How to build your own home linux router

Replacing plastic routers with your own software

Rouven Czerwinski – r.czerwinski@pengutronix.de

About me

Rouven Czerwinski

Pengutronix e.K.

Emantor

rcz@pengutronix.de

Embedded Linux Systems

IoT Security

Kernel Development

Why?

- Replace plastic router with your own known software
- Allows additional metrics collection (temperature dropped packets, logs,...)
- Regular software updates
- Learn how routing, firewalling, DHCP and other services work and are configured

Overview

- NixOS
- Hardware
- Services
- Firewalling
- VPN

Disclaimer!

- This is not a NixOS presentation
- NixOS is great but is another topic on its own
- This presentation contains generic configuration
- Author has a youtube video about NixOS on youtube: https://www.youtube.com/watch?v=dnAtcJu4Bgk&t=458s
- Nevertheless, we'll look at some NixOS basics

NixOS

- Linux OS based on nix programming language
- Idea: declaratively describe the system
- Packages described in nix lang
- Modules for individual services (DHCP, unbound DNS server, icecream,...)
- Allows sharing declarative system state across different machines

Nix Logo – CC-BY 4.0 @timbertson

NixOS: package example

```
 stdenv.mkDerivation rec {
  pname = "hello";
  version = "2.10";

  src = fetchurl {
    url = "mirror://gnu/hello/${pname}-${version}.tar.gz";
    sha256 = "0ssi1wpaf7plaswqqjwigppsg5fyh99vdlb9kzl7c9lng89ndq1i";
  };

  doCheck = true;

  passthru.tests.version =
    testVersion { package = hello; };

  meta = with lib; {
    […]
  };
}
```

NixOS: module usage

Enable unfree packages
Enable 32-bit support for pulseaudio
Enable 32-bit 3D acceleration support
Cache 32bit Fonts
Enable steam

```
nixpkgs.config.allowUnfree = true;
hardware.pulseaudio.support32Bit = true;
hardware.opengl.driSupport32Bit = true;
fonts.fontconfig.cache32Bit = true;
programs.steam.enable = true;
```

Define fqdn variable
Enable gitea
Set domain to fqdn
Only listen on localhost
Loglevel warning
Disable registration
Setup url to https

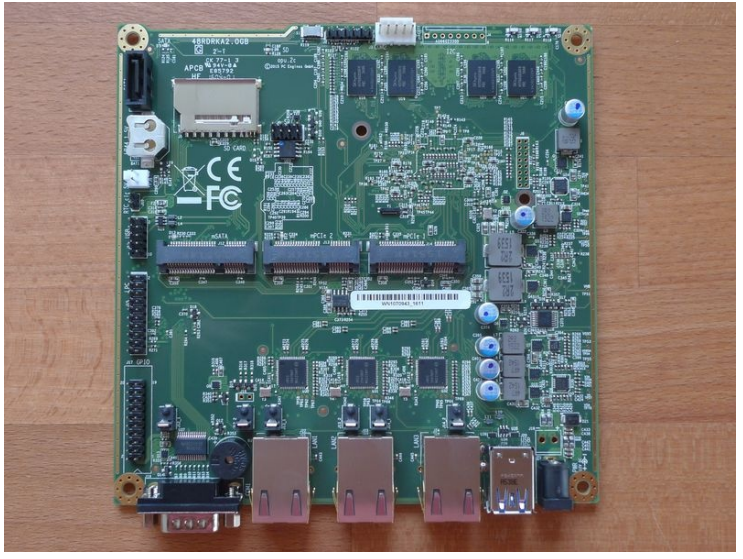
```
let
  fqdn = "gitea.discworld.emantor.de";
in {
  services.gitea.enable = true;
  services.gitea.domain = fqdn;
  services.gitea.httpAddress = "127.0.0.1";
  services.gitea.log.level = "Warn";
  services.gitea.disableRegistration = true;
  services.gitea.rootUrl = "https://${fqdn}";
}
```

NixOS: how to try

- Setup a Virtual machine
  - Try a few services
  - Package a new package
  - Migrate existing configuration
- Install nix into your distribution
  - Package your own project
  - Play with home manager to manage your configuration files
- Or try it on a spare laptop

Hardware: Computer

- APU 2C4
- 4GB RAM
- Embedded AMD 4-core CPU
- 3x Intel NICs
- fanless



APU2C4 – from https://www.pcengines.ch/apu2c4.htm

Hardware: VDSL Modem

- Zyxel DSL Router (VMG1312B)
- Can function as a PPPoE Modem
- Any PPPoE capable modem should work
- Rest of the talk assumes VDSL
- Fritz Box does not work to my knowledge!

Services: Overview

- DHCP: IP-address assignment
- DNS: name resolution (internal/external)
- PPPoE: internet access
- Firewall Setup
- Dynamic DNS
- Ipv6: prefix delegation

DHCP: ISC DHCP

- Dynamic Host Configuration Protocol
- Distribute IPs for devices
- Assign IPs based on MAC addresses
- Alternatives: KEA DHCP server, systemd-networkd DHCP functionality

```
default-lease-time 600;
max-lease-time 7200;
authoritative;
ddns-update-style interim;
log-facility local1; # see dhcpd.nix

ddns-update-style none;
subnet 192.168.128.0 netmask 255.255.255.0 {
  range 192.168.128.100 192.168.128.200;
  option routers 192.168.128.254;
  option ntp-servers 192.168.128.254;
  option domain-name-servers 192.168.128.254;
  option domain-name "discworld.emantor.de";
}
```

DNS: unbound

- Internal DNS resolver
- Forward request to foward server
- Caches results

```
 server:
  access-control: 127.0.0.0/8 allow
  […]
  interface: 127.0.0.1
  […]
  ip-freebind: yes
  module-config: iterator
  tls-cert-bundle: /etc/ssl/certs/ca-
certificates.crt
  val-permissive-mode: yes
forward-zone:
  forward-addr: 8.8.8.8
  forward-addr: 8.8.4.4
  name: .
```

DNS: stub zones

- Disable default zone handling
- Redirect stub zones
  - Discworld.emantor.de to internal server
  - Reverse DNS domains to internal server as well

```
server:
  local-zone: "168.192.in-addr.arpa."
nodefault
  local-zone: "d.f.ip6.arpa." nodefault
stub-zone:
  name: discworld.emantor.de
  stub-addr: 192.168.128.220
stub-zone:
  name: 6.f.a.1.2.6.5.a.6.6.d.f.ip6.arpa
  stub-addr: 192.168.128.220
stub-zone:
  name: 168.192.in-addr.arpa
  stub-addr: 192.168.128.220
```

PPPoE: pppd

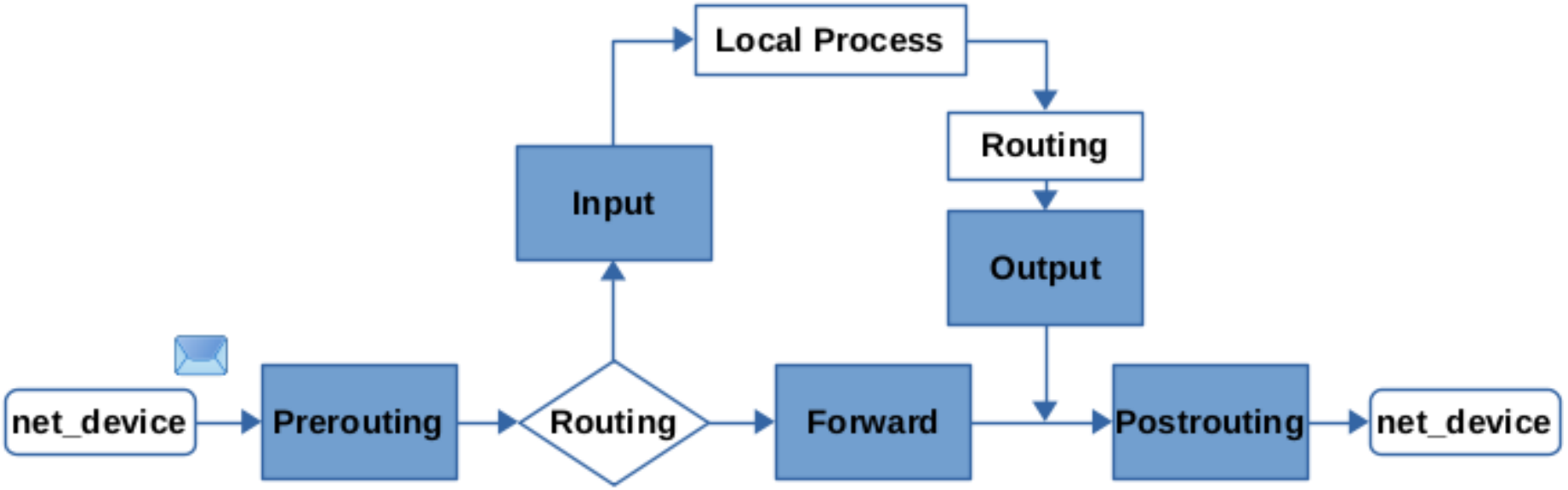
- Peer to Peer Protocol over Ethernet (PPPoE)
- Persist: do not exit after termination
- Maxfail 0: always retry
- Holdoff 5: wait 5 seconds before reconnect
- Noipdefault: dont determine local
- Defaultroute: setup default route

```
plugin rp-pppoe.so enp2s0

# login credentials in /etc/*-secrets
name "xxx@yyy.de"

persist
maxfail 0
holdoff 5

noipdefault
defaultroute
```

Firewall Setup: NFTables

- Use NFTables, successor to iptables
- Why? No separate IPv4 & IPv6 rules!
- Steps required:
  - Setup default policies
  - Allow access from specific interfaces
  - Don’t forget to setup conntrack rules to access the internet
  - Enable masquerading for IPv4

Firewall Setup: hook overview



Netfilter hooks CC BY SA 4.0 - from thermalcircle

Firewall Setup: default policies

- Chain output: accept (packets from the router)
- Chain input: drop (packets to the router)
  - Allow internal networks later
- Chain forward: drop (packets forwarded from router)
  - Allow internal networks later
- Chain prerouting: accept (packets before routing decision)
- Chain postrouting: accept (packets after routing)
  - Add masquerade rule for ppp0

Firewall: internet access

- enp1s0: internal
- ppp0: internet
- enp2s0: modem network
- Allow all ICMP packets to the router
- Allow forwarded ICMP traffic

```
table inet filter {
  chain input {
    ct state { established, related } accept
  }
  chain forward {
    iifname "ppp0" oif "enp1s0" ct state
established,related accept
    iifname "ppp0" oifname "wg-roadwarrior"
ct state established,related accept
    iif "enp2s0" accept
  }
}
table ip nat {
  chain postrouting {
    oifname "ppp0" masquerade
  }
}
```

Firewall Setup: debugging

- Tcpdump is your friend
- Run dump on input and output interface
- Remember if packets are for router or internal network
- Use logging mechanism to catch packets

Dynamic DNS: ddclient

- Dynamic DNS client
- Useful to provide outside access to your network via VPN
- Uses a hackerspace service here, others available
- Can also be self hosted

```
  services.ddclient = {
    enable = true;
    use = "web,
web=http://ipv4.stratum0.net/myip";
    ssl = true;
    server = "ipv4.stratum0.net";
    username = "discworld.stratum0.net";
    domains = [ "discworld.stratum0.net"
];
    passwordFile = "/var/keys/ddns-
stratum0";
  };
```

IPv6: firewall setup

- Enable Source Port 547 UDP receive for DHCPv6
- Allow all ICMP packets to the router
- Allow forwarded ICMP traffic

```
table inet filter {
  chain input {
    meta nfproto ipv6 udp sport 547 accept
    ip6 nexthdr ipv6-icmp accept
  }
  chain forward {
    meta l4proto ipv6-icmp accept
  }
}
```

Ipv6: systemd-networkd

- IPv6 prefix delegation for global addresses (internet access)
  - Distributed by provider
  - German provider: new prefix on reconnect
  - Also german provider: 24h reconnect
- IPv6 Prefix is usually a /48, leaving 16 Bits = 65536 Subnets for you
- German providers use DHCPv6 to assign prefixes
- Plan: fetch prefix from PPPoE interface, assign delegated prefixes to other interfaces

Ipv6: PPPoE interface config

- Match ppp interface
- Enable DHCPv6
- Create default route for dev
- Keep configuration from pppd
- Enable link local addressing for IPv6
- Request a /56 prefix
- Setup CAKE for 40MBit/s upload (VDSL 100/40)

```
[Match]
Name=ppp*

[Network]
DHCP=ipv6
DefaultRouteOnDevice=true
KeepConfiguration=static
LinkLocalAddressing=ipv6

[DHCPv6]
ForceDHCPv6PDOtherInformation=true
PrefixDelegationHint=::/56

[CAKE]
OverheadBytes = 65
Bandwidth = 40M
```

Ipv6: internal interface config

- Match enp1s0 interface
- Setup static address
- Setup domain
- Enable LLDP (convenience)
- Send Ipv6 Router ADvertisements
- Set SubnetId to 0x1 (enables PD)
- Announce an additional internal prefix

```
[Match]
Name=enp1s0

[Network]
Address=192.168.128.254/24
DHCPv6PrefixDelegation=true
Domains=discworld.emantor.de
EmitLLDP=yes
IPv6SendRA=true

[DHCPv6PrefixDelegation]
SubnetId=0x1

[IPv6Prefix]
Prefix=fd66:a562:1af6:1::/64
```

Linux Kernel configuration

- Enable forwarding for Ipv4/6
- Disable kernel IPv6 handling (will be done by systemd-networkd)

```
net.ipv4.conf.all.forwarding=1
net.ipv6.conf.all.forwarding=1
```

```
net.ipv6.conf.all.accept_ra=0
net.ipv6.conf.all.autoconf=0
net.ipv6.conf.all.disable_ipv6=0
net.ipv6.conf.all.use_tempaddr=0
net.ipv6.conf.default.disable_ipv6=0
net.ipv6.conf.default.use_tempaddr=2
```

VPN: Wireguard

- Requires a DDNS setup (unless you have a fixed IP)
- Public Key based scheme, available for linux, android, iOS, windows, openBSD, macOS,...
- Author has tested:
  - Linux ModemManager integration (on Laptop)
  - iOS integration (on iPad)
  - Android integration (on smartphone)

VPN: Wireguard setup

- Collect public keys from devices
- Assign IPs
- Public key transfer to smartphone/tablet by QR code
- Setup DNS to use router IP

```
[interface]
Address = 192.168.129.254/24

ListenPort = 51820
PostUp =
/nix/store/1lbj91n2182cv81fr2lmrvk7w9fckwis-
postUp.sh/bin/postUp.sh
[Peer]
PublicKey =
FiwOiRnnMNlTf8nSlIqdzZXWBvbx8WIVAvbi7b/HHE0=
AllowedIPs = 192.168.129.12/32

[Peer]
PublicKey =
OoaatUQ37OI1HnKNYavgDcp84fF5Ze3k1Q2ELsoqrA0=
AllowedIPs = 192.168.129.11/32

[Peer]
PublicKey =
Zd0Y+IeApNqd0j22XrYWDAbEwfxObqnNxFDFBNbs70Q=
AllowedIPs = 192.168.129.10/32
```

VPN: Firewall setup

- Wg-roadwarrior: VPN interface
- Wireguard listen port is 51820
- Allow all traffic forwarding from VPN clients
- Allow traffic back from the internet
- VPN only traffic is a client side configuration

```
table inet filter {
  chain input {
    iifname "wg-roadwarrior" accept
    meta nfproto ipv4 udp dport 51820 accept
  }
  chain forward {
    iifname "wg-roadwarrior" accept
    iifname "ppp0" oifname "wg-roadwarrior"
ct state established,related accept
  }
}
```

TODO

- Enable IPv6 for all services (DNS, chronyd,…)
- Enable IPv6 VPN services
  - Assign IPs, check routing on all platforms
- Ensure offline internal network availability
  - There is some dependency to have internet access
- Recheck and simplify firewall configuration
- Use systemd-networkd for DHCP

Questions?

Answers!

Thank you

Thank you very much, have a nice sunday and CLT!

Links and Licenses

- Nix Logo: https://github.com/NixOS/nixos-artwork/tree/master/logo & https://creativecommons.org/licenses/by/4.0/
- APU2C4: APU2C4 – from https://www.pcengines.ch/apu2c4.htm
- Netfilter hooks CC BY SA 4.0 - from https://thermalcircle.de/doku.php?id=blog:linux:nftables_packet_flow_netfilter_hooks_detail
- Nixos router configuration: https://gist.github.com/Emantor/4f43646b8ada633455b902bc6f4c4d20