OpenWrt/LEDE: when two become one

Florian Fainelli
About Florian

• 2004: Bought a Linksys WRT54G
• 2006: Became an OpenWrt developer
• 2013: Joined Broadcom to work on Set-top Box and Cable Modem Linux kernel, toolchain, bootloader, root filesystem
• 2016: Joined the LEDE team...
• ... while remaining in OpenWrt
Summary

- Introduction to OpenWrt and LEDE
- Design, features and examples
- OpenWrt/LEDE reunification status
Introduction to OpenWrt/LEDE
What are OpenWrt and LEDE?

- Build systems
- Linux distributions
- Communities:
  - Wiki, forums, mailing-lists and git repositories
  - Users, contributors, developers
OpenWrt and LEDE in a nutshell

- Open source Software: Http, git, svn, files
- OpenWrt/LEDE User space components
- OpenWrt/LEDE
  - Makefile scripts
  - .config
  - Tools
- Kernel image
- Root filesystem
- Bootloader
- Packages
- Toolchain
- Firmware image(s)
- SDK
- Image builder
Design goals

• Maintainability
  – Working with latest technologies
  – Frequent updates to solve security flaws

• Ubiquity
  – Most off the shelf routers supported within weeks/months following public availability
  – With LEDE: extend scope beyond traditional network devices
  – Work with vendors to support OpenWrt/LEDE natively

• User empowerment
  – It’s open source!
  – Superior quality and control over vendor provided firmware

• Selected differentiation
  – Provide a state of the art network device experience
  – Turn-key solution to build real products
OpenWrt/LEDE in the landscape

- **Yocto/OE**: High Complexity, 1000+ Number of components/packages
- **OpenWrt/LEDE**: Medium Complexity, 100 Package feeds, 1 Core packages
- **buildroot**: Low Complexity, 1 Number of components/packages
A word or two about router security

• Home routers are a great attack targets
  – Use vendor SDKs, old software, with custom NIH software
  – Millions of vulnerable devices out there running Linux
Design, features and examples
Build system

- Written in GNU Makefile
- Produces *.ipk files for software packages and kernel modules
- Abstracts autotools, cmake, bare-Makefile, libtool
- Make menuconfig based user interface
- Dependencies resolution and configuration validation
- Partial rebuild of everything (packages, toolchain, kernel)
- Supports building for different targets within the same source tree
- Parallel whenever possible
Why not use buildroot or Yocto?

• Buildroot
  – Does not support packages
  – But was a great basis to work from!

• Yocto/OE
  – Too slow, too complex
Menuconfig based interface
Toolchain & kernel

- **Toolchain**
  - Internal build (default)
  - External (crosstool-ng, custom…)
  - Supports glibc, uClibc-ng and musl-libc

- **Kernel**
  - Vanilla kernel + OpenWrt/LEDE patches + platform specific patches
  - External kernel: directory or git repository
Package makefile

- Define name, revision
- Git URL, git commit, date
- Distribution metadata
- Include cmake macros
- Define package metadata (dependencies, location in menuconfig)
- How to create the package
- Add to the build system
Example work flow

- Clean, build and install jsonfilter into rootfs:
  ```
  make package/jsonfilter/\{clean,compile,install\}
  ```

- Force ethtool selection and download sources:
  ```
  CONFIG_PACKAGE_ethtool=m make package/ethtool/download
  ```

- Manage package patches with quilt:
  ```
  make package/ethtool/prepare QUILT=1
  cd build_dir/*//*/ethtool-*/
  quilt push/pop/delete/add
  ```
Platform definition

- Include macros
- Define architecture
  - Features
  - CPU type (ABI, family)
- Kernel version
- Default package selection
- Distribution (menuconfig) presentation
- Indicate what kernel image(s) to build
- Add to build system
Kernel example work flow

- **Build kernel modules**
  
  ```
  make target/linux/compile
  ```

- **Build kernel image and firmware**
  
  ```
  make target/linux/install
  ```

- **Manage kernel patches with quilt**
  
  ```
  make target/linux/prepare QUILT=1
  cd build_dir/target*/linux*/linux-x.y/
  quilt push/pop/add/delete
  ```

- **Switching between environments**
  
  ```
  ./scripts/env/new arm-platform
  ./scripts/env/switch arm-platform
  make -j42
  ./scripts/env/switch mips-platform
  ```
Even kernel modules are packages!

- Kernel package name
- Kconfig option to enable
- Dependencies
- File to install
- Insmod loading hints
- Add to build system

```
define KernelPackage/tg3
    TITLE:=Broadcom Tigon3 Gigabit Ethernet
    KCONFIG:=CONFIG_TIGON3
    DEPENDS:+=!TARGET_brcm47xx:kmod-libphy +kmod-hwmon-core +kmod-ptp
    SUBMENU:=$(NETWORK_DEVICES_MENU)
    FILES:=$(LINUX_DIR)/drivers/net/ethernet/broadcom/tg3.ko
    AUTOLOAD:=$(call AutoLoad,19,tg3,1)
endef

define KernelPackage/tg3/description
    Kernel modules for Broadcom Tigon3 Gigabit Ethernet adapters
endef

$(eval $(call KernelPackage,tg3))
```
Feeds

• Locations to package recipes
  src-git packages https://git.lede-project.org/feed/packages.git
  src-link custom /usr/src/openwrt/custom-feed

• Search, install and update additional packages
  scripts/feeds update packages
  scripts/feeds search "snmp"
  scripts/feeds/install snmpd
Development and deployment

- Image Builder
  - Kernel image
  - Recipes
  - Tools
  - Firmware image(s)

- SDK
  - Toolchain
  - Recipes
  - Tools
  - Packages

- Open source Software: Http, git, svn, files

Packages
Custom user-space, why?

- Modern systems require coordination between heterogeneous and discrete components
- User interfaces (CLI, web, GUI) change system configuration
- Networking devices are incredibly more complex (tunnels, provisioning etc.)
- Requirement for a proven, solid and consistent update mechanism
OpenWrt/LEDE software stack

**netifd**
- Event driven networking
- IPv4/v6 configuration
- Tunnels, VLAN, Wi-Fi
- Protocol handling

**procd**
- Process monitoring
- Jailing
- Hotplug, watchdog, syslog
- Init scripts support

**LuCI**
- Web interface
- Supports plugins/modules
- JSON-RPC
- Ubus export

**libubox**
- Event loop
- Utility library
- Socket abstraction
- Common data structures

**ubus**
- Socket-based IPC bus
- ACLs
- Export methods & signals
- Binary & JSON data format

**uci**
- Configuration storage
- C & LUA bindings
- Commit & rollback
System upgrades and failsafe

- System upgrades work consistently across devices:
  - Independent of the boot medium (SPI, NAND, eMMC)
  - Platform layer provides how to identify firmware image and where to flash kernel and root filesystem (partitions, mangling)
  - Scripts freeze system, preserve configuration files, and pivot_root to /tmp
  - Reboot into new version!

- Overlay FS allows marking the base system as read-only
  - But still allow read/write partition(s) for installable packages
  - Avoids wiping your entire system by accident

- Failsafe allows recovery of devices using device-specific buttons
  - Provides a recovery mechanism in case configuration is botched
Networking today

- Ethernet
- 3G/4G
- xDSL
- (euro)DOCSIS
- DHCP
- RA + DHCPv6
- IP(6)CP
- 6rd
- DS-Lite
- MAP-E
- MAP-T
- 464XLAT
Configure only the minimum

**Ethernet**
config interface wan
  option ifname eth1
  option proto dhcp

config interface wan6
  option ifname eth1
  option proto dhcpv6

**3G/4G**
config interface wan
  option ifname wwan
  option pincode 1234
  option apn #apn#

**PPPoX**
config interface wan
  option ifname eth1
  option proto pppoe
  option username john
  option password doe

**Wi-Fi**
config wifi-iface
  option device radio0
  option mode ap
  option encryption psk-mixed
  option key ...
  option ssid ELC
  option network lan

**Ethernet**
config interface lan
  option ifname eth0
  option type bridge
  option proto static
  option ipaddr 192.168.1.1
  option netmask 255.255.255.0
And let netifd do the magic

**Protocol handlers**
- DHCP, DHCPv6
- PPP

**netifd**
- Orchestration
- Event generation
- L2/L3/L4 stacking

**Physical devices**
- Ethernet
- xDSL
- Wi-Fi
- 3G/4G Modems

**firewall3**
- Netfilter/nftables frontend

**DNSmasq**
- DHCP server
- DNS cache
- DNSseq

**Network aware services**
- SMB
- UPnP/DNLA
- Dynamic DNS client

**Protocol clients**
- PPP
- DHCP client
- DHCPv6 client
Build-time security features

- Full/partial RelRO (configurable)
- Format-security checking (-Werror=format-security)
- Source fortification (-D_FORTIFY_SOURCE)
- Stack-smashing protector (user & kernel)
- Packages (*.ipk) are signed
Run-time security features

- Jails through procd to restrict filesystem access:
  ```
  procd_add_jail dnsmasq ubus log
  procd_add_jail_mount $CONFIGFILE $TRUSTANCHORSFILE $HOSTFILE /etc/passwd /etc/group /etc/TZ /dev/null /dev/urandom $dnsmasqconffile $dnsmasqconfdir $resolvfile $dhcpscript /etc/hosts /etc/ethers $EXTRA_MOUNT
  procd_add_jail_mount_rw /var/run/dnsmasq/ $leasefile
  ```

- Flexible seccomp definitions to white list system calls:
  ```
  procd_set_param seccomp /etc/seccomp/mdns.json

  
  "whitelist": [ 
    "read"
    "write"
    ..
    "brk"
  ]

  ```
And many more!

- Has existing ARM, MIPS and x86 targets that run in QEMU
- Packages with separate debug info
- Ex/inclusion of patented/specifically licensed packages
- Local package mirror, alternate download directory (corporate/development environments)
- Default IP, init-scripts, banner customization
Areas of improvements

- More continuous testing
  - Harder because of the wide variety of hardware
  - Leverage community and provide clear reporting guidelines
- Send more patches upstream
  - About 170 patches against Linux 4.9!
  - Migrate Qualcomm/Atheros AR71xx towards Device Tree (ath79)
- Opt-in security updates
- Documentation
  - Wiki
  - Table of hardware
  - Recommended, best supported, ranking of models
Conclusions

- It works great on your router, but equally well anywhere else!
- Fast, versatile, and flexible
- Turn-key user-space solution for products…
- … that you can ignore for development only
- Extremely active communities

OMG BEST

THING EVER!!!
OpenWrt/LEDE reunification status
What happened?

• On March 5\(^{th}\) 2016, a group of OpenWrt developers announced the formation of LEDE

• Two types of reaction:
  – Most people immediately welcomed LEDE and switched to it
  – A smaller group did not acknowledge the problem, and a flurry of emails ensued

• But essentially, it did signal there was a problem to be fixed with OpenWrt
Why LEDE?

• More transparency
  - All decisions made public
  - Give equal decisions rights to all project members
  - Establish clear processes and guidelines to operate the project (conflicts, external communication, release decisions..)

• Less centralization
  - Do not rely on single person owned infrastructure (DNS, servers, repositories…)
  - Freedom to move code and services based on newer requirements (CI, capacity etc.)

• Predictability
  - Make frequent releases
  - Leverage community testing
  - Easier integration process from contributor to developer
Meanwhile in OpenWrt

- Surprise,
Where are we today?

- Reunification terms:
  - LEDE code base to be used moving forward
  - OpenWrt team given LEDE repository access
  - Discussions on whether OpenWrt should stick as a name (trademark, larger popularity...)

- But right now, it’s a stalled discussion...
What next?

- **Release 17.01.0**
  - So we can focus energy again on bringing the two projects together again
  - We critically need open source, recent and better software for our routers, users should have control and freedom!

- **Meet, discuss and agree**
  - In person
  - More frequently
  - On the the reunification terms

- **And move forward together from there**

http://lists.infradead.org/pipermail/lede-adm/2017-February/000380.html
References

• Websites
  http://lede-project.org
  http://openwrt.org

• Mailing-lists
  lede-dev@lists.infradead.org
  openwrt-devel@lists.openwrt.org

• IRC
  #lede-dev @ freenode
  #openwrt @ freenode
Questions!

Florian Fainelli
f.fainelli@gmail.com

Slides under CC-by-SA 3.0