DPDK

DATA PLANE DEVELOPMENT KIT

DPDK Load Balancers

RSS H/W LOAD BALANCER –
DPDK S/W LOAD BALANCER –
L4 LOAD BALANCERS –
L7 LOAD BALANCERS

NOV 2018

Contact

- Vincent, Jay L - Your Contact For Load Balancer Follow up
  jay.L.vincent@intel.com
- M Jay - For Feedback & For DPDK Cookbook
- Muthurajan.Jayakumar@intel.com

Contributions And Acknowledgements

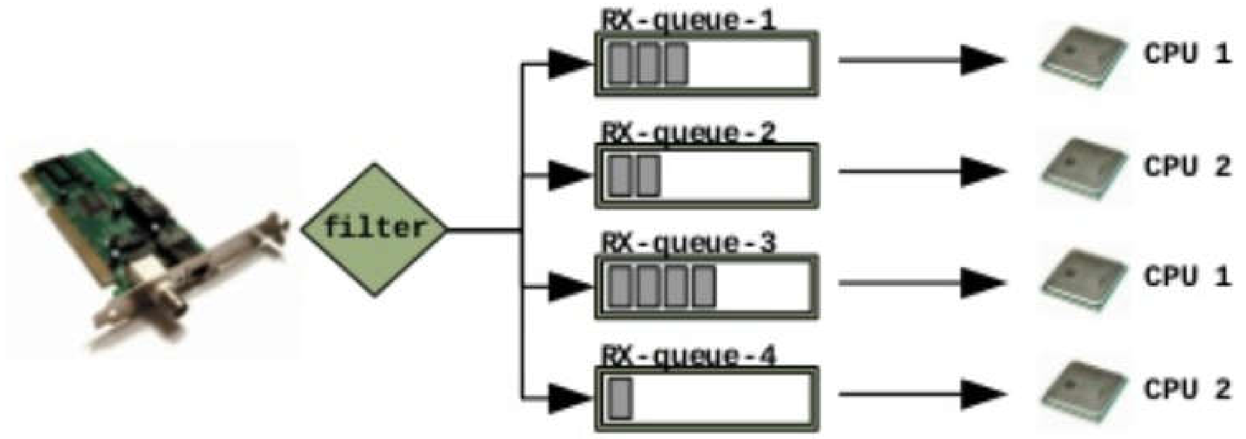
• M Jay	Jay L Vincent	John M Morgan	
• Bob Ghaffari	Jianwei Ma	Ray Kinsella	
• Walter Gilmore	DPVS Community	Brian Aherne	
• Jim St Leger	Tim O’Driscoll	Hongjun Ni	
• Jokul Li	Steve Cunming	Liu, Yu Y	
• Edward Verplanke	DPDK Community	Network Builders	
• Sujata Tibrewala	Heqing Zhu	Bill Carlson	Alejandro Corredor
• Ed Pullin	Zhang Pan1	Mike Glynn	And Many others..

- Great Documentation in https://github.com/iqiyi/dpvs
- http://ja.ssi.bg/#lvsgw
- And Many others

Agenda

- RSS Load Balancer – NIC
- DPDK s/w Load Balancer – Cache Sharing
- L4 Load Balancer – Fast Forwarder
- Balance The Load? Or Benefit From Locality? - Stickiness
- Why to have Man in the Middle? – Direct Response
- Quiz - Can you have 2 devices with same IP addresses in same subnet?
- Direct Response, Full NAT – What? Where?
- One Arm Versus Two Arms Load Balancer – Devil is in the Details
- Call To Action

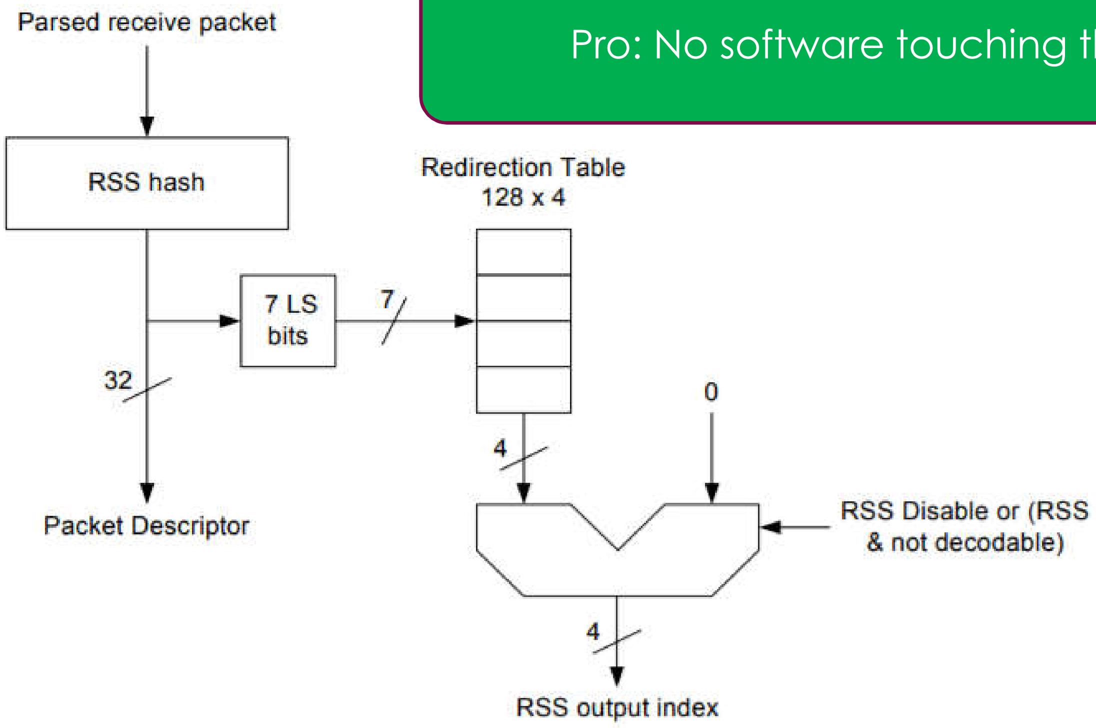
Receive Side Scaling – NIC Load Balancing**



NIC H/W Distributes Packets to different CPUs through queues

Receive Side Scaling – NIC Load Balancing

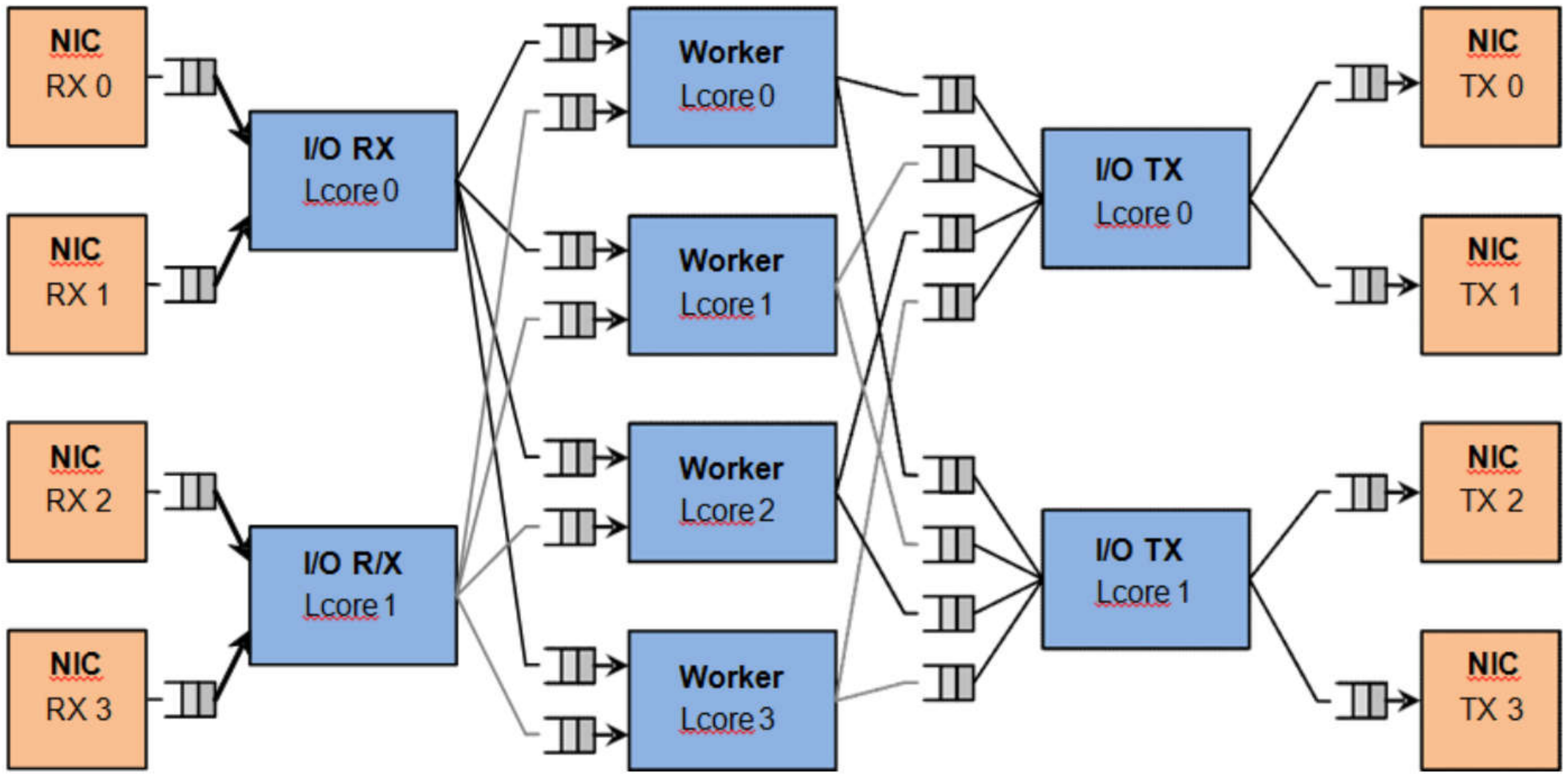
Pro: No software touching the packets for distribution decision



RSS Block Diagram

Con: Scope Is Only For Packet Types That Are Supported by NIC H/W RSS

DPDK S/W Load Balancer



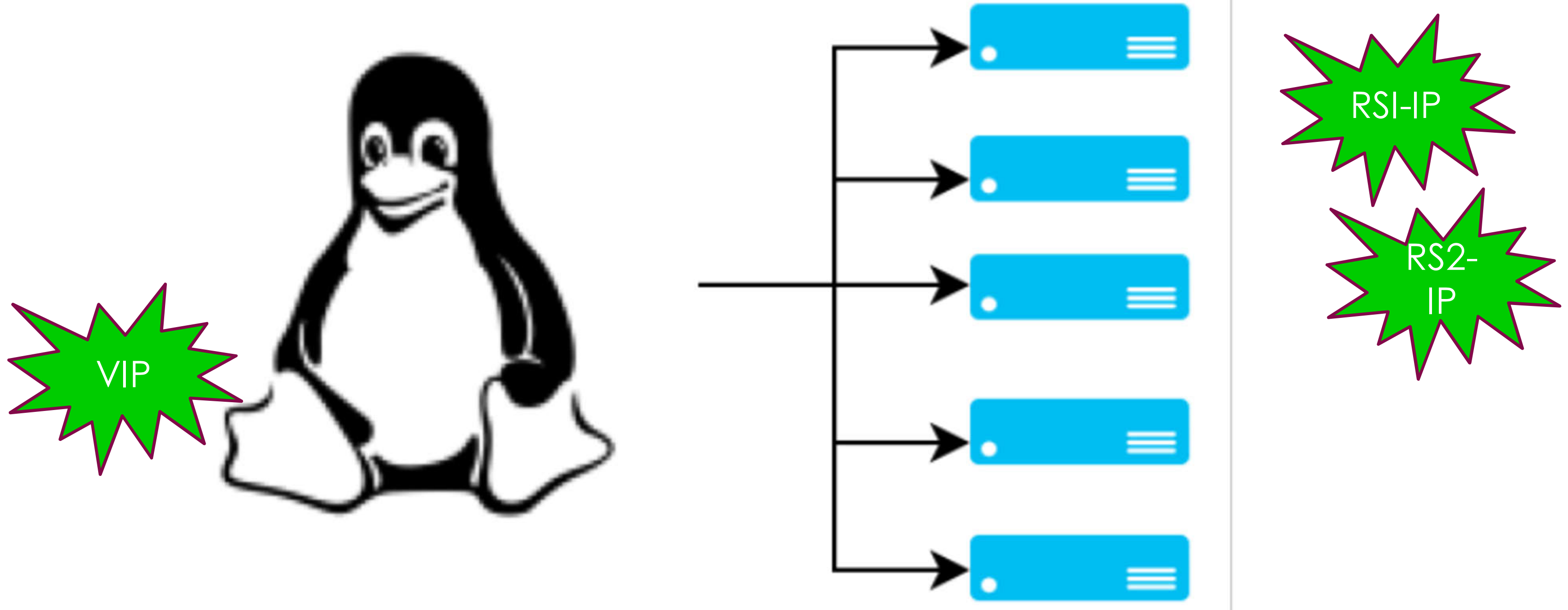
Since S/W, packet types not supported by NIC also can be supported

DPDK S/W Load Balancer

- I/O Core Decides which Worker Core to Balance the Load with.
  - In this case worker core transitions the cache line from “M” state to “S” state
- Exclusive to Shared
  - Packet lookup (5 tuple) – I/O Core Cache Pulls The Packet (Header) – “E” state
  - Worker core transitions the cache line to “S” state. – “E” state to “S” state
- Transition from Modified State (More Expensive)
  - In case I/O Core wanted to store meta data as the result of lookup – “M” state
  - Worker core transitions the cache line from “Modifed” State in I/O Core - Expensive.

Use Prefetch Hint

L4 Load Balancer – Kernel Space – Full NAT



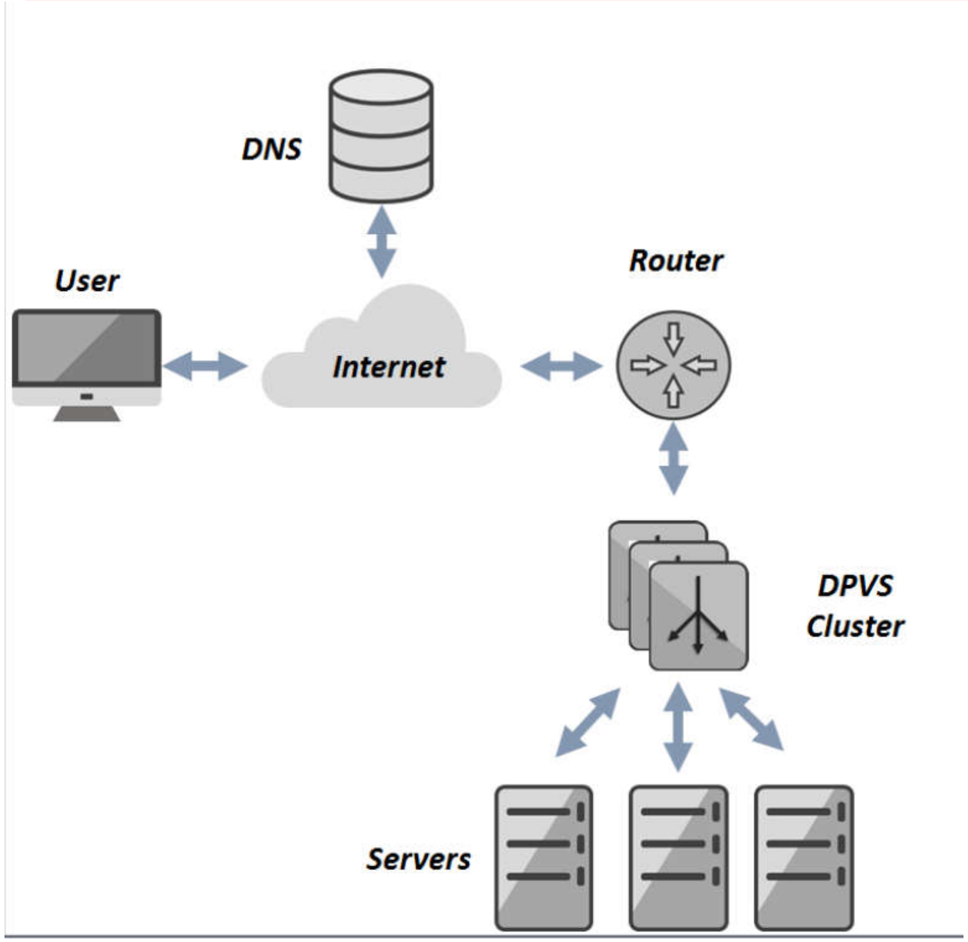
VIP – “Virtual IP” = “Service IP; ”RS1, RS2 – “Real Server IP”

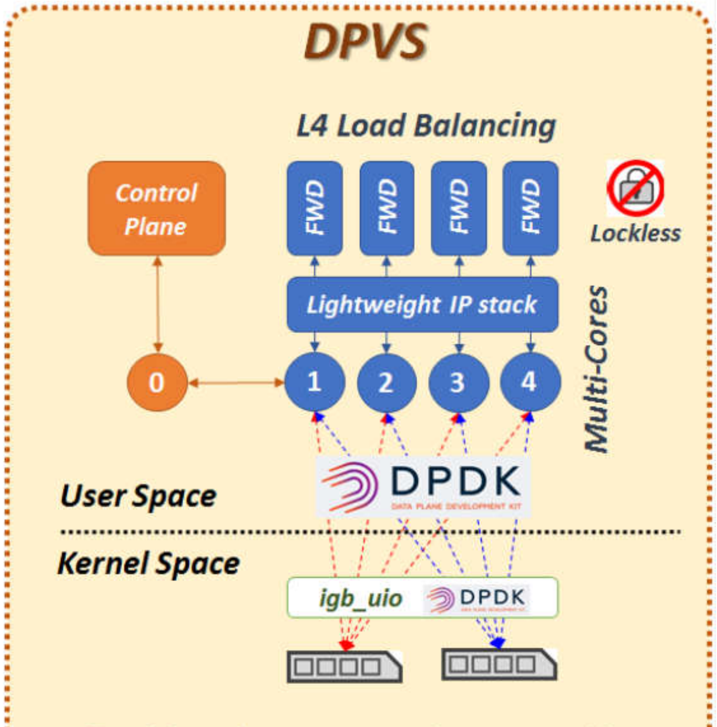
The benefit of Full NAT – Servers can be used as is. No change needed

What About DPDK Based Load Balancer?

- DPVS - DPDK Based Load Balancer
  - DP – From DPDK VS – From Linux Virtual Server
- https://github.com/iqiyi/dpvs
- Runs on commodity Linux Servers
- Easily Scales Up / Down – Simply Add / Remove Servers
- DPVS matches the packets to their required services and matches/ balances to “Real” servers
- Optimized for packet processing performance – DPDK Based
  - Tested With 10 / 25 / 40 Gig NICs
- Control Plane using IPVS (IP Virtual Server) - http://www.linuxvirtualserver.org/software/ipvs.html#kernel-2.6

DPVS - Data Path Benefiting From DPDK





- ***High Performance:*** Multi-cores, Lockless
- ***All-in-One:*** Reverse Proxy, SNAT
- ***Security:*** SYN-proxy, blacklist
- ***IDC Friendly:*** VLAN, Bonding
- **QoS:** conn limitation, traffic control

DPVS Features & DPDK High Performance Technology

Major features of DPVS including:

- *L4 Load Balancer*, including FNAT, DR mode, etc.
- Different *schedule algorithm* like RR, WLC, WRR, etc.
- User-space *Lite IP stack* (IPv4, Routing, ARP, ICMP ...).
- *SNAT* mode for Internet access from internal network.
- Support *KNI*, *VLAN*, *Bonding* for different IDC environment.
- Security aspect, support *TCP syn-proxy*, *Conn-Limit*, *black-list*.

* QoS: *Traffic Control*

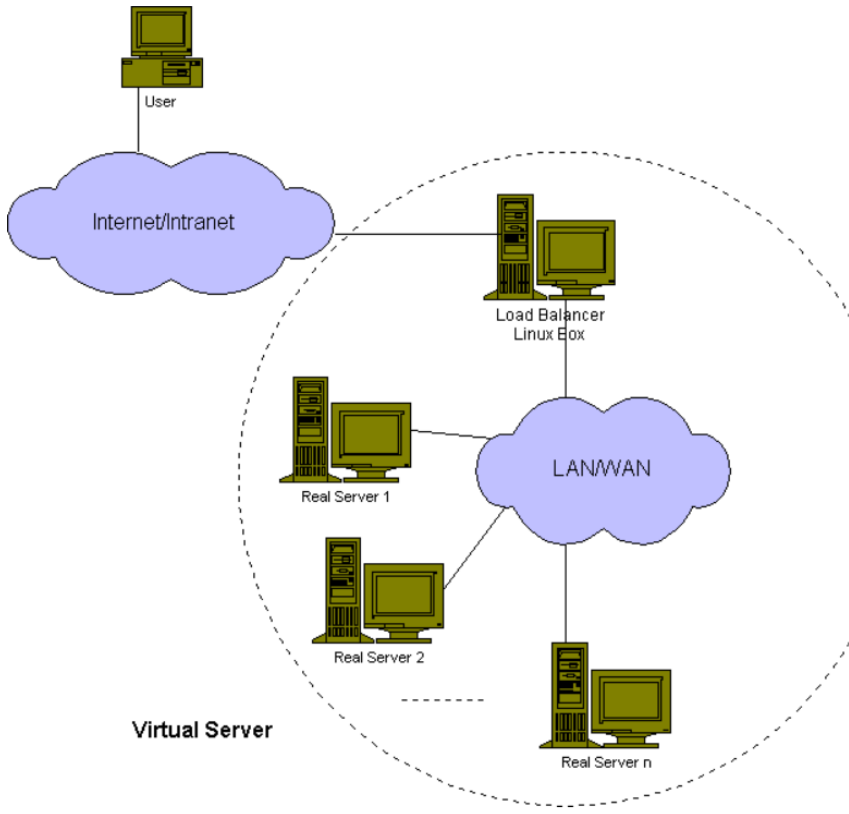
DPDK High performance Techniques:

- Kernel by-pass (user space implementation)
- Share-nothing, per-CPU for key data (Lockless)
- RX Steering and CPU affinity (avoid context switch)
- Batching TX/RX
- Zero Copy (avoid packet copy and syscalls).
- Polling instead of interrupt.
- lockless message for high performance ICP.
- other techs enhanced by DPDK.

https://github.com/iqiyi/dpvs

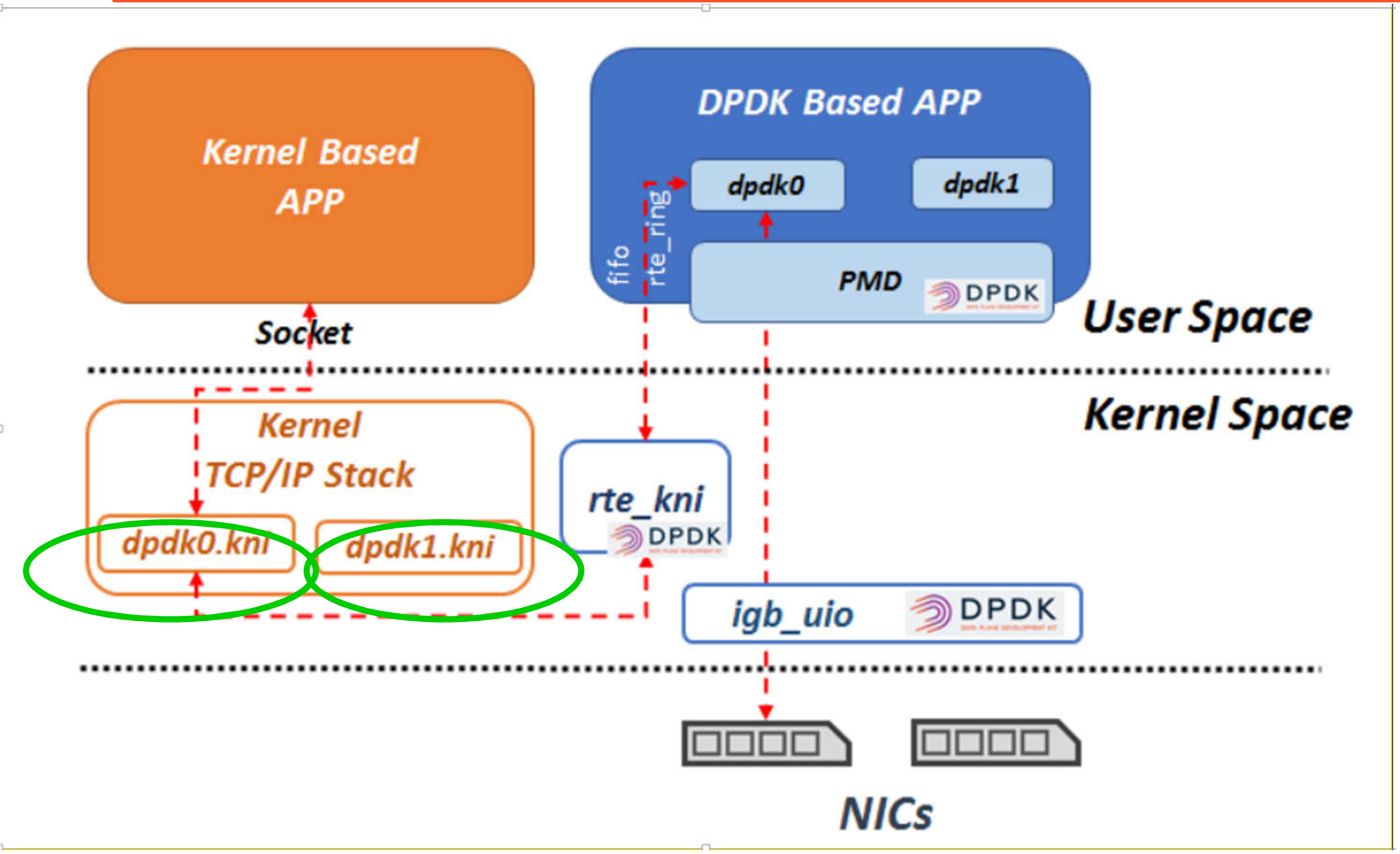
Control Path Benefiting From IP Virtual Server (IPVS)

- IPVS Implements transport-layer load balancing inside the Linux kernel, so called Layer-4 switching.
- IPVS running on a host acts as a load balancer at the front of a cluster of real servers.
- IPVS can direct requests for TCP/UDP based services to the real servers
- Thus making services of the real servers to appear as a virtual service on a single IP address



http://www.linuxvirtualserver.org/software/ipvs.html#kernel-2.6

DPVS - Data Path (DPDK) + Control Path (IPVS)



ONE ARM
- Both clients & Servers
    - On same side
- Dpdk0.kni alone enough

TWO ARMS
- Both clients & Servers
    - On same side
- Dpdk0.kni & DPDK1.kni

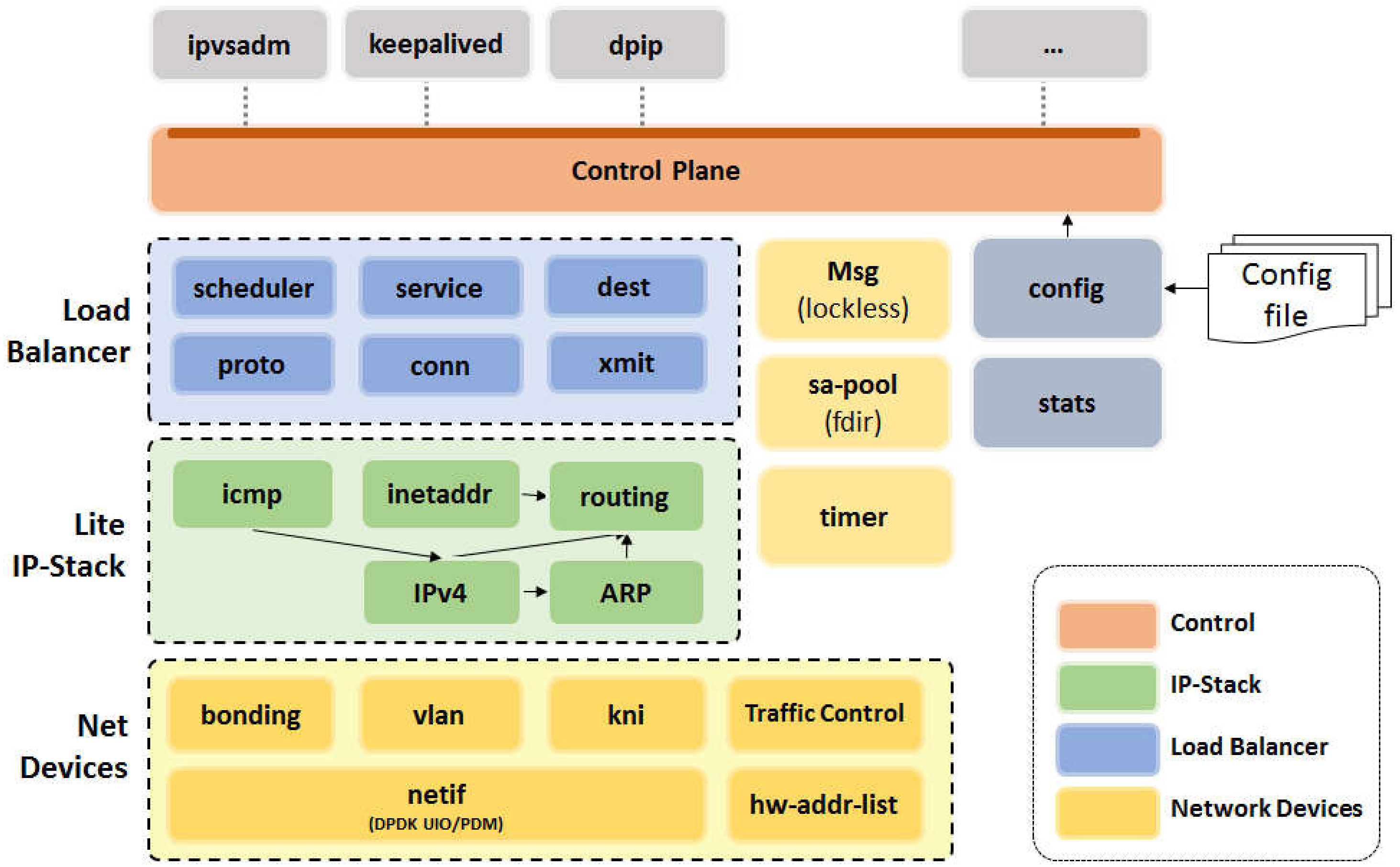
One- arm and Two-arm Terminology

- ***Two-arm***
- The term *two-arm* means, you have clients in one side of *load-balancer* (LB)
- And servers (RS) in another side,
- LB forwards packets between its two logical network interfaces.
- For example, *WAN-to-LAN* load balancing.

- ***One-arm***
- On the other hand, *one-arm* means all clients and servers are in same side of load-balancer.
- LB forwards traffic through the same logical network interface.

- *Logical interface* (or *device*) could be
- Physical DPDK interface
- DPVS virtual devices like
-     *bonding*,
-     *vlan* and
-     T*unnel* devices.

DPVS With IPVS - Block Diagram



Spread To Servers? Or Benefit From Stickiness?

- Flow Pinning has the benefit of Locality
- Caching Benefits thereof.

- When Spreading the Load, say for instance **Round Robin**
- You trade locality for overall utilization.

- What Other Scheduling Algorithms Are Out There?
- **Least Connected**
- **Tagging Weight To The Server**

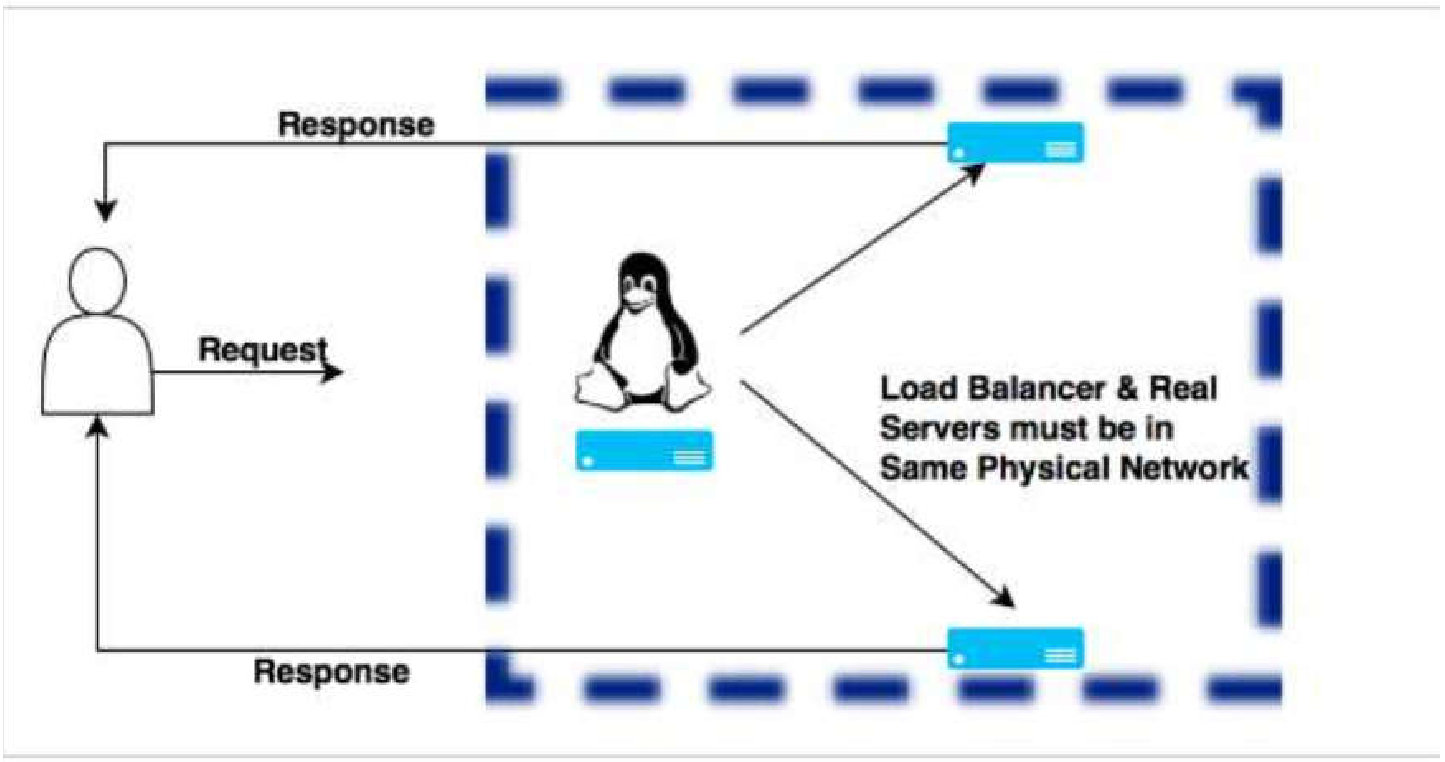
That Is About Scheduling. What About Data Flow?

Forwarding Methods - Thin Request. Bulky Response

- For Single Request, server returns multiple objects.
- So Why Have Load Balancer in the path of bulky Response?
- Why not Load Balancer Become Transparent During Server Response?
- That is **Direct Response Server – DRS**
- How?

Why To Have Man In The Middle During Response?

Request Is Served By Load Balancer But Not Response


Response
Request
Load Balancer & Real Servers must be in Same Physical Network
Response

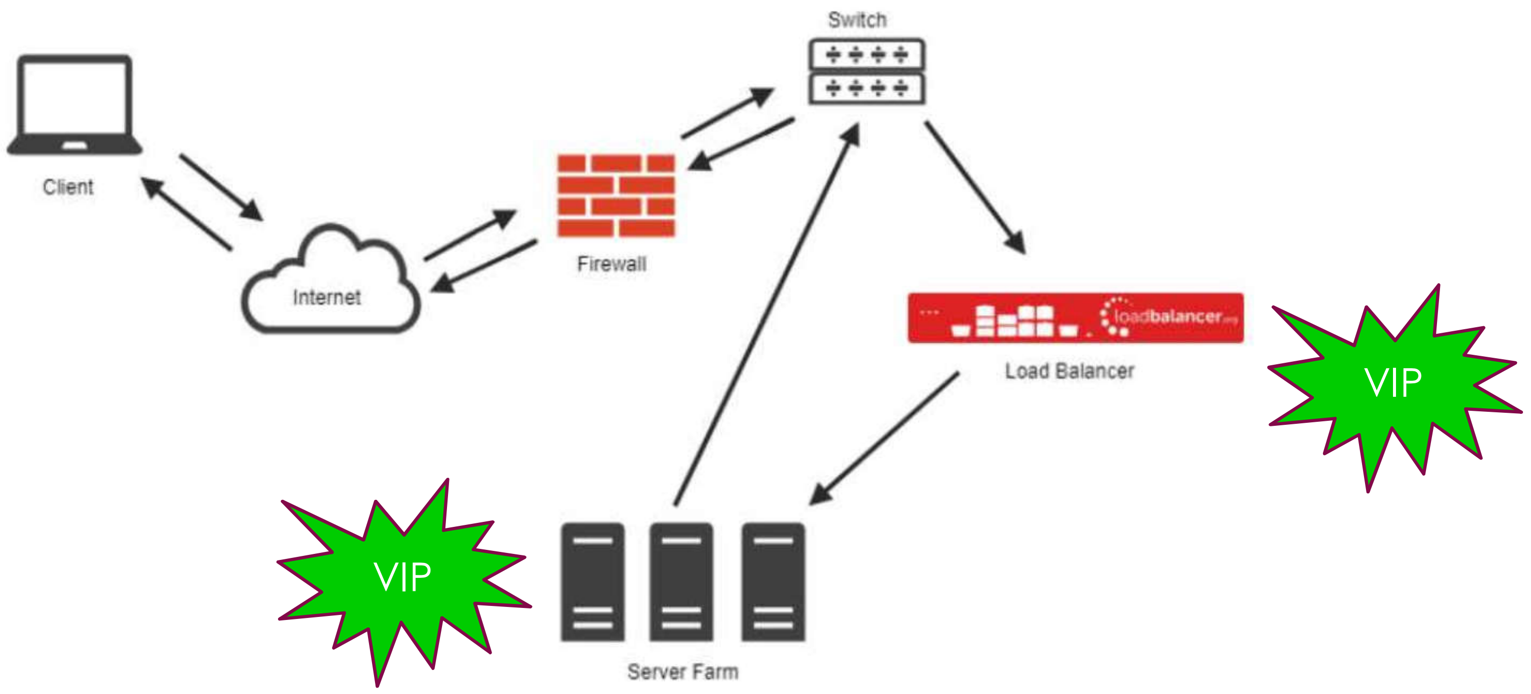
Ready For Quiz?

Quiz

- **<u>Quiz</u>**
- Can you have 2 devices with same IP addresses in same subnet?

- If no, Why not?
- If yes, how can it?

Load Balancer & Server Farms Have Same Service IP Address



PRO: Great For Scalability

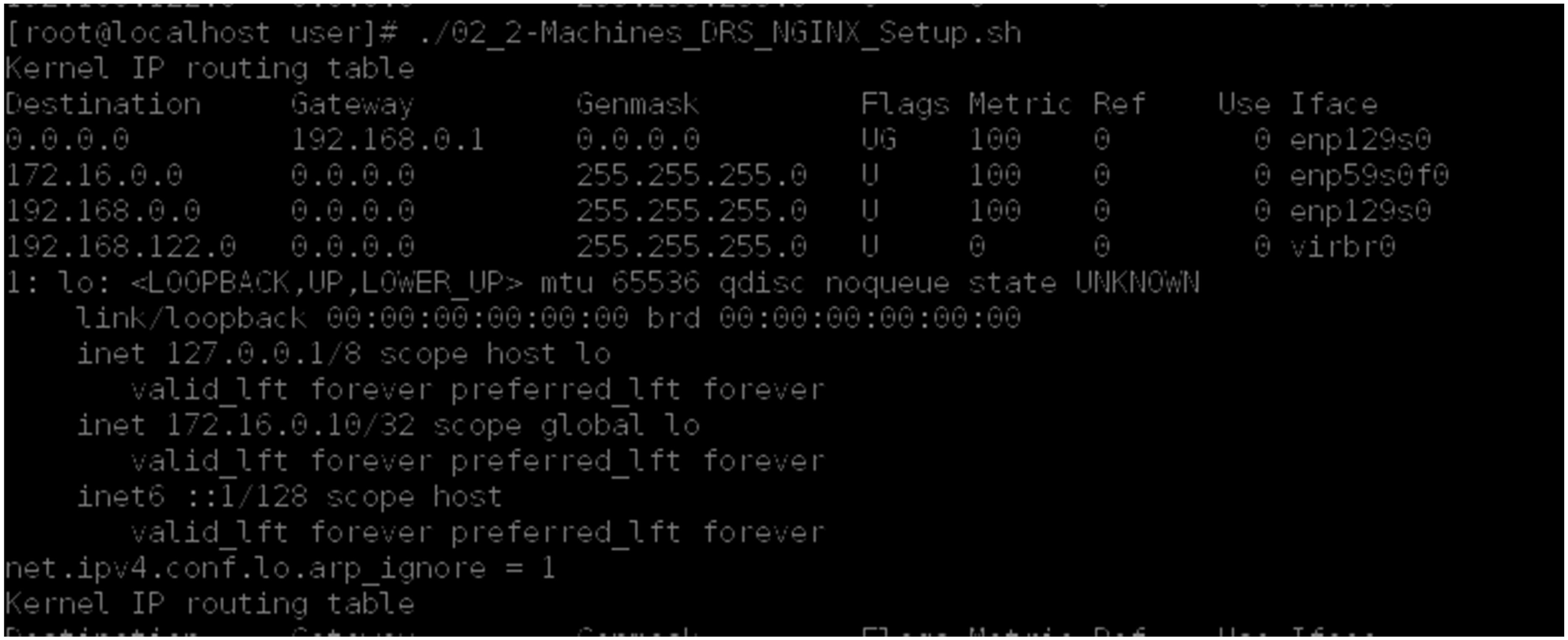
Consideration: Real Servers must host the service at VIP (rather than in RS IP)

Role Of Dummy Interface In Direct Routing

- **<u>Direct Routing:</u>**
- Packets from end users are forwarded directly to the real server. The IP packet is not modified
- **So, the real servers must be configured to accept traffic for the virtual server's IP address.**
- **This can be done using a dummy interface or packet filtering to redirect traffic addressed to the virtual server's IP address to a local port**.
- The real server may send replies directly back to the end user.
- Thus, the Load Balancer does not need to be in the return path.

```
net.ipv4.conf.lo.arp_ignore=1
```

net.ipv4.conf.lo.arp_ignore=1



```
[root@localhost user]# ./02_2-Machines_DRS_NGINX_Setup.sh
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.0.1     0.0.0.0         UG    100    0        0 enp129s0
172.16.0.0      0.0.0.0         255.255.255.0   U     100    0        0 enp59s0f0
192.168.0.0     0.0.0.0         255.255.255.0   U     100    0        0 enp129s0
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet 172.16.0.10/32 scope global lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
net.ipv4.conf.lo.arp_ignore = 1
Kernel IP routing table
```

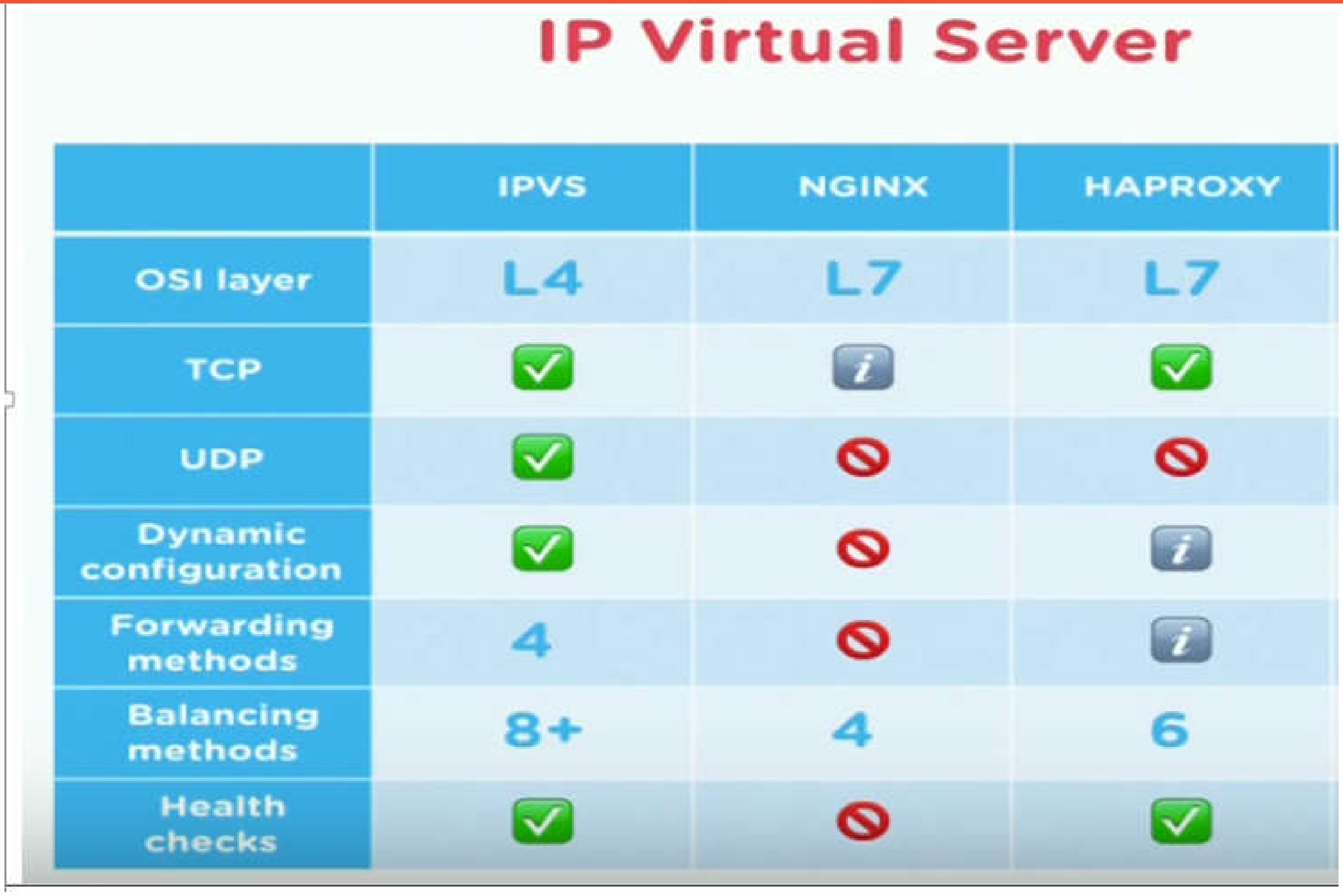
Tunnelling Versus Direct Response

- **IP-IP Encapsulation (Tunnelling):**
- The main advantage of using tunnelling is that real servers can be on a different networks.
- Allows packets addressed to an IP address to be redirected to another address, possibly on a different network.
- In the context of layer 4 switching the behavior is very similar to that of direct routing
- Except that when packets are forwarded they are encapsulated in an IP packet, rather than just manipulating the Ethernet frame.

Direct Response, Full NAT – What? Where?

- **Direct Response Use Case Scope:**
  - **Higher bandwidth During Response Compared To Request**
  - Configuring Server Is OK – apr_ignore=1
  - **Service should be hosted in VIP Server address**

- **Full NAT Use Case Scope**
  - **Off The Shelf Server should be used as it is**.
  - No change to Server Configuration
    - Hosting at Default RS IP address
  - **Note that Rewrite happens for both Source IP Address and Destination IP Address**



IP Virtual Server

	IPVS	NGINX	HAPROXY
OSI layer	L4	L7	L7
TCP	✓	ℹ	✓
UDP	✓	🚫	🚫
Dynamic configuration	✓	🚫	ℹ
Forwarding methods	4	🚫	ℹ
Balancing methods	8+	4	6
Health checks	✓	🚫	✓

Note: Reference IPVS Documentation. Some of the above data may be dated.

One Arm Vs Two Arm – Devil Is In The Details

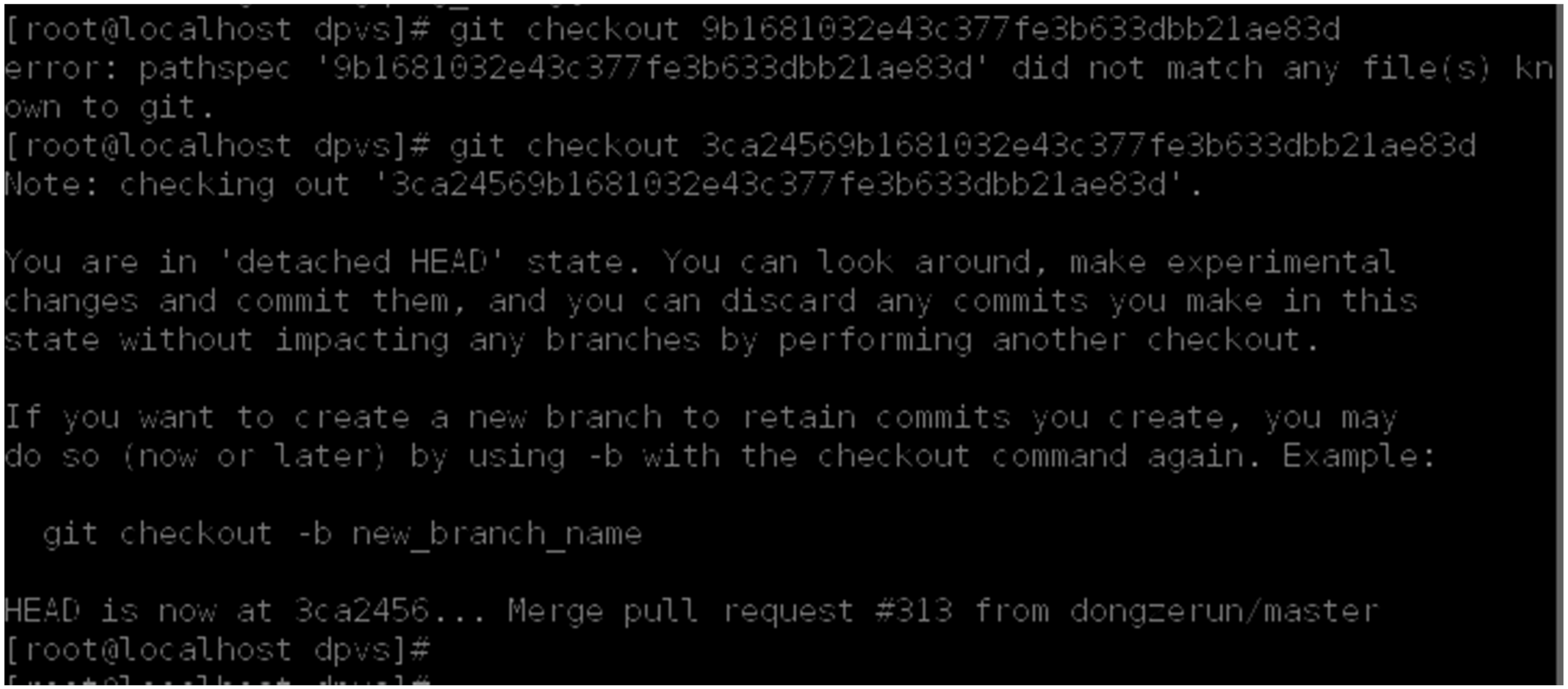
- **One arm:** The real servers can send reply packets directly to the end users without them needing to be altered by the Load Balancer.
- Thus, we saw that LB does not need to be the gateway for the real servers.
- **Two arm:** However, in some situations (say Two Arm), for instance
  - because the load balancer really is the gateway to the real server's network.
- In such case, it is desirable to route return packets from the real servers via LB.
- The source address of these packets will be the VIP.
- However the VIP belongs to an interface on the LB.
- Thus, it will drop the packets as being bogus.
- There are several approaches to this problem.
- Probably the best is to apply a kernel patch supplied by Julian Anastasov.
- This adds proc entries that allow this packet-dropping behaviour to be disabled on a per-interface basis.
- This patch can be obtained from http://www.ssi.bg/~ja/#lvsgw

Current Status - DPVS

- **<u>10 Gig NIC</u>**
- Successfully builds.
- Verified FNAT works Successfully
- Verified DRS works Successfully

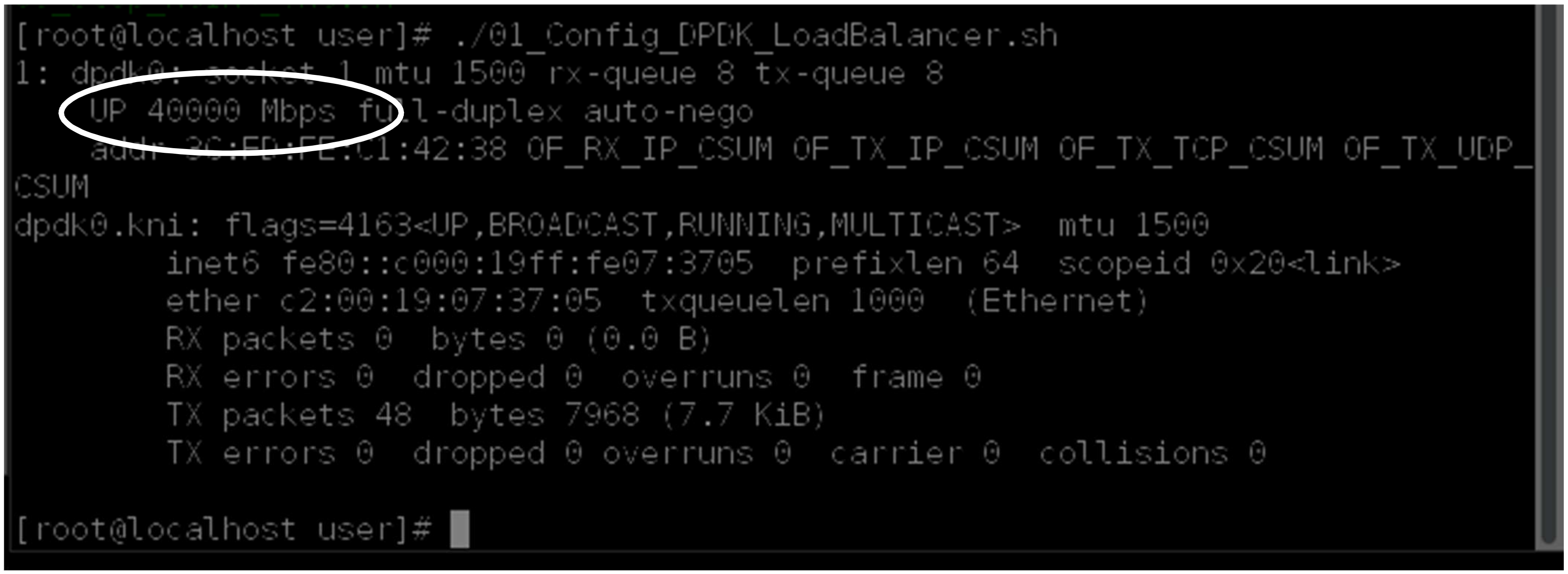
- **<u>40 Gig NIC</u>**
- Successfully builds – Screenshot on next page

Git checkout 9b16……ae83d



```
[root@localhost dpvs]# git checkout 9b1681032e43c377fe3b633dbb21ae83d
error: pathspec '9b1681032e43c377fe3b633dbb21ae83d' did not match any file(s) kn
own to git.
[root@localhost dpvs]# git checkout 3ca24569b1681032e43c377fe3b633dbb21ae83d
Note: checking out '3ca24569b1681032e43c377fe3b633dbb21ae83d'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b new_branch_name

HEAD is now at 3ca2456... Merge pull request #313 from dongzerun/master
[root@localhost dpvs]#
```

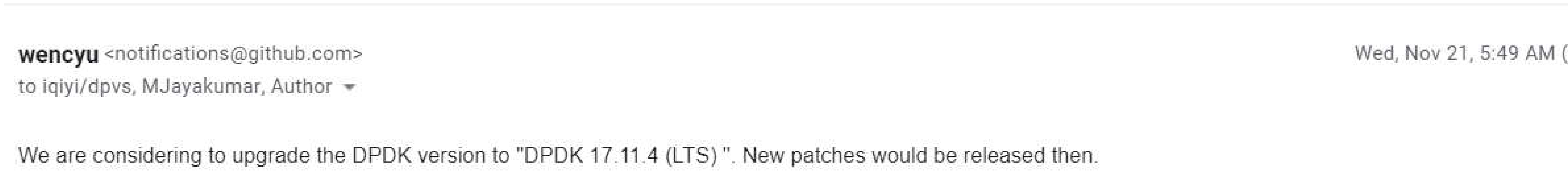
Status – 40 Gig NIC – Successful Build Of Load Balancer



```
[root@localhost user]# ./01_Config_DPDK_LoadBalancer.sh
1: dpdk0: socket 1 mtu 1500 rx-queue 8 tx-queue 8
   UP 40000 Mbps full-duplex auto-nego
   addr 3C:FD:FE:C1:42:38 OF_RX_IP_CSUM OF_TX_IP_CSUM OF_TX_TCP_CSUM OF_TX_UDP_
CSUM
dpdk0.kni: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::c000:19ff:fe07:3705  prefixlen 64  scopeid 0x20<link>
        ether c2:00:19:07:37:05  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 48  bytes 7968 (7.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@localhost user]#
```

40 Gig NIC Load Balancer – Next Steps

- Community Is Considering to upgrade the DPDK Version to "DPDK 17.11.4 (LTS). New patches would be released then.



wencyu <notifications@github.com>
to iqiyi/dpvs, MJayakumar, Author

Wed, Nov 21, 5:49 AM (

We are considering to upgrade the DPDK version to "DPDK 17.11.4 (LTS) ". New patches would be released then.

L4, L7 Load Balancers – Next Step

- Interested to reduce infrastructure cost?
- With L4 Load Balancers?
- With L7 Load Balancers?

Call To Action

- Want To Benefit From Cost Reduction of Network Appliances?.
- Interested in Do it yourself white box load balancer?
- Much of this work is being driven by innovative Cloud Service Providers in PRC
    - To reduce cost of network appliances in their infrastructure

Contact

- Jay L Vincent - Your Contact For Load Balancer Follow up
- jay.L.vincent@intel.com
- M Jay - For Foils Feedback
- Muthurajan.Jayakumar@intel.com

**Acknowledgements & References

- https://www.slideshare.net/ThomasGraf5/linuxcon-2015-linux-kernel-networking-walkthrough
- https://docs.fd.io/vpp/17.07/lb_plugin_doc.html
- https://software.intel.com/en-us/article/get-the-dpdk-cookbook
- https://ai.google/research/pubs/pub44824
- http://ja.ssi.bg/#lvsgw