TCP Session Load-balancing in Active-Active HA Cluster

Nishit Shah
Jimit Mahadevia

Agenda

- Defining Active-Active HA Cluster
- Packet Flow
- Load-Balancing
- ARP Problem
- To Do
- Questions/Discussion
- Credits
- Thank You

Active-Active HA Cluster

HA Cluster

- Prerequisites & Definitions
- HA Cluster
  - Group of more than one identical units working virtually as a single unit.
  - Identical in terms of,
    - Same number of NICs
    - Same IP assignment on all the units
    - Same snapshot of iptables/ipset Rules

HA Cluster

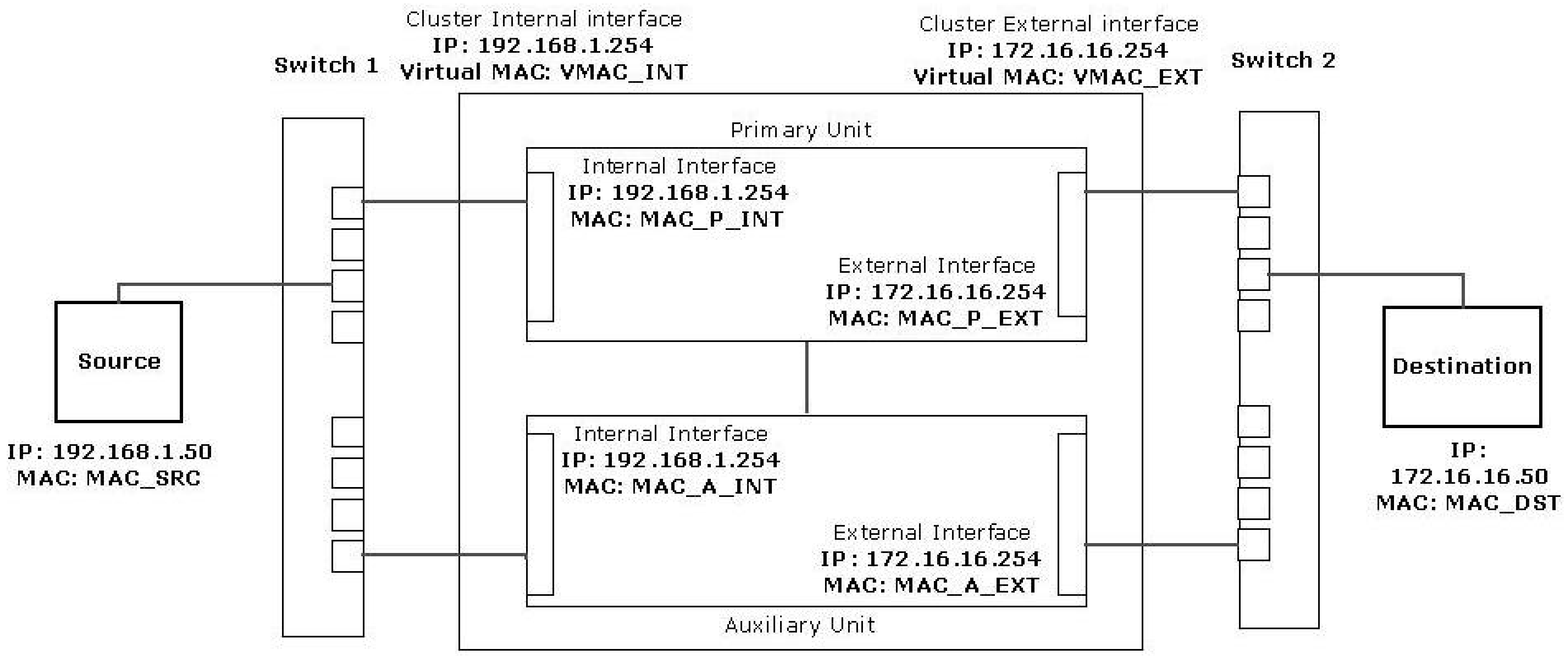
- Active-Active
  - A configuration of HA cluster which consists of a primary unit and one or more auxiliary units connected via dedicated link used for heartbeat and keep-alive.
  - Primary unit is in charge of receiving all the traffic and then balancing the traffic with auxiliary units using some predefined load balancing method.
  - Auxiliary unit(s) receives the traffic to be processed by it from the primary unit. Auxiliary unit simply processes the traffic it receives and does not participate in any load balancing decisions.

HA Cluster

- Virtual MAC
  - It is a MAC address associated with the HA cluster. This address is sent as response when any of the machines make ARP request to HA cluster.
  - There will be one virtual mac address per interface. It is as same as normal mac address. So, one can predefine or auto generate them.
  - Virtual mac address(es) are binded on the primary unit.
  - Auxiliary units are binded with their own physical mac address(es)

HA Cluster

A typical 2-node HA Cluster



Note:
1. Only one internal and one external interface are shown for each unit in the cluster for simplicity. Actual setup may have more interfaces on each unit

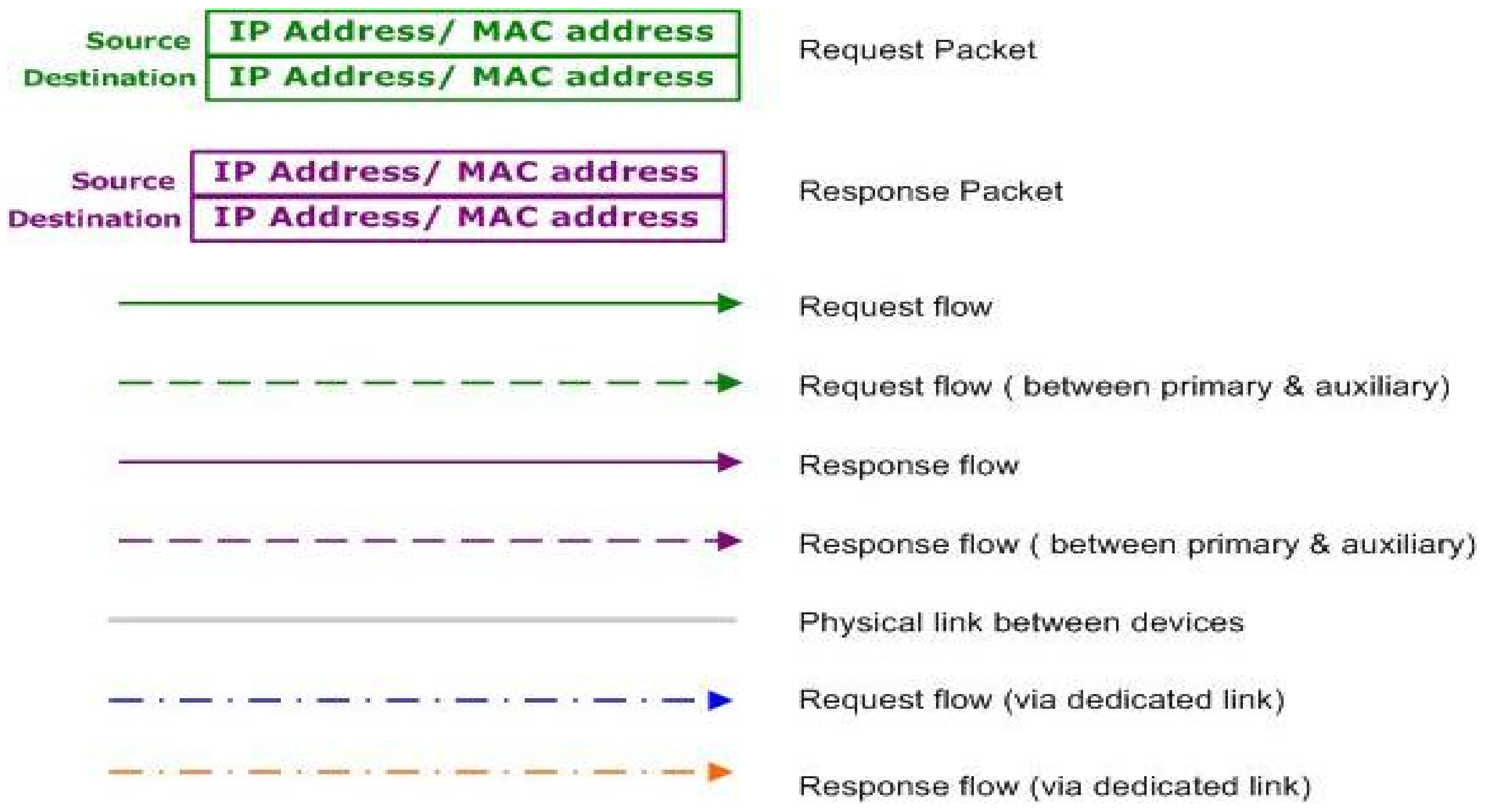
HA Cluster

- Notes
  - ARP will be enabled only on primary unit. On auxiliary it is disabled either using NOARP in ifconfig or dropping ARP packets through arptables.
  - Primary unit needs to have information of physical mac addresses of all the auxiliary units running in Cluster. (Part of cluster join operation)
  - Internal interfaces of both the appliances will be connected to switch 1
  - External interfaces of both the appliances will be connected to switch 2
  - Source is connected to switch 1
  - Destination is connected to switch 2

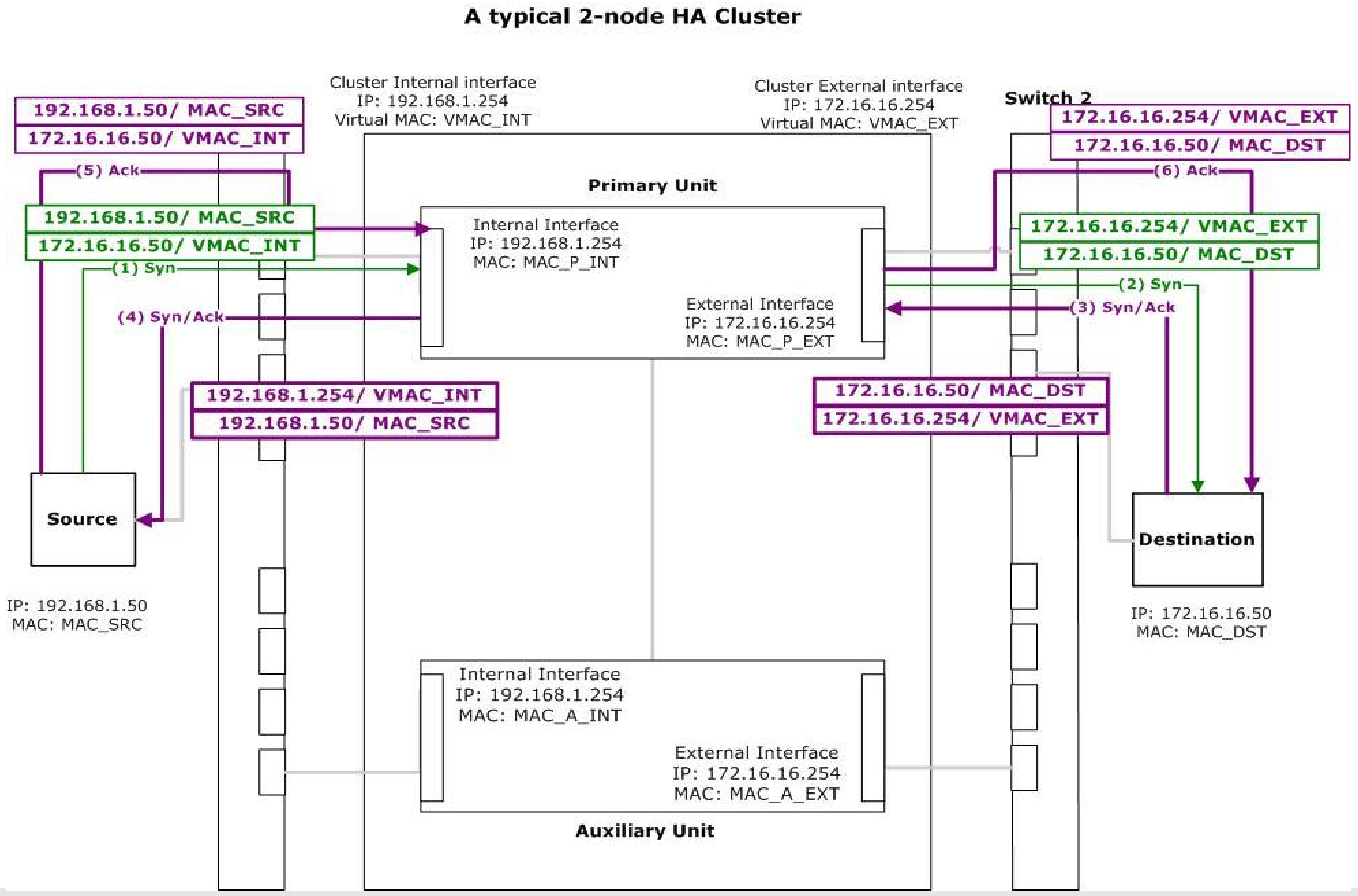
Packet Flow

Packet Flow

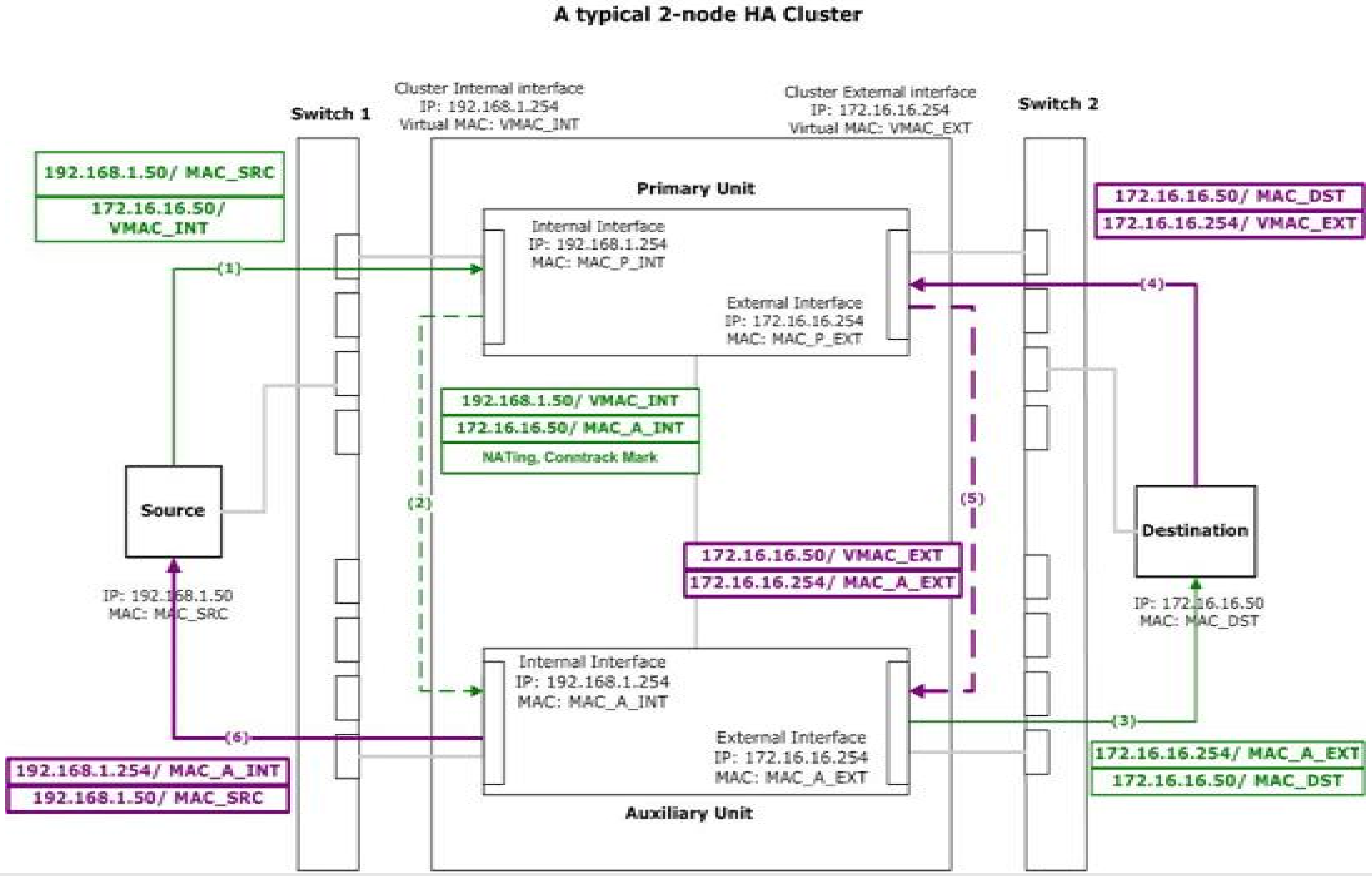
- Legends


Source
IP Address/ MAC address
Destination
IP Address/ MAC address
Request Packet
Source
IP Address/ MAC address
Destination
IP Address/ MAC address
Response Packet
Request flow
Request flow (between primary & auxiliary)
Response flow
Response flow (between primary & auxiliary)
Physical link between devices
Request flow (via dedicated link)
Response flow (via dedicated link)

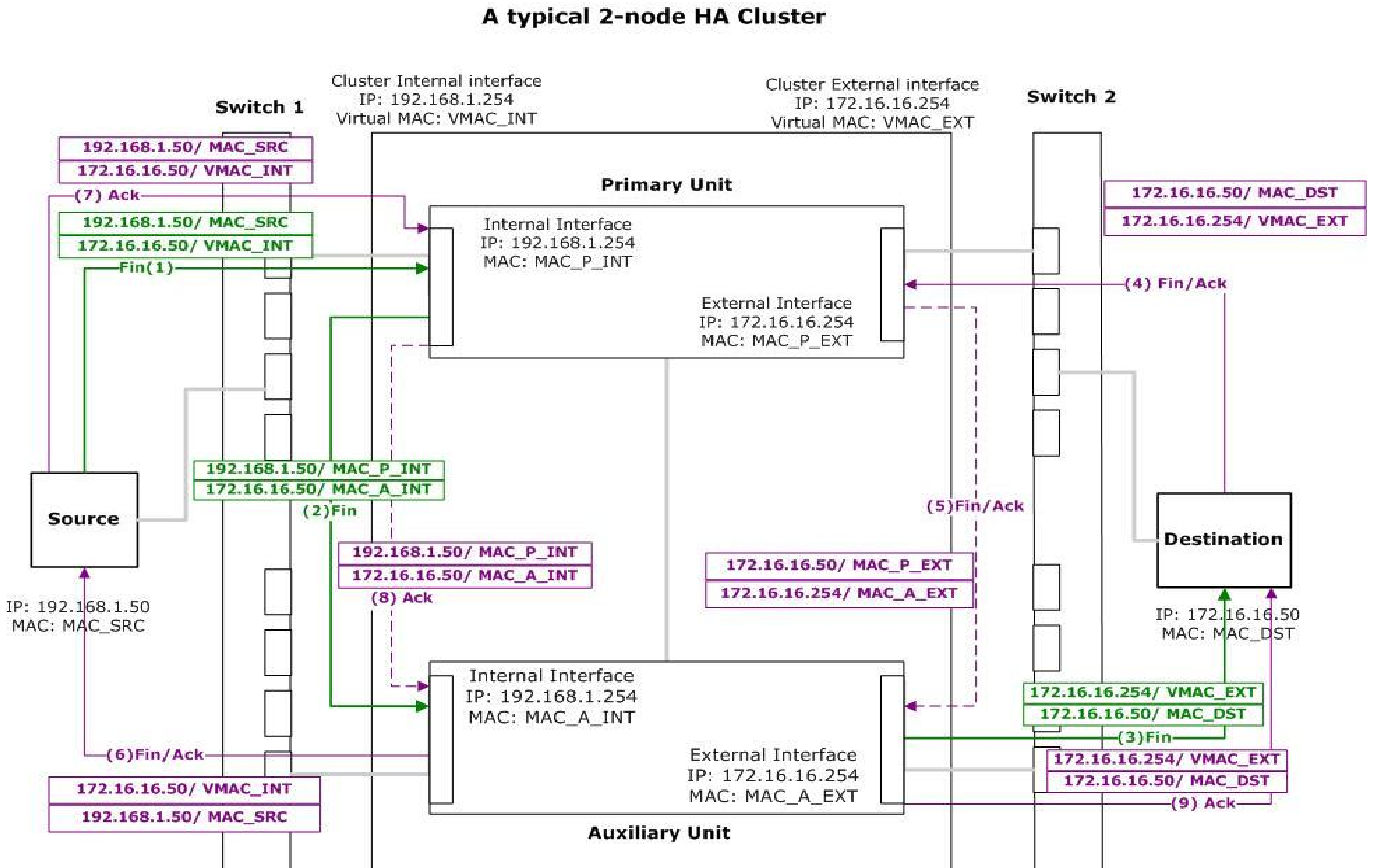
Primary unit is serving the request - (TCP SYN)



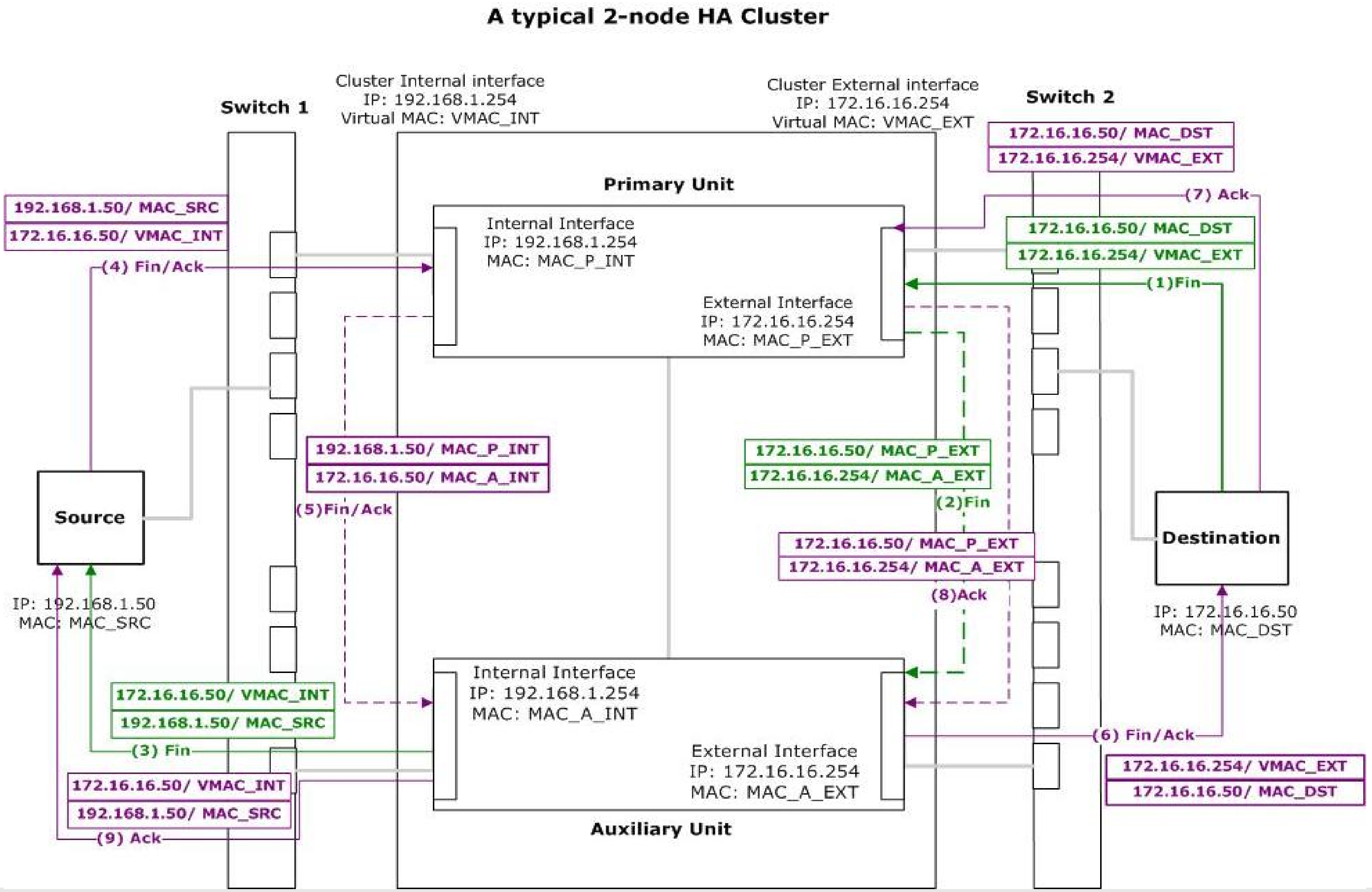
Auxiliary unit is serving the request - (TCP SYN)



Auxiliary unit is serving the request - (Connection termination by Source)



Auxiliary unit is serving the request - (Connection Termination by Destination)



- primary will process the Syn packet entirely and create a connection on itself for it before sending it outside or to auxiliary. Therefore primary will always have connection entry for all the connection to be served by itself as well as by the auxiliary.
- For SYN packets to be sent to auxiliary units, all the decisions taken while processing like NATing and conntrack mark are appended as data in SYN packet and then sent to the auxiliary unit. This is must to prevent split-brain syndrome. i.e. Decision should be taken at single place and other should use them.
- ipt_SYNDATA.ko rule placed in PREROUTING mangle will extract the information like NAT and routing, calls ip_nat_setup_info, copy mark information to skb->mark and nfct->mark, shrink the data and continue the packet traversal.

Load Balancing

Load Balancing

- Load Balancing Module contains,
- ipt_LB.ko (loaded on primary unit)
    - Contains 2 hook functions and one target
    - Target is used in PREROUTING nat to specify the traffic for load balancing.
        - All the units in cluster are given an integer id for identification and __u8 clnodeid field is added in struct nfct.
        - This target will decide which unit will process the TCP session and stores the decision clnodeid field. (Expected connections (like FTP Data) will use master connection's clnodeid.)
        - Currently implemented load-balancing methods
            - Round-robin.
            - Wighted Round-robin.

Load Balancing

- ipt_LB.ko.
  - POSTROUTING hook function is added after conntrack_confirm.
    - If primary unit is processing the TCP session it will do nothing for TCP SYN.
    - If auxiliary unit needs to process the TCP session, it will mangle the SYN packet's source and destination if required and also append the necessary data. (NATing and conntrack mark)
  - PREROUTING hook function added before mangle table is used to send subsequent packets of TCP session directly to auxiliary unit in case of auxiliary unit processing the session (call of helper->help function is added to track the related connections of the respective session) calling NF_HOOK_THRESH(POSTROUTING) bypassing all other hook functions.
  - To send the packets to auxiliary unit's respective interface, hook functions use auxiliary unit's mac address information for routing and neighbor resolution.

Load Balancing

- ipt_SYNDATA.ko target (loaded on auxiliary unit)
  - ipt_SYNDATA target is placed as first rule in PREROUTING mangle.
  - Job of this target is to extract the information added in SYN packet by primary unit, use that information and shrink the data from TCP SYN packet and continue the traversal.

ARP Problem

ARP Problem

- As one or more identical units are on the same network and all require to forward the traffic, all units need to do the ARP queries to resolve the neighbor information.
- Primary unit will never have a problem in ARP queries.
- In case of ARP from auxiliary units,
  - Can't do ARP queries either using physical mac address or virtual mac address as they confuse the connected switch and/or machines.

ARP Problem

- Solution thought of,
- Routed ARP Request
  - Whenever an auxiliary unit needs to know ARP of any machine, it will ask primary via dedicated link to get the ARP of that machine on its behalf.
- Routed ARP Response
  - Whenever primary will get any ARP Response, it will forward this response to auxiliary. Auxiliary will check whether the response is in context of any of the routed ARP request made by it. If yes, it will consider it as Routed ARP response else it will drop that response.

arpreq_proxy

- Two arptables modules have been written
- arpreq_proxy: This module is used for two purposes-
- For sending ARP request form auxiliary to primary through dedicated link using
  - arptables -I OUTPUT --opcode 1 -j arpreq_proxy --flags aux_to_primary --halink <dedicated link>
- It will do following with arp request packet.
  - Change opcode to 201 (reserved)
  - Change the skb->dev to dedicated link
  - Changes source hw address in the Ethernet packet header to physical mac address of dedicated link.
  - Transmit the packet.

arpreq_proxy

- For sending ARP request from primary to outside using,
  - arptables -I INPUT -i <dedicated_link> --opcode 201 -j arpreq_proxy --flags primary_to_external
- It will do following with arp request packet.
  - Gets output device by ip_route_output_key with destination ip present in the arp payload.
  - create a new ARP request packet with,
    - opcode ARPOP_REQUEST, source hw as physical mac address of device got from ip_route_output_key, target and destination hw as NULL, dest_ip same as got in arp payload, device got from ip_route_output_key.
  - Transmit above ARP request over device got using ip_route_output_key and DROP the original packet.

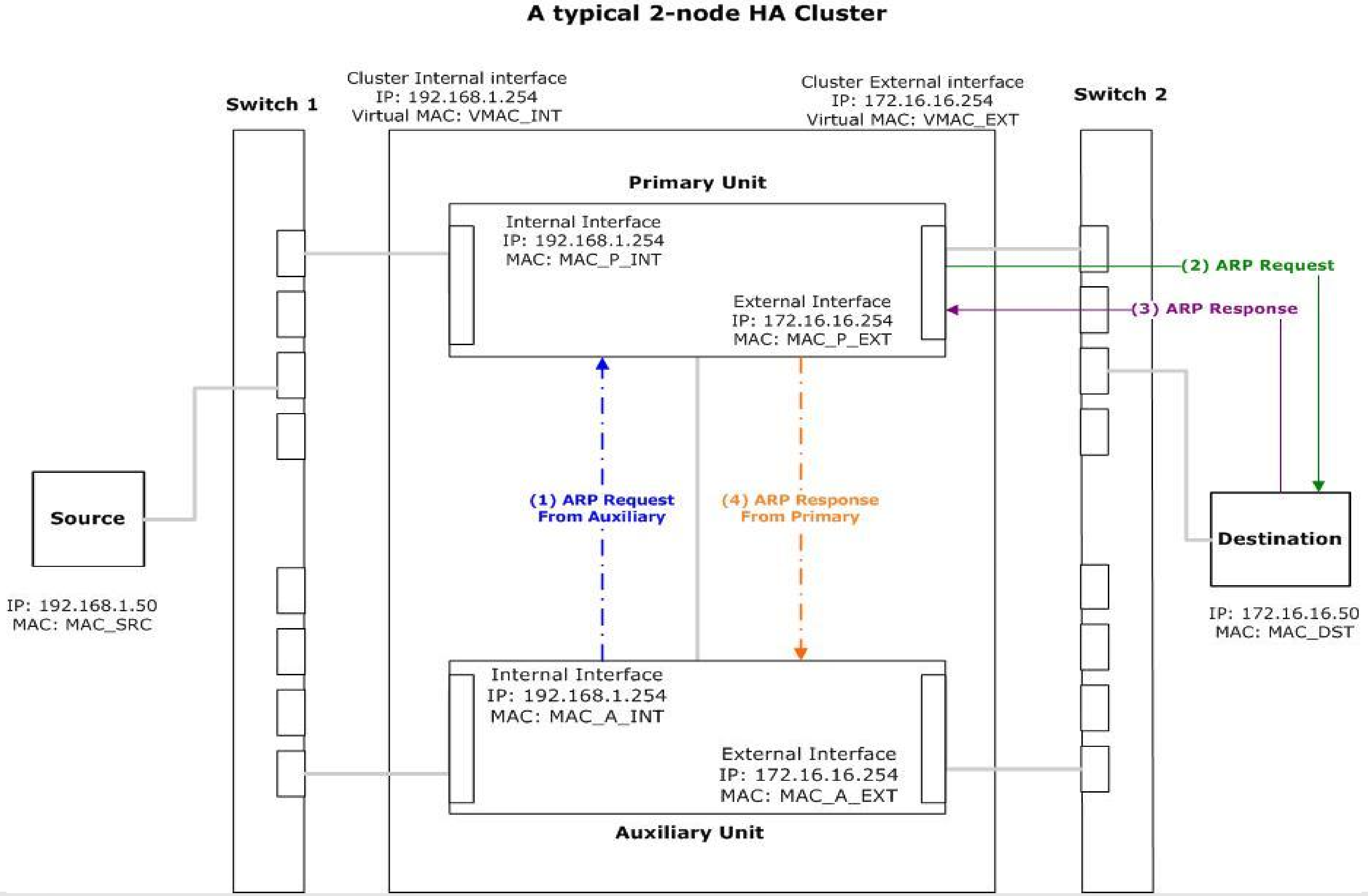
arpreply_proxy

- arpreply_proxy: This module is used for two purposes-
- For sending ARP reply received on primary to auxiliary through dedicated link using,
  - arptables -I INPUT --opcode 2 -j arpreply_proxy --flags primary_to_aux --halink <dedicated_link> --hamac <mac address of auxiliary unit's dedicated link>
- It will do following with arp response packet.
  - create a new ARP response packet with,
    - Opcode 202(reserved), source hw address and srcip as same in payload, target hw address as destination hw address in payload, dest_ip same as got in arp payload, device as halink and destination hw address as hamac.
  - transmit the created packet on dedicated link and submit the original packet to Primary.

arpreply_proxy

- To modify the ARP reply from auxiliary to arp code using,
  - arptables -I INPUT --opcode 202 -j arpreply_proxy --flags aux_to_arp
- It will do following with arp response packet.
  - Gets the output device by ip_route_output_key with source ip present in the arp payload.
  - Changes the skb->dev to the device got from ip_route_output_key.
  - Changes the opcode of arp reply packet to ARPOP_REPLY
  - Continue the packet traversal towards the ARP code.

Auxiliary unit is serving the request - (ARP)



To Do

- arpreq_proxy
  - Use primary's arp cache first for arp queries from auxiliary units.
- 2-node cluster is working. n-node cluster !!!
  - arpreply_proxy: primary to auxiliary response need to think of.

Questions/Discussion

???

Credits

- Cyberoam Team. (Richa, Krunal and other team members)
- Netfilter Core Team.
- Prachi Shah for creating all the presentation images.

Thank You