table
}
```

```
Learning FDB
Table
{
   match :
packet.src_MAC,BR_ID
   action :
   default:
       learn_MAC(br_port)
}
```

```
Ingress vBridge STP
Table
{
   match :
      bridge_port
   action :
setSTPstate{Discarding,
      Learning ,Forwarding}
  default:
set STP state (Discarding)
}
```

```
vBrideg
{
   match :
       {bridge port}
   action :
       set BR_ID<X>
default:
       Drop
}
```

Metadata
STP
state=forwarding

Metadata
BR_ID=X

```
Unknown unicast
Table
{
 match :
      BR_ID
 action :
forward(out_if_list)
  default:
      drop
}
```

Out_if==tunnel port → yes

Egress .1D Bridge (tunnel 1) Port<n>

Metadata
VNI=VNI

ACL table(egress port)

```
Tunnel table
 table
{
   match :
      bridge_port
action :
{set_encpap_param(tunnel.encap
param)}
{go to tunnel mapper-vxlan}
 default:
       Drop
}
```

```
Tunnel mapper-vxlan
 table
{
  match :BR_ID

action :
    set_vni(VNI)
  default:
      Drop
}
```

Tunnel (enacap)

RIF<n>

```
Ingress VRF
 table
{
   match :
       ingress_rif
   action :
       set_vrf(VRF)
   default:
       drop
}
```

ACL table

```
Router
 table
{
  match (LPM)   ingress_vrf,dst_ip(prefix)
  action :   ---
{trap,copy_to_cpu,forward,drop}
{Set_next_hop_id,go to table net_hop}
{Set_next_hop_group_id,
          go to table next hop group}
  default:
      drop
}
```

```
Next hop group
 table
{
   match :
        next hop group,hash_val
   action :
{Set_next_hop_id,
          go to table net_hop}
   default:
       drop
}
```

```
Next hop
 table
{
  match :
       next hop
  action :
{set_egrress_rif(ERIF),
       set_nh_dstippacket.dst_ip}
{set_nh_dstip(NH_IP),
       set_egrress_rif(ERIF)}
  default:
      drop
}
```

Metadata
Next hop =NH

Metadata
Next hop group=GR

Metadata
Ingress_vrf=VRF

Metadata
Ingress_rif=IRIF

Ingress port flow