Advanced networking with android

Preface

${android_sdk} refers to your android sdk root folder

Your emulator consists of three images:

- ${android_sdk}/tools/lib/images/ramdisk.img (cpio[cnew] archive, gzip compressed)
- ${android_sdk}/tools/lib/images/system.img (yaffs2 formated)
- ${android_sdk}/tools/lib/images/userdata.img (yaffs2 formated)

By now noone has managed to mounting the yaffs2 formated partitions. Loopback mounting does not work. So for now the best way to hack the emulator is ${android_sdk}/tools/lib/images/ramdisk.img

Step 0 – backup

cp ${android_sdk}/tools/lib/images/ramdisk.img ${android_sdk}/tools/lib/images/ramdisk.img.old

Step 1 - extracting the ramdisk

mkdir ramdisk
cd ramdisk
cat ${android_sdk}/tools/lib/images/ramdisk.img | gunzip | sudo cpio –i

Step 2 - looking at a simple boot skript

Android requires some special code to boot in the emulator, the code can be found in etc/qemu-init.sh
sudo vim etc/qemu-init.sh
Here is the original content:
#!/system/bin/sh

```
# Some special case stuff for running under emulation
qemu=`getprop ro.kernel.qemu`
case "$qemu" in
    "1" )
    ifconfig eth0 10.0.2.15 netmask 255.255.255.0 up
    route add default gw 10.0.2.2 dev eth0
    radio_ril=`getprop ro.kernel.android.ril`
    case "$radio_ril" in
```

```
        ttyS*) # a modem is emulated as a serial device
            ;;
        *) # no need for the radio interface daemon
            # telephony is entirely emulated in Java
            setprop ro.radio.noril yes
            stop ril-daemon
            ;;
    esac
    setprop net.eth0.dns1 10.0.2.3
    setprop net.gprs.local-ip 10.0.2.15
    setprop ro.radio.use-ppp no
    setprop ro.config.nocheckin yes
    setprop status.battery.state Slow
    setprop status.battery.level 5
    setprop status.battery.level_raw  50
    setprop status.battery.level_scale 9
    stop dund
    stop usbd
    ;;
esac
```


Step 3 - the target networking environment

Android is hardcoded to use 10.0.2.3 and 10.0.2.2 for comunication - especially with adb. You can not use anything except this userspace emulation for eth0.

Android

- device: eth0, ip: 10.0.2.15, net: 10.0.2.0/255.255.255.0 - this will keep adb happy
- device: eth1, ip: 192.168.1.2, net: 192.168.1.0/255.255.255.0 - this will be used for real communication
- default gateway: 192.168.1.1 - or whatever suits your network
- DNS: 10.0.2.3 - we simply keep this, no disadvantages except harder debugging

Host

- net_android, virtual android network device
- br0, network bridge linking eth0 and net_android

Depending on what you want to test you might want to turn your host system into a nat gateway, a dhcp server, transparent proxy, routing gateway or whatever you need.

Step 4 - altering the android network

```
#!/system/bin/sh

# Some special case stuff for running under emulation
qemu=`getprop ro.kernel.qemu`
case "$qemu" in
    "1" )

    if ifconfig eth1 up; then
        ifconfig eth1 down
        # Should we really check for eth1 that way?

        # DHCP - broken?
        # netcfg eth1 dhcp

        ifconfig eth1 192.168.1.2 netmask 255.255.255.0 up
        route add default gw 192.168.1.1 dev eth1

        ifconfig eth0 10.0.2.15 netmask 255.255.255.0 up

        setprop net.eth0.dns1 10.0.2.3
        setprop net.eth1.dns1 10.0.2.3
        setprop net.gprs.local-ip 192.168.1.2
    else
        ifconfig eth0 10.0.2.15 netmask 255.255.255.0 up
        route add default gw 10.0.2.2 dev eth0
        setprop net.eth0.dns1 10.0.2.3
        setprop net.gprs.local-ip 10.0.2.15
    fi

    # Proxy support - broken too?
    # setprop net.gprs.http-proxy http://proxy:8080/

    # No longer needed
    # ifconfig eth0 10.0.2.15 netmask 255.255.255.0 up
    # route add default gw 10.0.2.2 dev eth0
    radio_ril=`getprop ro.kernel.android.ril`
    case "$radio_ril" in
        ttyS*) # a modem is emulated as a serial device
            ;;
        *) # no need for the radio interface daemon
            # telephony is entirely emulated in Java
            setprop ro.radio.noril yes
            stop ril-daemon
            ;;
```

```
    esac
    # no longer needed
    # setprop net.eth0.dns1 10.0.2.3
    # setprop net.gprs.local-ip 10.0.2.15
    setprop ro.radio.use-ppp no
    setprop ro.config.nocheckin yes
    setprop status.battery.state Slow
# Interresting - is this the way to check battery status?
    setprop status.battery.level 5
    setprop status.battery.level_raw   50
    setprop status.battery.level_scale 9
    stop dund
    stop usbd
    ;;
esac
```

This will keep your classic config if eth1 doesn't exist and if it exists adb will still work.

Step 5 - packing your new ramdisk

```
sudo find | sudo cpio -o -H newc | gzip -9 >
${android_sdk}/tools/lib/images/ramdisk.img
```


Step 6 - Host configuration

```
sudo brctl addbr br0
sudo tunctl -u $USER -t net_android
sudo ifconfig eth0 0.0.0.0 promisc up
sudo ifconfig net_android 0.0.0.0 promisc up
sudo brctl addif br0 eth0
sudo brctl addif br0 net_android
sudo dhclient3 br0
```

Depending on you sodu configuration you might need to substitute $USER with your username. Depending on your local network you might need to substitute sudo dhclient3 br0 with your normal network initialization.

Step 7 - boot and test

```
${android_sdk}/tools/emulator -debug-kernel -qemu -net nic -net user -net nic -net
tap,ifname=net_android
```

Now on your host system do

```
ping 192.168.2.2
adb shell
```

If everything went well both commands will work.

Enjoy!

TODO1: Tutorial how to use the new network link from within dalvik.
TODO2: Find a windows user who will try the ramdisk with windows and openvpn/tun/tap driver