NFV à la VDE way

Renzo Davoli

FOSDEM 2018
Virtual Distributed Ethernet

- A virtual networking standard since 2004
- Supported by Qemu/KVM, VirtualBOX, (user-mode-linux), PicoTCP, LWIPv6
- VDE4 features:
  - Modular design
  - Networking implementation plugins
  - Networks of Namespaces
VDEPLUG4 Modular Design

VDE compatible program

LIBVDEPLUG4

VDE PLUGIN

VDE compatible program

LIBVDEPLUG4

VDE PLUGIN
It is like a virtual SFP port

- Small form-factor pluggable (SFP) socket
- VDE plugins = SFP virtual tranceivers
Libvdeplug4

- Backwards compatible with libvdeplug2 (for applications: virtual machines, user-mode stacks)
- It provides an unified API for virtual networking specification
- e.g. kvm:
  
  ```
  kvm ... -netdev vde,id=vde0,sock=tap://tap0
  kvm ... -netdev vde,id=vde0,sock=vxvde://
  kvm ... -netdev vde,id=vde0,sock=slirp://
  ```
UVDEL: new URL-like network locators

- UVDEL = Unified VDE Locator
- Plugins are shared libs. The library `libvdeplug_foo.so` defines a new type of UVDELs `foo://...`
- Examples:
  - null://
  - vde://var/run/myswitch
  - tap://tap0
  - vxvde://234.0.0.1
  - slirp://
Some vde plugins:
**vde_plug:**

**the vde cable builder**

- vde_plug command syntax supports either one or two UVDELS.
  - In case of one UVDEL a vde_plug translates the network traffic in a stream on stdin/stdout
vde_plug

vde_plug foo:// bar://
dpipe vde_plug foo:// = vde_plug bar://
Example: 

tap + switch + 2*kvm

```bash
sudo ip tuntap add tapx mode tap user renzo
vdePlug tap://tapx switch:///tmp/swx
kvm ... -netdev vde,id=vde0,sock=vde:///tmp/swx
kvm ... -netdev vde,id=vde0,sock=vde:///tmp/swx
```
More vde plugins
Example:

\texttt{kvm + cmd + remote slirp}

\texttt{kvm ... -netdev vde,id=vde0,\}
\texttt{sock=cmd://’ssh remote.host vdeplug slirp://’}
VXVDE & Local Area Clouds

- Just use vxvde://mcastaddr.

Apps of the same mcastaddr flock together
VXVDEX

- VDEplug plugin library + Kernel module
- VXVDEX inherits all the pros of VXVDE but:
  - VXVDEX provides access control and “network privacy”. A sysadm can define which virtual networks a user can join or not.
    - The current implementation uses the effective group id as the VXVDEX net identifier. (see getegid(2))
    - A user can join a virtual network only if she is a member of the corresponding group.
- Users can have full shell access.
Nested VDEPLUGIN
(to be released soon)

- Some plug-ins are designed as wrappers for other modules
  - agno: agnostic encryption
  - vlan: add/del/change 802.1Q tags
- e.g.
  
  agno://vde:///tmp/myswitch
  vlan://2{tap://mytap}
## Security Table

<table>
<thead>
<tr>
<th></th>
<th>Virtual Machine user</th>
<th>Shell access</th>
<th>Access to the net cable</th>
</tr>
</thead>
<tbody>
<tr>
<td>VXVDE</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>VXVDEX</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Encryption + VXVDE</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
VDENS: life in a networking namespace

- VDENS creates a networking namespace and connects it to a VDE network.
Example:

vdens + ptp + kvm

kvm ... -netdev vde,id=vde0,sock=ptp:///tmp/xxx
vdens ptp:///tmp/xxx
VDENS usage cases

• Server side:
  – IoTh Virtual hosting

• Client side
  – Different security requirements
    • VPN and local services

• IoTh in a backwards compatible way
Networks of Namespaces

VDE namespaces can be scattered around the Local Area Cloud (zero configuration!)
VDENS –multi

- A VDE namespace can have several virtual interfaces
- Standard bridging, routing, packet filtering and shaping methods can be implemented in the namespace
- It is possible to run Network Function virtualization tools for virtual networks (VNFV).
NFV à la VDE
Virtual Network Functions in VDENS

• Standard Linux programs and features can run in a VDENS (VNF)
  - Servers (apache, nginx, tftpd, ....)
  - DHCP servers (and clients)
  - Proxy servers
  - DNS servers
  - Iptables:
    • Packet filtering/shaping (iptables)
    • Load balancing
    • Firewall

• A network namespace is a light and safe choice...
A virtual demo: set up the gateway

# create a tap
$ sudo ip tuntap add name tapr mode tap user renzo

# create a bridge and add eth0 and tapr to it
$ sudo brctl addbr br0
$ sudo brctl addif br0 eth0
$ sudo brctl addif br0 tapr

# enable all the interfaces
$ sudo ip link set eth0 up
$ sudo ip link set br0 up
$ sudo ip link set tapr up

# set the ip addr of the bridge interface
$ sudo ip addr add 10.0.0.1/24 dev br0

# from now on everything is virtual and distributed
# no more need for sudo
# A virtual demo: set up a “router”

## create a multi-interface name space

```
$ vdens --multi tap://tapr vxvde://234.0.0.1 vxvde://234.0.0.2
```

## enable the virtual interfaces

```
vdens$ ip link set vde0 up
ddens$ ip link set vde1 up
ddens$ ip link set vde2 up
```

## set up the link to the gateway

```
vdens$ ip addr add 10.0.0.2/24 dev vde0
```

## now any tool working as a NVF can be applied here

```
# this example creates a NAT-masqueraded virtual neton vxvde://234.0.0.1
vdens$ echo "1" > /proc/sys/net/ipv4/ip_forward
vdens$ /sbin/iptables -t nat -A POSTROUTING -o vde0 -j MASQUERADE
vdens$ ip addr add 10.10.10.1/24 dev vde1
```

## let us check the configuration

```
vdens$ ip addr
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default qlen 1000
   link/loopback 00:00:00.00:00:00:00 brd 00:00:00:00:00:00
2: vde0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state...
   link/ether 1e:c2:7e:cf:89:60 brd ff:ff:ff:ff:ff:ff
   inet 10.0.0.2/24 scope global vde0
      valid_lft forever preferred_lft forever
   inet6 fe80::1cc2:7eff:fecf:8960/64 scope link
      valid_lft forever preferred_lft forever
3: vde1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state...
   link/ether 22:8e:2f:32:2f:32:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
   inet 10.10.10.1/24 scope global vde1
      valid_lft forever preferred_lft forever
   inet6 fe80::208e:2fff:fe2f:322f/64 scope link
      valid_lft forever preferred_lft forever
4: vde2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state...
   link/ether b6:9e:13:56:f9:cc brd ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
   inet6 fe80::b49e:13ff:fe56:f9cc/64 scope link
      valid_lft forever preferred_lft forever
```
A virtual demo:
set up a network node

# now on a random box of your LAN
$ vdens vxvde://234.0.0.1
# set up the interface
vdens$ ip link set vde0 up
vdens$ ip addr add 10.10.10.2/24 dev vde0
vdens$ ip route add default via 10.10.10.1
# run your favourite processes (maybe servers)
...

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License. (unless otherwise specified)
Degrees of Virtualization:
0 – no virtualization
Degrees of Virtualization: 1 – virtual machines
Degrees of Virtualization:
2 – NFV
Degrees of Virtualization: 3 – NFV + VDE + NoN
We are still creating art and beauty on a computer:

the art and beauty of revolutionary ideas translated into (libre) code...

renzo, rd235, iz4dje