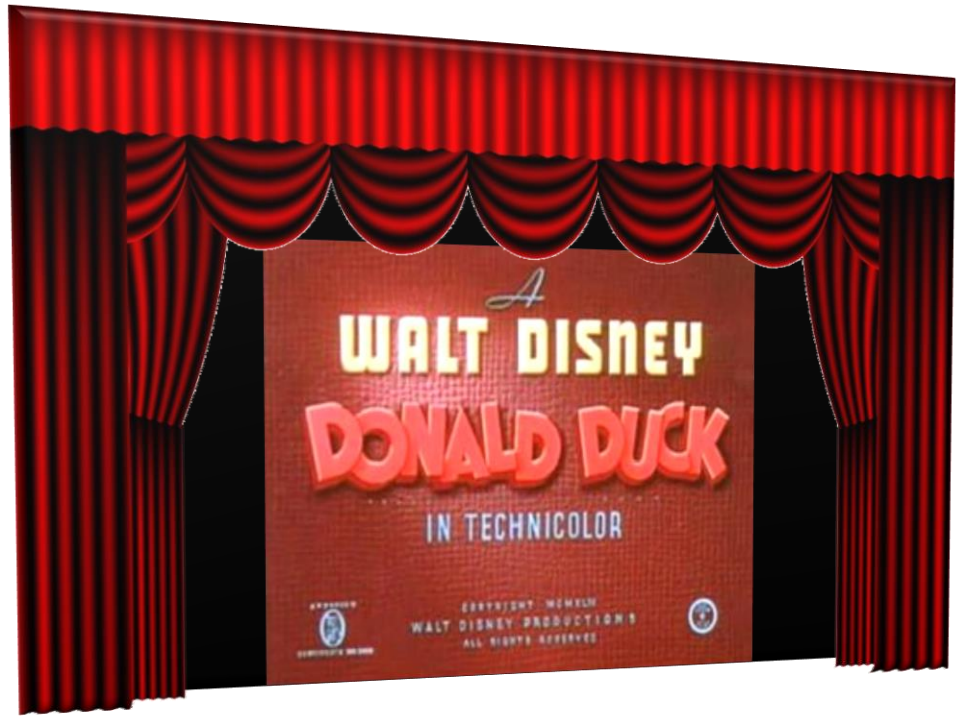
BRINGING OPENWRT TO MARKET

OPENWRT SUMMIT 2016 – BERLIN – 13 OCTOBER

JOS DELBAR

Introducing Technicolor

We are known for **Entertainment**
(film processing, special effects …)

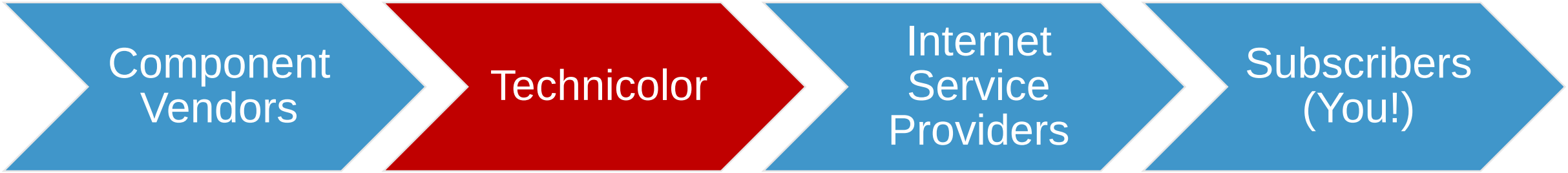


We also design and manufacture equipment for the **Connected Home** (routers, set-top-boxes …)

And we work with OpenWrt / LEDE!

technicolor
FEEL THE WONDER

Market Dynamics



Technicolor is not in the retail business – Our direct customers are ISPs

- Large scale operations: hundreds of products, tens of millions of homes
- Extensive and diverse set of requirements: triple play (Data, IPTV, VoIP), remote management …
- Access to component vendor's proprietary code: physical layer (DSL, Wi-Fi, LTE), hardware accel …
- High quality expectations: uptime, Quality of Service …
- Robust software processes: traceability, release management, continuous integration …

These market dynamics influence the way we work with OpenWrt

technicolor
FEEL THE WONDER

Why OpenWrt?

Solid platform

- Provides the right functionality for a standard home router
- Modular and platform agnostic
- Easy to build upon to create your own router products

Accessible

- Well known throughout the community and the industry
- Anyone can get started, no red tape
- Common reference platform for third parties

Diverse ecosystem

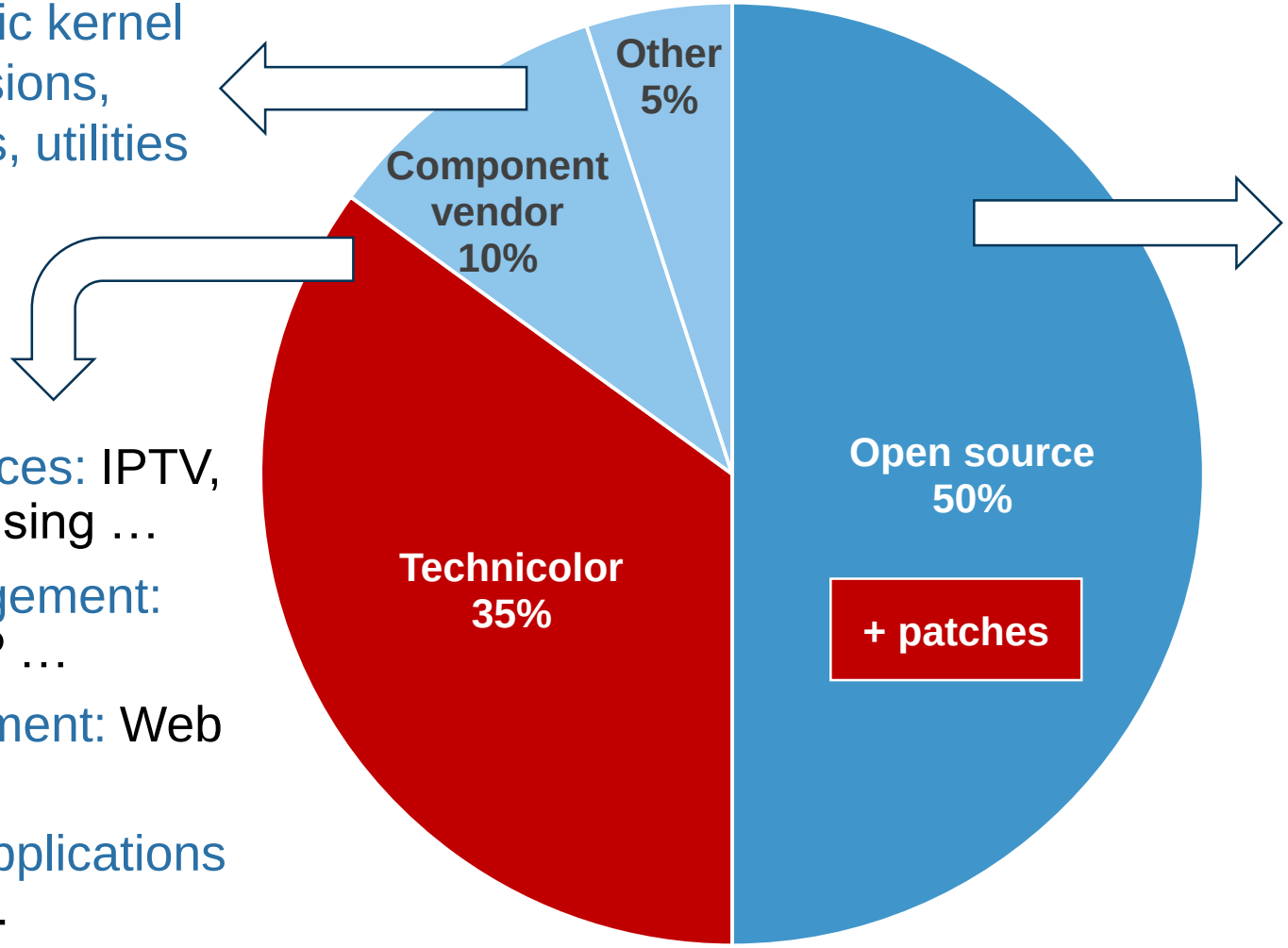
- Platform of choice for new router software initiatives ranging from commercial to non-profit
- Innovation across boundaries

technicolor
FEEL THE WONDER

Combining Proprietary and Open Source SW

(Simplified package distribution of a Technicolor OpenWrt based router)

Specific kernel extensions, drivers, utilities

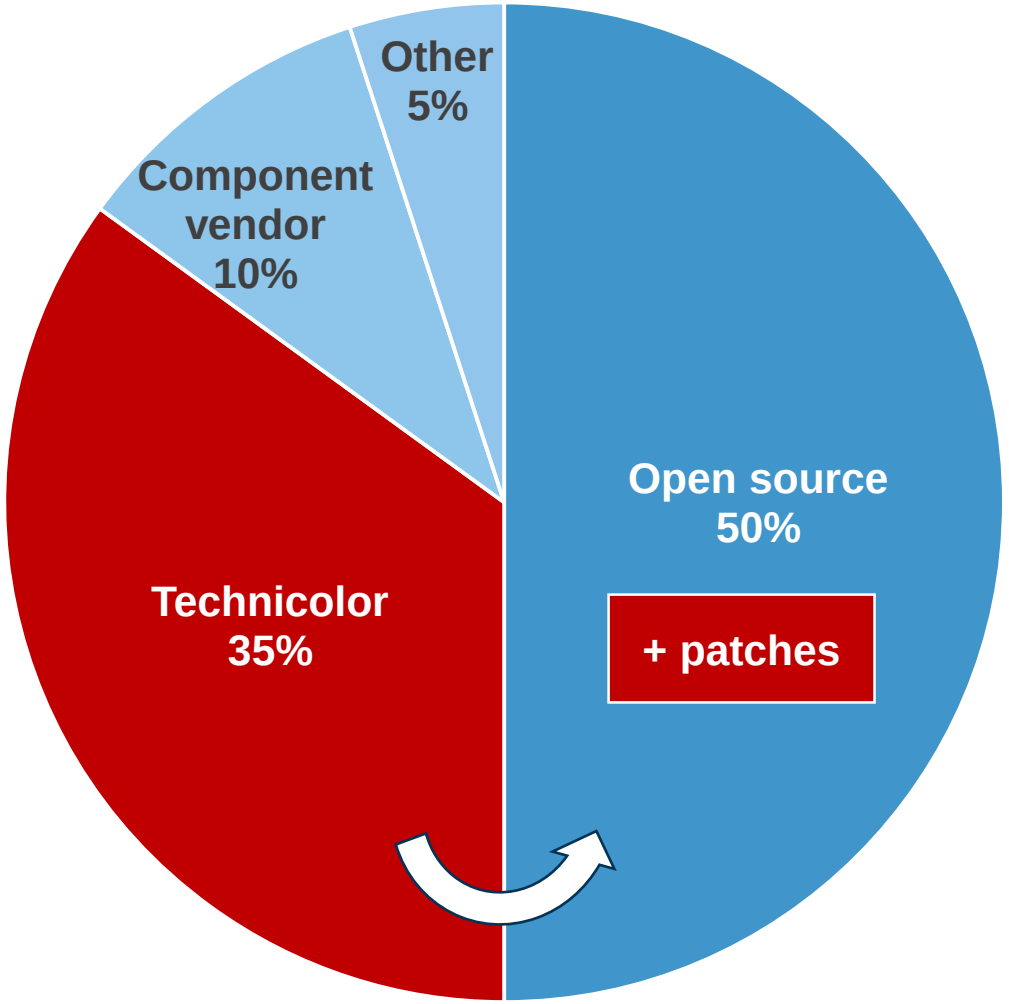


- Managed services: IPTV, VoIP, WAN sensing …
- Remote management: TR-069, SNMP …
- Local management: Web UI, TR-064 …
- Value-added applications and services …

- Base Linux system: kernel, drivers …
- OpenWrt framework: UCI, uBUS, rpcd …
- Standard networking: Ethernet, IPv4, IPv6, firewall, NAT, DHCP, DNS, QoS …
- Various applications: VPN, Samba, MQTT …

technicolor
FEEL THE WONDER

OpenWrt Contribution Strategy



Goal: achieving the right balance between software available in OpenWrt as open source, and software maintained in house

Reasons for Technicolor to contribute:

- Strengthen the community, increase adoption
- Stimulate new contributions, increase quality
- Reduce maintenance cost, avoid forking

Reasons for Technicolor *not* to contribute:

- Loss of competitive advantage
- Open source licensing constraints
- Effort to contribute and to maintain, low chance of adoption

technicolor
FEEL THE WONDER

Past and Present Examples

Adding new features

- GRE tunneling
- PPP Unnumbered
- IPinIP
- netifd link state propagation
- NTP server config via DHCP

Making existing packages ready for field deployment

- 464xlat
- odhcp6c and odhcpd

Package upsteps, additional config options

- dnsmasq, nginx and ngx_lua, strongswan, curl, openssl, openssh

Bug fixes

- ubus, libubox, firewall3, Ldoc, luacheck, procd

Unsuccessful contributions (*it happens!)

- MWAN

Potential future contributions

- TR-069 framework (first drop on github!)
  - https://github.com/dirkfeytons/transformer
  - https://github.com/dirkfeytons/lua-tch
- WAN sensing
- Device discovery

Working with Component Vendors

Integrating component vendors' proprietary code into OpenWrt is challenging

- Reference software kits sometimes not OpenWrt based
- Different kernels (older AND newer) than supported by latest OpenWrt release
- More functionality offered than supported by vanilla OpenWrt
- Different vendors implement the same function in different ways

Vendor kernels are supported by backporting patches to older kernel versions

- Currently supporting 3.3 (AA), 3.4, 3.10 (BB), 3.18 (CC), 4.1 (latest)

Functional extensions are supported by abstraction through UCI and uBUS

DSL Use Case

Context

- Establishing a DSL connection requires proper configuration of multiple layers:
  - Physical layer → mode (G.DMT, VDSL2, G.fast …), profile (8a, 17a, 30a …), handshake, EOC …
  - ATM or PTM → virtual path, virtual circuit, encapsulation, contract (CBR, rt-VBR …), priority …
  - PPP or DHCP
- Additionally, DSL connectivity events and statistics are required during operation

Problem statement

- Only the higher layers are well-defined in UCI; lower layers are ad-hoc and incomplete
- As a consequence, both users and vendors come up with their own alternatives for configuring DSL
- As a result, time is lost reinventing the wheel and supporting multiple vendors

Solution: establish a complete and vendor agnostic UCI and uBUS schema to model specialized hardware like DSL in OpenWrt

DSL Use Case

Context

- Establishing a DSL connection requires proper configuration of multiple layers:
  - Physical layer → mode (G.DMT, VDSL2, G.fast …), profile (8a, 17a, 30a …), handshake, EOC …
  - ATM or PTM → virtual path, virtual circuit, encapsulation, contract (CBR, rt-VBR …), priority …
  - PPP or DHCP
- Additionally, DSL connectivity events and statistics are required during operation

Problem statement

- Only the higher layers are well-defined in UCI; lower layers are ad-hoc and incomplete
- As a consequence, both users and vendors come up with their own alternatives for configuring DSL
- As a result, time is lost reinventing the wheel and supporting multiple variants

Solution: establish a complete and vendor agnostic UCI and uBUS schema to model specialized hardware like DSL in OpenWrt

OpenWrt example

```
package network

config vdsl 'dsl'
	option annex 'b'
	option firmware '/lib/firmware/vdsl.bin'
	option tone 'bv'
	option xfer_mode 'atm'

config atm-bridge
	option unit '0'
	option vpi '8'
	option vci '35'
```

Technicolor example

```
package xdsl

config xdsl 'dsl0'
	list multimode 'gdmt'
	list multimode 'glite'
	list multimode 't1413'
	list multimode 'adsl2'
	list multimode 'adsl2annexl'
	list multimode 'adsl2annexm'
	list multimode 'adsl2plus'
	list multimode 'vdsl2'
	list multimode 'gfast'
	list profile '8a'
	list profile '8b'
	list profile '8c'
	list profile '8d'
	list profile '12a'
	list profile '12b'
	list profile '17a'
	list profile '30a'
	option enabled '1'
	option rncenabled '1'
	option eoc_vendor_id 'BETMMB'
	option handshake_switch_timeout '0'
	option demod_cap_value '0x0010447a'
	option demod_cap_mask '0x0010447a'
	option aux_features_value '0x1064003'
	option aux_features_mask '0x1064003'
	option vdsl_cfg_flags_value '0x1200e00'
	option vdsl_cfg_flags_mask '0x1200000'
	option bondingsupport '1'

package xtm

config atmdevice 'atm_wan'
	option ulp 'eth'
	option vpi '8'
	option vci '35'
	option path 'fast'
	option enc 'llc'
	option td 'UBR'

config atmdevice 'atm_iptv'
	option ulp 'eth'
	option vpi '0'
	option vci '32'
	option path 'fast'
	option enc 'llc'
	option td 'VBR_nrt_iptv'

config trafficdesc 'UBR'
	option servicecat 'ubr'

config trafficdesc 'VBR_nrt_iptv'
	option servicecat 'nrtvbr'
	option pcr '512'
	option scr '128'
	option mbs '528'
```

TR-069 Use Case

Context

- TR-069 is a standardized protocol for managing a CPE install base remotely
- The protocol defines configuration actions including Get/Set of parameters and Add/Delete of objects
- TR-098 (InternetGatewayDevice) and TR-181 (Device) standardize these parameters and objects

Problem statement

- IGD does not map cleanly to UCI:
  - Some information is not available in UCI
  - Other information is available in UCI, but spread out over different objects or formatted differently
- TR-069 expects configuration to be applied to a running system while UCI is just a configuration store

Solution: reduce the gap between UCI and IGD/Device, create a mapping framework (transformer)

TR-069 Use Case

Vanilla OpenWrt UCI

- **config 'wifi-device'** → defines a physical radio device
  - option ‘channel’
- **config 'wifi-iface’** → defines a wireless network
  - option ‘ssid’
  - option ‘network’
  - option ‘encryption’

Device:2

- **Device.WiFi.** → defines interface objects (Radio and SSID) and application objects (AccessPoint and EndPoint)
  - **Radio.{i}.** → models an 802.11 wireless radio on a device
    - Channel
  - **SSID.{i}.** → models the MAC layer
    - SSID
    - LowerLayers
  - **AccessPoint.{i}.** → models an 802.11 connection from the perspective of a wireless access point
    - SSIDReference
    - Security.

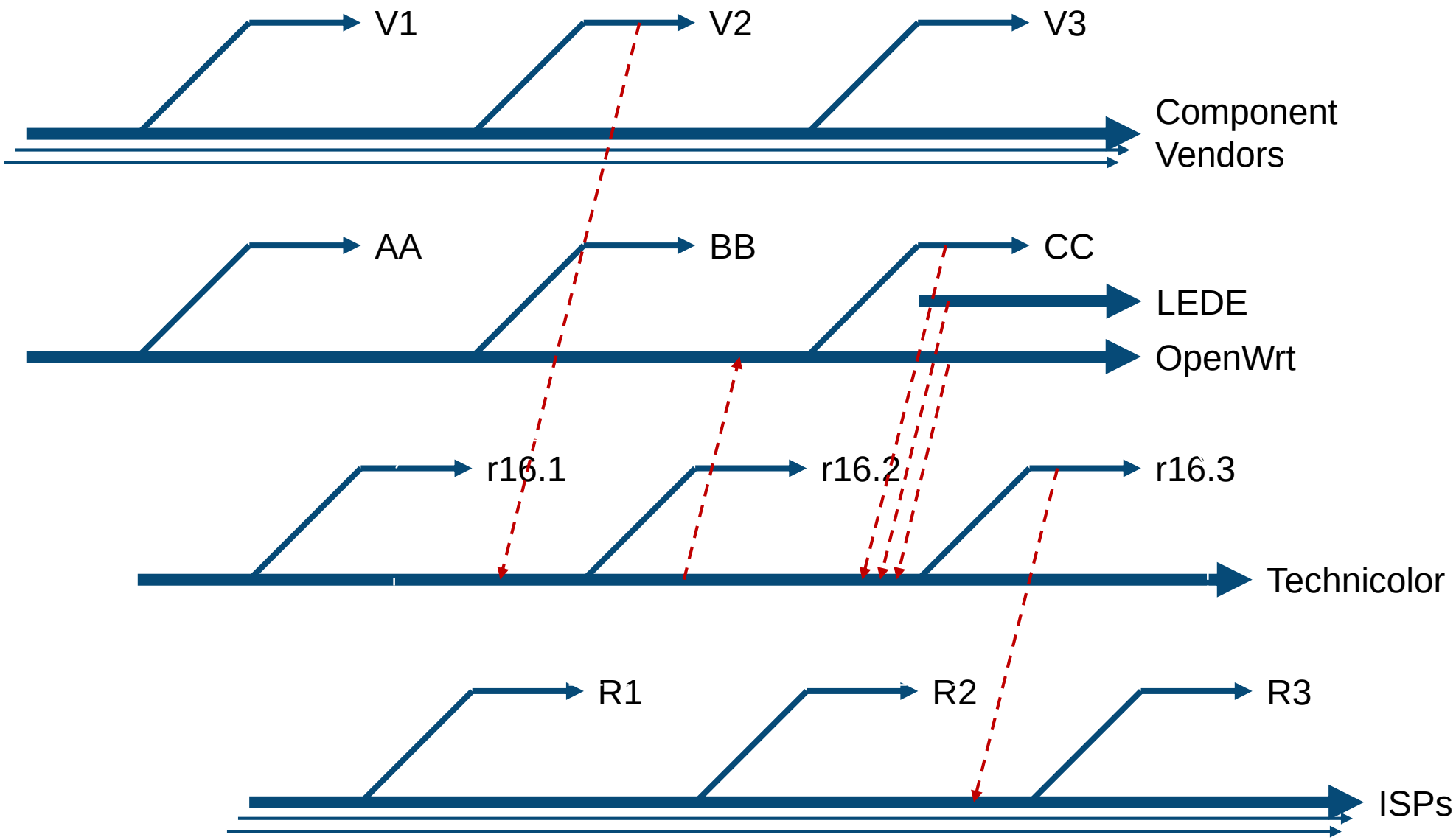
Technicolor UCI

- **config 'wifi-device’**
  - option ‘channel’
- **config 'wifi-iface’**
  - option ‘ssid’
  - option ‘network’
- **config ‘wifi-ap’**
  - option ‘iface’
  - option ‘encryption’

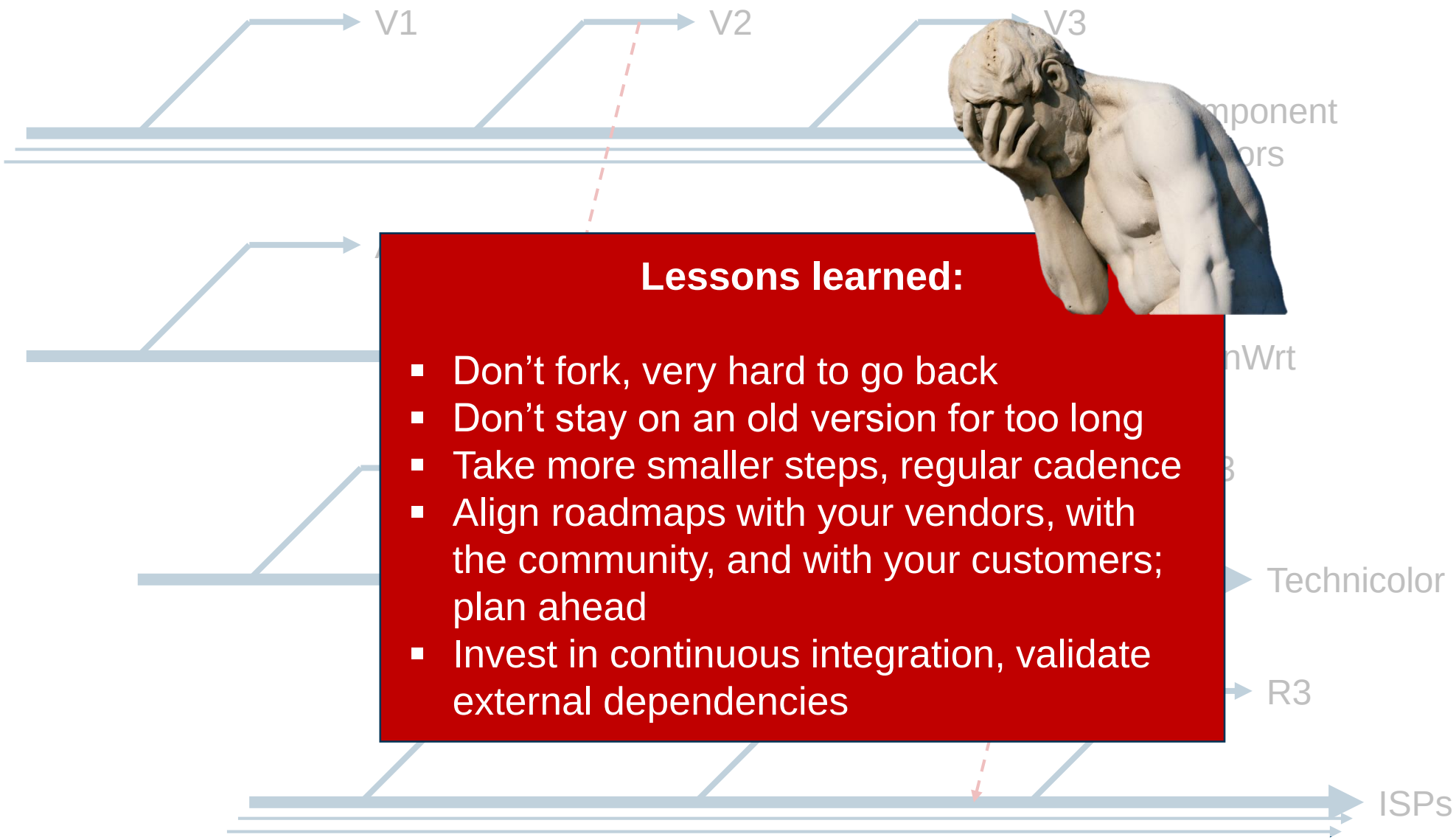
+ lots of additional parameters for chipset support, advanced Wi-Fi and remote management

technicolor
FEEL THE WONDER

Software Release Management



Software Release Management



Lessons learned:

- Don’t fork, very hard to go back
- Don’t stay on an old version for too long
- Take more smaller steps, regular cadence
- Align roadmaps with your vendors, with the community, and with your customers; plan ahead
- Invest in continuous integration, validate external dependencies

technicolor
FEEL THE WONDER

Wrapping Up

- OpenWrt / LEDE is important for Technicolor – and for Technicolor's customers
- OpenWrt works in the carrier industry – with some dedication and perseverance
- We look forward to continuing to work together to make OpenWrt better
- We hope you feel the same way!

Thank you

technicolor

FEEL THE WONDER