vmd

Reyk Flöter
reyk@openbsd.org

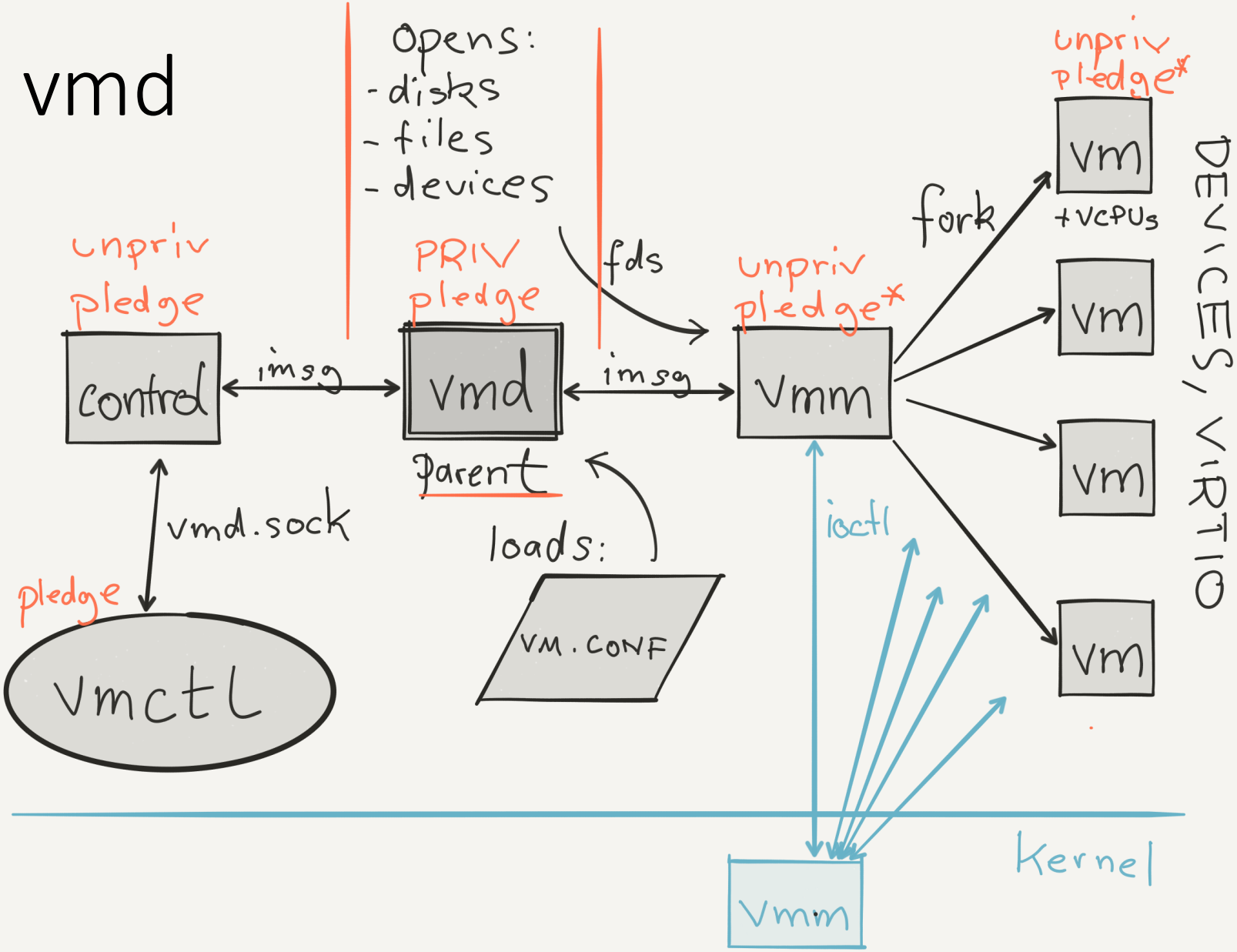
About vmd

- “vmd is a daemon responsible for the execution of virtual machines (VMs) on a host.”
  - vmd(8) interfaces with vmm(4) in the kernel
  - It handles the VM setup, vCPUs, exists, and device layer
  - vmd(8) and vmctl(8) manage the VMs
- We want to provide complete functionality in base
  - Ready to use, built and designed for OpenBSD
  - Focus on features that we need
  - An alternative device layer could be provided by qemu

History of vmd

- Mike Larkin wrote vmm(4) and the initial vmd(8)
  - vmd(8) was a simple but functional daemon
    - One parent process, the VMs, and a simple vmmctl tool
  - It included the implementation of a VIRTIO device layer
    - Disks
    - Network Interfaces
    - Virtual CPUs (VCPUs)
- I turned vmd(8) into an “OpenBSD-style” daemon:
  - Fully privilege-separated (privsep) process model
  - Well-defined configuration grammar (/etc/vm.conf)
  - Improved status and control tool (vmctl)

vmd


Opens:
- disks
- files
- devices
unpriv
pledge
PRIV
pledge
fds
unpriv
pledge*
unpriv
pledge*
fork
vm
+VCPUs
vm
vm
vm
DEVICES, VIRTIO
control
imsg
vmd
imsg
vmm
parent
vmd.sock
pledge
loads:
ioctl
VM.CONF
vmctl
Kernel
vmm

vmctl Control Tool

- vmctl is used to control and monitor vmd(8)
  - Advanced configuration is done via vm.conf
- It implements sub-commands with options
  - Unlike other ctls in OpenBSD, it does not use CLI-style
- Create a 4.5 Gigabyte disk image, disk.img:

```
# vmctl create disk –s 4.5G
```

- Create a new VM with 512MB memory:

```
# vmctl start “myvm” –m 512M –i 1 –d disk.img –k /bsd -c
```

- Terminate the VM ”myvm”:

```
# vmctl stop myvm
```

vm.conf Configuration File

- A well-defined and human-readable grammar
  - No need for “getopt hell” and shell scripts calling vmctl
  - Based on OpenBSD’s configuration parser, as used in
    - pf, bgpd, relayd, httpd, ospfd, snmpd, … and many others.
  - Supports macro variables, comments and includes
- vmd(8) loads the vm.conf on boot or reload

```
openbsd=“/bsd”
vm “myvm” {
    memory 512M
    interfaces 1
    disk “/var/vmm/myvm.img”
    # Use the default
    kernel $openbsd
}
```

vmm and the VM Processes

- “sandboxed” VMs using privsep and pledge
  - New pledge “vmm” restricts allowed ioctls to vmm(4)
- The vmm process communicates with the kernel
  - It forks and monitors the VM processes
  - It receives devices (disks, kernel, NICs) from vmd

```
if (pledge(“stdio vmm recvfd proc”) == -1)
    fatal(“pledge”);
```

- The VM processes represent each virtual machine:
  - Each process runs with multiple threads, one per VCPU
  - Handles exits and device I/O from vmm(4) in the kernel

```
if (pledge(“stdio vmm”) == -1)
    fatal(“pledge”);
```

Future Work in vmd

- I‘m waiting for Mike Larkin‘s interrupt controller
  - Networking will be much easier when it is ready
- Change the network “interfaces” configuration
  - Define virtual switches in vm.conf
  - Assign VMs to virtual switches
  - Integrate with upcoming work on switch(4) / switchd(8)
- Add support for VM templates and instances
- Support additional disk formats, eg. VMX export
  - Enable it, enable full pledge