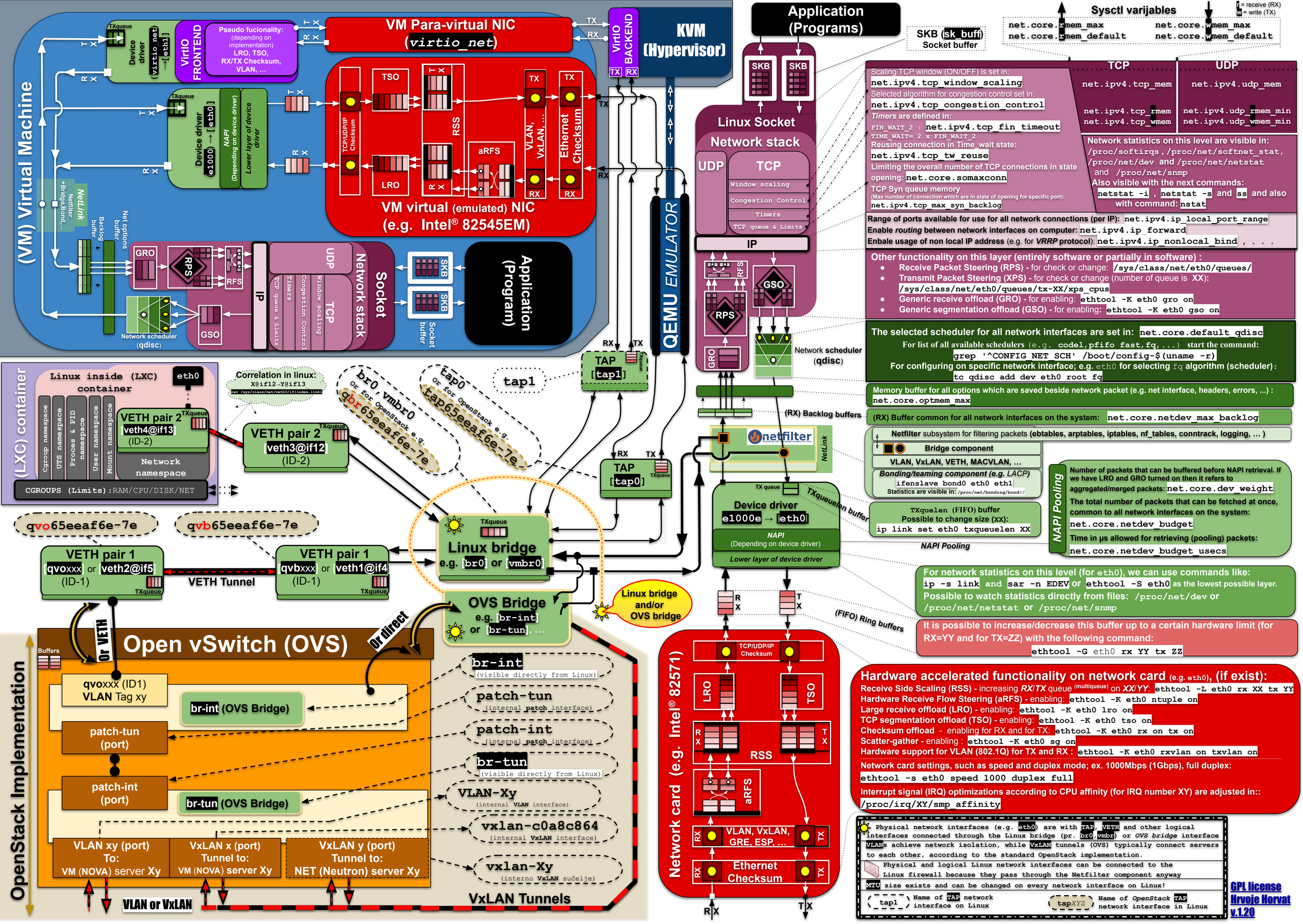
(VM) Virtual Machine
VM Para-virtual NIC (virtio_net)
VirtIO FRONTEND
Pseudo fucionality: (depending on implementation) LRO, TSO, RX/TX Checksum, VLAN, ...
VirtIO BACKEND
KVM (Hypervisor)
Device driver (virtio_net) → [eth1]
Device driver (e1000) → [eth0]
NAPI (Depending on device driver)
Lower layer of device driver
TSO
LRO
RSS
aRFS
TCP/UDP/IP Checksum
VLAN, VxLAN, ...
Ethernet Checksum
VM virtual (emulated) NIC (e.g. Intel® 82545EM)
NetLink
Netfilter +Bridge,Bond,...
Backlog buffer
Net options buffer
GRO
RPS
RFS
GSO
IP
UDP
TCP
Window scaling
Congestion Control
Timers
TCP queue & Limits
Network stack
Socket
SKB
Socket buffer
Application (Program)
Network scheduler (qdisc)
QEMU EMULATOR
TAP [tap1]
Application (Programs)
Linux Socket
Network stack
Network scheduler (qdisc)
(RX) Backlog buffers
netfilter
Device driver e1000e → [eth0]
TXqueuelen buffer
(FIFO) Ring buffers
Linux bridge e.g. [br0] or [vmbr0]
OVS Bridge e.g. [br-int] or [br-tun], ...
Linux bridge and/or OVS bridge
VETH pair 1 [qvoxxx] or [veth2@if5] (ID-1)
VETH pair 1 [qvbxxx] or [veth1@if4] (ID-1)
VETH Tunnel
qvo65eeaf6e-7e
qvb65eeaf6e-7e
VETH pair 2 [veth4@if13] (ID-2)
VETH pair 2 [veth3@if12] (ID-2)
Network namespace
Linux inside (LXC) container
eth0
(LXC) container
Cgroup namespace
UTS namespace
Proces & PID namespace
User namespace
Mount namespace
CGROUPS (Limits): RAM/CPU/DISK/NET
Correlation in linux:
X@if12→Y@if13
cat /sys/class/net/veth3/if(index,link)
br0, vmbr0 or for OpenStack e.g. qbr65eeaf6e-7e
tap0 or for OpenStack e.g. tap65eeaf6e-7e
tap1
TAP [tap0]
Open vSwitch (OVS)
Or VETH
Or direct
Buffers
qvoxxx (ID1) VLAN Tag xy
br-int (OVS Bridge)
patch-tun (port)
patch-int (port)
br-tun (OVS Bridge)
VLAN xy (port) To: VM (NOVA) server Xy
VxLAN x (port) Tunnel to: VM (NOVA) server Xy
VxLAN y (port) Tunnel to: NET (Neutron) server Xy
br-int (visible directly from Linux)
patch-tun (internal patch interface)
patch-int (internal patch interface)
br-tun (visible directly from Linux)
VLAN-Xy (internal VLAN interface)
vxlan-c0a8c864 (internal VxLAN interface)
vxlan-Xy (interno VxLAN sučelje)
VxLAN Tunnels
VLAN or VxLAN
OpenStack Implementation
Network card (e.g. Intel® 82571)
TCP/UDP/IP Checksum
LRO
TSO
RSS
aRFS
VLAN, VxLAN, GRE, ESP, ...
Ethernet Checksum
RX
TX
Sysctl varijables
r = receive (RX)
w = write (TX)
net.core.rmem_max
net.core.rmem_default
net.core.wmem_max
net.core.wmem_default
SKB (sk_buff) Socket buffer
TCP
UDP
net.ipv4.tcp_mem
net.ipv4.tcp_rmem
net.ipv4.tcp_wmem
net.ipv4.udp_mem
net.ipv4.udp_rmem_min
net.ipv4.udp_wmem_min
Scaling TCP window (ON/OFF) is set in: net.ipv4.tcp_window_scaling
Selected algorithm for congestion control set in: net.ipv4.tcp_congestion_control
Timers are defined in:
FIN_WAIT_2 : net.ipv4.tcp_fin_timeout
TIME_WAIT= 2 x FIN_WAIT_2
Reusing connection in Time_wait state: net.ipv4.tcp_tw_reuse
Limiting the overall number of TCP connections in state opening: net.core.somaxconn
TCP Syn queue memory (Max number of connection which are in state of opening for specific port): net.ipv4.tcp_max_syn_backlog
Network statistics on this level are visible in: /proc/softirqs, /proc/net/softnet_stat, /proc/net/dev and /proc/net/netstat and /proc/net/snmp
Also visible with the next commands: netstat -i, netstat -s and ss and also with command: nstat
Range of ports available for use for all network connections (per IP): net.ipv4.ip_local_port_range
Enable routing between network interfaces on computer: net.ipv4.ip_forward
Enbale usage of non local IP address (e.g. for VRRP protocol): net.ipv4.ip_nonlocal_bind , . . .
Other functionality on this layer (entirely software or partially in software) :
Receive Packet Steering (RPS) - for check or change: /sys/class/net/eth0/queues/
Transmit Packet Steering (XPS) - for check or change (number of queue is XX): /sys/class/net/eth0/queues/tx-XX/xps_cpus
Generic receive offload (GRO) - for enabling: ethtool -K eth0 gro on
Generic segmentation offload (GSO) - for enabling: ethtool -K eth0 gso on
The selected scheduler for all network interfaces are set in: net.core.default_qdisc
For list of all available schedulers (e.g. codel,pfifo fast,fq,...) start the command: grep '^CONFIG_NET_SCH' /boot/config-$(uname -r)
For configuring on specific network interface; e.g. eth0 for selecting fq algorithm (scheduler): tc qdisc add dev eth0 root fq
Memory buffer for all options which are saved beside network packet (e.g. net interface, headers, errors, ...) : net.core.optmem_max
(RX) Buffer common for all network interfaces on the system: net.core.netdev_max_backlog
Netfilter subsystem for filtering packets (ebtables, arptables, iptables, nf_tables, conntrack, logging, ...)
Bridge component
VLAN, VxLAN, VETH, MACVLAN, ...
Bonding/teaming component (e.g. LACP)
ifenslave bond0 eth0 eth1
Statistics are visible in: /proc/net/bonding/bondXY
TXquelen (FIFO) buffer
Possible to change size (XX):
ip link set eth0 txqueuelen XX
NAPI Pooling
Number of packets that can be buffered before NAPI retrieval. If we have LRO and GRO turned on then it refers to aggregated/merged packets: net.core.dev_weight
The total number of packets that can be fetched at once, common to all network interfaces on the system: net.core.netdev_budget
Time in µs allowed for retrieving (pooling) packets: net.core.netdev_budget_usecs
For network statistics on this level (for eth0), we can use commands like: ip -s link and sar -n EDEV or ethtool -S eth0 as the lowest possible layer.
Possible to watch statistics directly from files: /proc/net/dev or /proc/net/netstat or /proc/net/snmp
It is possible to increase/decrease this buffer up to a certain hardware limit (for RX=YY and for TX=ZZ) with the following command:
ethtool -G eth0 rx YY tx ZZ
Hardware accelerated functionality on network card (e.g. eth0), (if exist):
Receive Side Scaling (RSS) - increasing RX/TX queue (multiqueue) on XX/YY: ethtool -L eth0 rx XX tx YY
Hardware Receive Flow Steering (aRFS) - enabling: ethtool -K eth0 ntuple on
Large receive offload (LRO) - enabling: ethtool -K eth0 lro on
TCP segmentation offload (TSO) - enabling: ethtool -K eth0 tso on
Checksum offload - enabling for RX and for TX: ethtool -K eth0 rx on tx on
Scatter-gather - enabling : ethtool -K eth0 sg on
Hardware support for VLAN (802.1Q) for TX and RX : ethtool -K eth0 rxvlan on txvlan on
Network card settings, such as speed and duplex mode; ex. 1000Mbps (1Gbps), full duplex:
ethtool -s eth0 speed 1000 duplex full
Interrupt signal (IRQ) optimizations according to CPU affinity (for IRQ number XY) are adjusted in::
/proc/irq/XY/smp_affinity
Physical network interfaces (e.g. eth0) are with TAP, VETH and other logical interfaces connected through the Linux bridge (pr. br0,vmbr) or OVS bridge interface
VLANs achieve network isolation, while VxLAN tunnels (OVS) typically connect servers to each other, according to the standard OpenStack implementation.
Physical and logical Linux network interfaces can be connected to the Linux firewall because they pass through the Netfilter component anyway
MTU size exists and can be changed on every network interface on Linux!
tap1 Name of TAP network interface on Linux
tapXYZ Name of OpenStack TAP network interface in Linux
GPL license
Hrvoje Horvat
v.1.20