Systemsoftware

Emulatoren (Qemu)

Prof. Dr. Michael Mächtel

Informatik, HTWG Konstanz

Version vom 26.03.17

Übersicht

Übersicht

Was ist QEMU?

- ‘Fast processor emulator’: Ein Emulator für einen Prozessor und Peripherie
- QEMU emuliert User-level Prozesse für die Target CPU auf der Host CPU
- QEMU emuliert komplette Hardwaresysteme (CPU+Peripherie)
- QEMU kann als ein Virtual Machine Monitor (Hypervisor) gesehen werden
- Quelle: http://qemu.org

Warum QEMU?

- Zum Testen von Systemsoftware
  - RootFS
  - Kernel
- Emuliert Prozessor + Peripherie, ohne Hardware
  - z.B. Verschiedene ARM Boards:
    - *qemu-system-arm -M ?*
- Zeitersparnis
  - Programmierung
  - Debugging

Übersicht

Qemu Tasten

Tastenkombination von QEMU	
Tasten	**Erklärung**
Strg + Alt	Maus aus dem QEMU-Fenster befreien
Strg + Alt + 2	vom Gast in den QEMU-Monitor wechseln
Strg + Alt + 1	vom QEMU-Monitor ins Gast-Betriebssystem wechseln
Strg + Alt + F	zwischen Fenster- und Vollbildmodus wechseln

Qemu Monitor

Befehle für den QEMU-Monitor	
Befehl	**Erklärung**
`info GERÄT`	gibt Infos über das virtuelle Gerät aus; mögliche Geräte sind u.a. `block` (Festplatte[n], CD-ROM), `snapshot`, `usb` und `network`
`change GERÄT GERÄTEDATEI`	Tauscht ein Wechselmedium (CD/DVD, Diskette) aus (siehe: Wechseldatenträger)
`commit`	schreibt einen Snapshot, sofern QEMU mit der Option `-snapshot` gestartet wurde
`screendump DATEI`	erstellt ein Bildschirmfoto, wobei das recht ungewöhnliche Dateiformat **ppm** verwendet wird. Beispiel: `screendump BILDNAME.ppm`
`help [befehl]`	zeigt eine Hilfe für alle Befehle oder nur für den Befehl `befehl`

Reine Prozessemulation

- QEMU beherrscht auch die “reine” Prozessemulation, auch “User-Space-Emulation” genannt.
- D.h. dass anstatt eines kompletten Systems wird “nur” ein einzelnes Programm (“Binary”) im Emulations-Modus ausgeführt.
- Die Prozess-Emulation für ein 32-bit i386 System wird z.B. mit folgendem Befehl aufgerufen:
  - qemu-i386 PROGRAMMNAME
- Die Emulation funktioniert natürlich nur, wenn das Programm keine weiteren Bibliotheken dynamisch nachlädt.
- Außer der i386-Emulation beherrscht QEMU u.a. auch die Prozessemulation für
  - SPARC, PPC, ARM und einige mehr.

Andere Architekturen emulieren

- QEMU ist nicht auf die Virtualisierung /Emulation von x86 Prozessoren beschränkt,
- es können auch eine Vielzahl von anderen Architekturen emuliert werden.
- Die allgemeine Syntax ist
  - **qemu-system-ARCHITEKTUR [OPTIONEN]**
  - wobei ARCHITEKTUR entsprechend ersetzt werden muss.

Systemparameter

- *-kernel Image*
  - Linux Kernel Image zu booten
- *-initrd Image*
  - initrd /initramfs Image
- *-append <“...”>*
  - Um dem Kernel Parameter zu übergeben
  - z.B. *-append “rdinit=/init”*
- *-hda Platte.img*
  - Bietet Platte.img als hda Mount
- *-cdrom cd.iso*
  - Biete cd.iso als CDROM Image

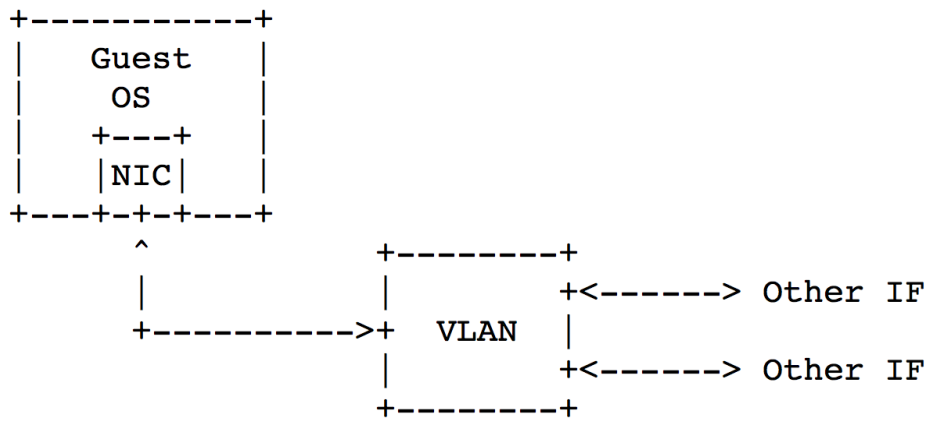
Serial /Monitor

- *-serial [stdio,file,tcp,...]*
  - Serielles Device vorgeben
- *-monitor dev*
  - dev siehe -serial !

VLAN

- Ein VLAN ist ein Network-Switch der im Kontext des QEMU Prozesses läuft
- Mehrere VLANs können pro QEMU Prozess definiert werden.
- Mehrere Interfaces können mit einem VLAN verbunden werden
- Werden Daten von einem Interface über das VLAN verschickt, gelangen die Daten an alle Interfaces, die mit diesem VLAN verbunden sind.

VLAN und NIC

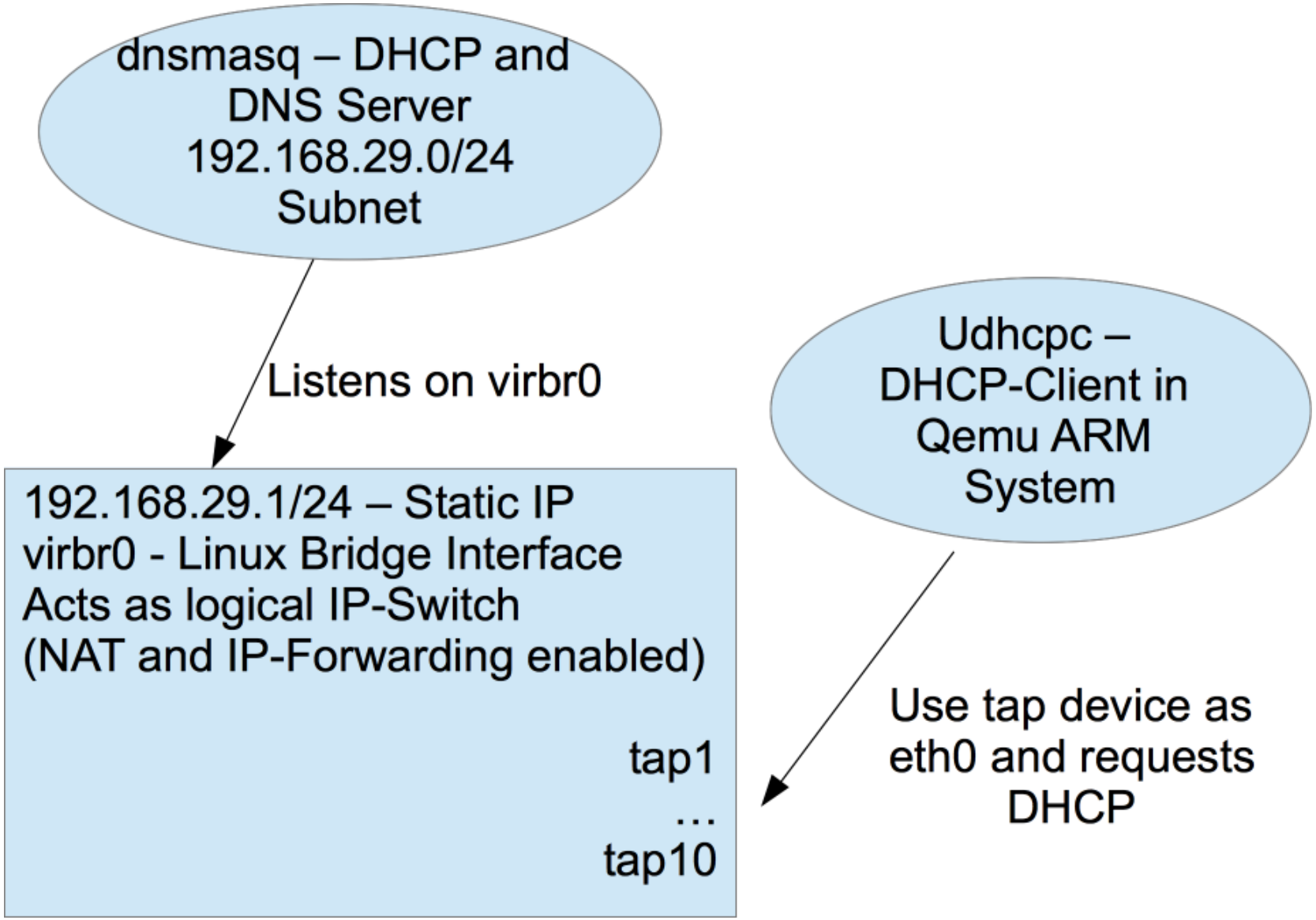


So, for example, if you do:

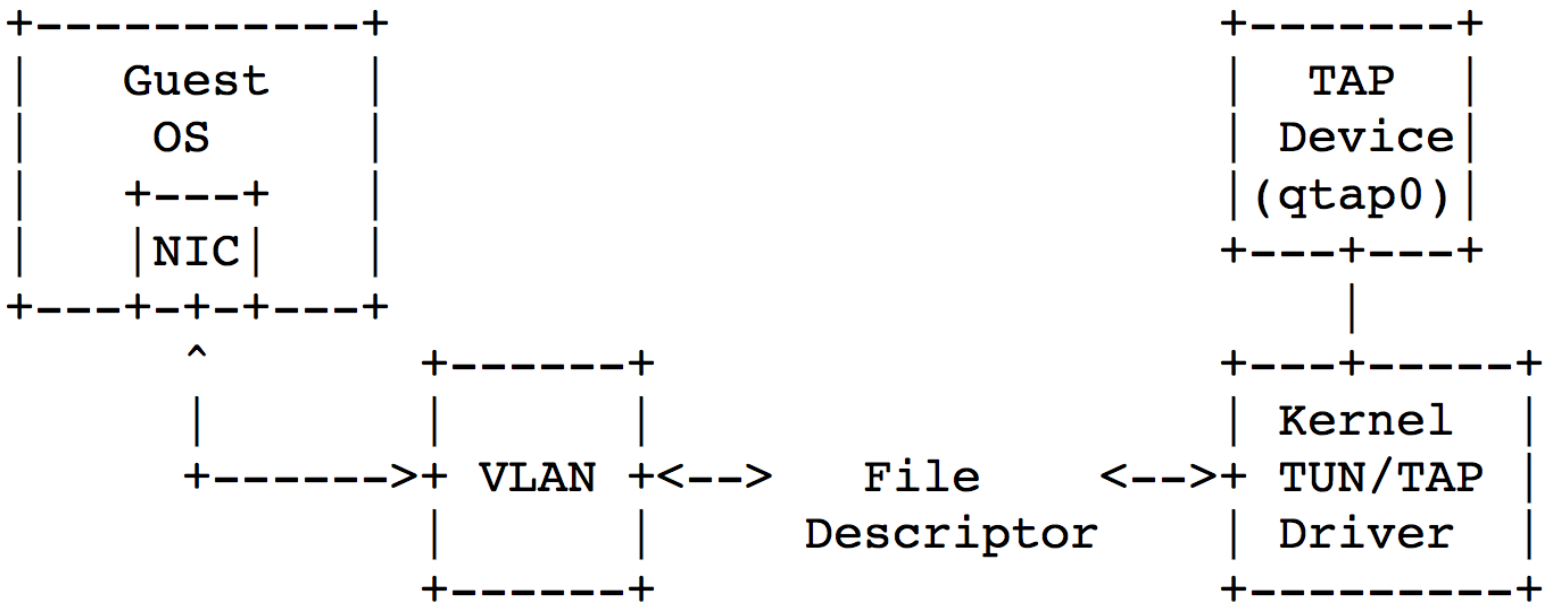
```
$> qemu -net nic,model=rtl8139,vlan=1,macaddr=52:54:00:12:34:56 ...
```

It will create a rtl8139 nic in the guest, with the given MAC address, and connect to vlan numbered 1. You can then use further -net options to connect other interfaces to that vlan.

Netzwerk: homer<->qemu



VLAN und TUN/TAP Device




```
$> qemu -net nic -net tap,ifname=qtap0 ...
```

Übersicht

gdb

- GDB kann auf dem Host-System benutzt werden, um den Kernel und Userprogramme, die in QEMU laufen, zu debuggen.
- Dazu hat QEMU selbst einen gdbserver eingebaut, mit welchem sich GDB verbinden kann.
- Aufruf: **qemu -hda disk.image -S -gdb tcp::12345**
  - Die Option *-S* friert die CPU(s) beim Start der Instanz ein
  - *-gdb* startet den gdbserver von qemu auf TCP Port 12345

debug kernel

- Um den Linux Kernel zu debuggen:
  - **gdb** auf dem Host z.B. mit dem Kommando **gdb vmlinux** starten
    - **gdb-multiarch**, wenn das Target eine andere Architektur hat
    - *Vmlinux* ist das pre-build Linux image mit debug Informationen
  - gdb debuggt somit vmlinux und liest die Symbole aus dem Image
  - Nach dem Start von gdb wird über die Anweisung “target remote localhost:12345” gdb mit qemu verbunden
    - Kommando “c”: to continue the original program
    - Kommando “b some_symbol_name” to set break points at symbols in the program
    - Kommando “p var_name” to print variable values

kgdb

- Kgdb ist ein Kernel Patch.
- Mit kgdb kann der kernel im laufenden System debuggt werden:
  - Der kgdb Kernel Patch fügt dem Kernel einen gdb Stub hinzu sowie
  - eine serielle ‘Verbindung’
- Da kgdb ein extra Kernel Patch ist, muss dieser von Version zu Version angepasst werden.
- Es gibt auch eine zu bezahlende kgdb-pro Version, die aktuelle Kernel direkt unterstützt
- Quelle: kgdb.linsyssoft.com/getting.htm

Übersicht

Overlay

- QEMU bietet die Möglichkeit, mit Overlay-Dateien (übersetzt: “Überlagerungsdateien”) zu arbeiten.
- Dabei werden neue und geänderte Daten nicht in die Original Imagedatei sondern in die Overlay-Datei geschrieben.
- Das Original Image bleibt unverändert.
  - Diese ist z.B. dann praktisch, wenn man eine “saubere” Grundinstallation eines Systems hat und “Experimente” mit dem Overlay-Image gemacht werden, welche sich nicht auf das Original auswirken.
  - Ebenso kann man mehrere Overlays für ein Image anlegen.

Overlay anlegen

- *qemu-img create -b mein_image.img -f qcow2 mein_overlay.ovl*
  - Der Name der Overlay-Datei kann dabei beliebig sein und muss nichts mit den Namen des Images zu tun haben.
  - Danach kann man die virtuelle Maschine normal booten, nur dass anstatt des Images die Overlay-Datei als Festplatte angegeben werden muss, also z.B.
- *qemu -hda mein_overlay.ovl*
  - Die Overlay-Dateien sind “Delta-Images”, d.h. in der Overlay-Datei werden Änderungen relativ zum Original Image gespeichert.
  - Ändert man das zugrunde liegende Image, nachdem eine Overlay-Datei angelegt wurde, funktioniert die Overlay-Datei nicht mehr!