SUPPORTING MULTIPLE TYPES OF WAN INTERFACES

OPENWRT SUMMIT BERLIN

13 OCTOBER 2016

JOHAN PEETERS

Who am I

- Johan Peeters
  - Software architect at Technicolor
  - Connected Home Division
  - Firmware development on DSL gateways
  - Strong interest in :
    - forwarding core : qos, acceleration…
  - Using OpenWRT in our products since end 2012

Agenda

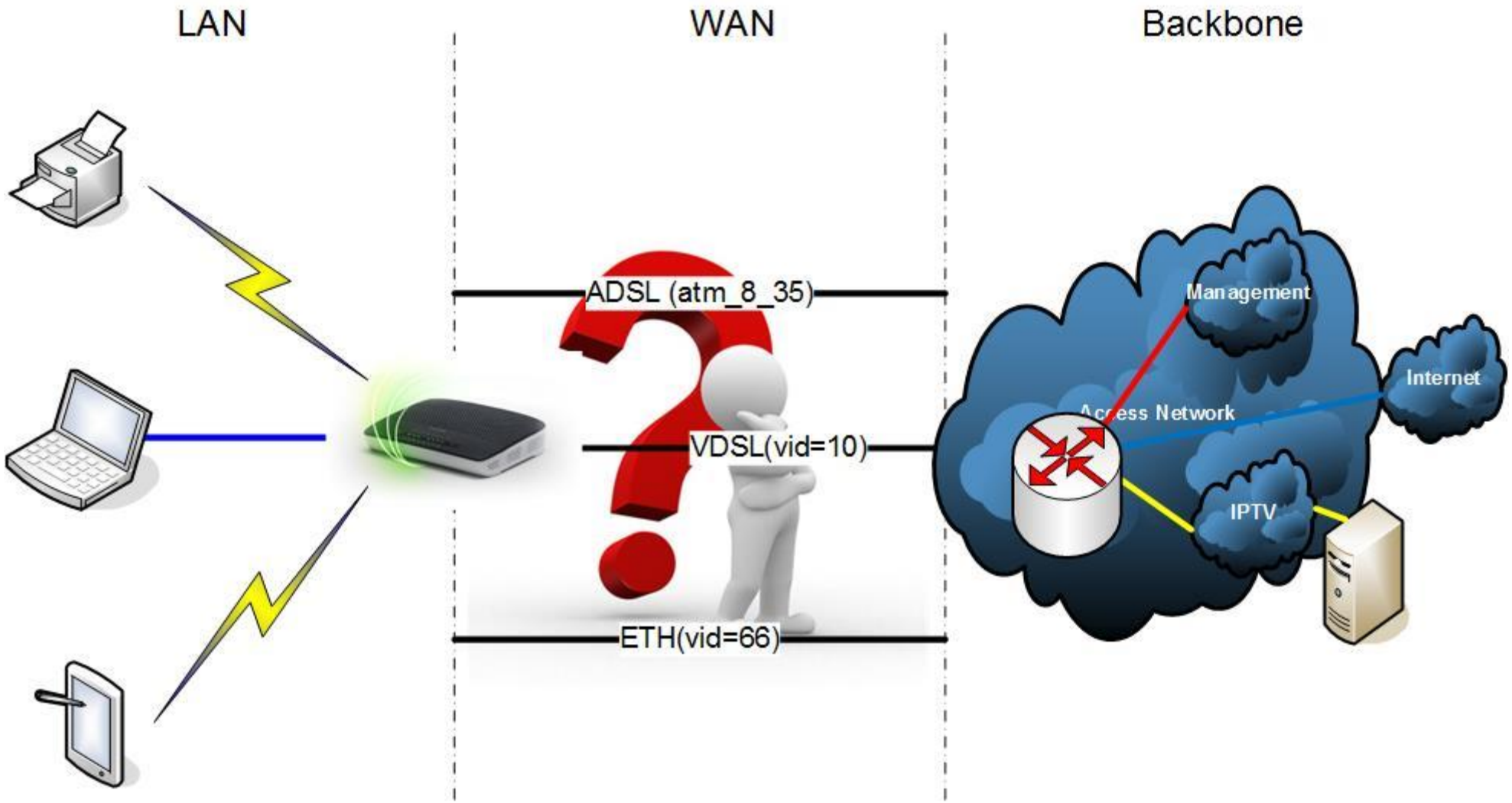
1. **Problem statement**
2. Technicolor solution
3. Next steps
4. Feedback

Problem statement

- Network operators do have different deployment models:
  - ADSL
  - VDSL
  - ETH (FTTH dual box solution)
  - Multiple Wan versus Single Wan (dependent on the region)

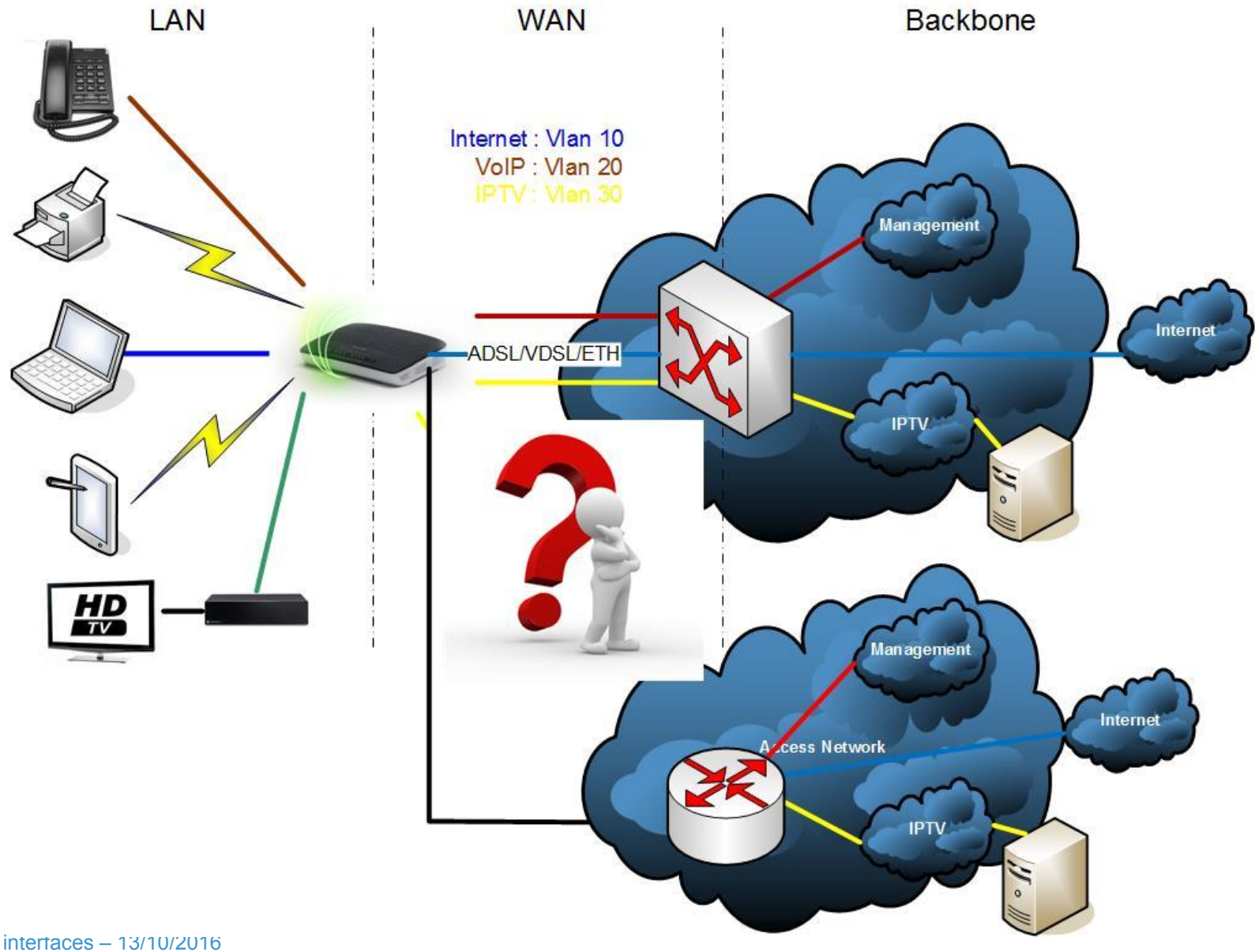
- Some network operators are in transition:
  - Switching from Differentiated Service towards Single Service
  - Switching from PPPoE towards IPoE
  - No flag day, migration is done step by step (region by region, customer by customer)

Problem statement



technicolor
FEEL THE WONDER

Problem statement

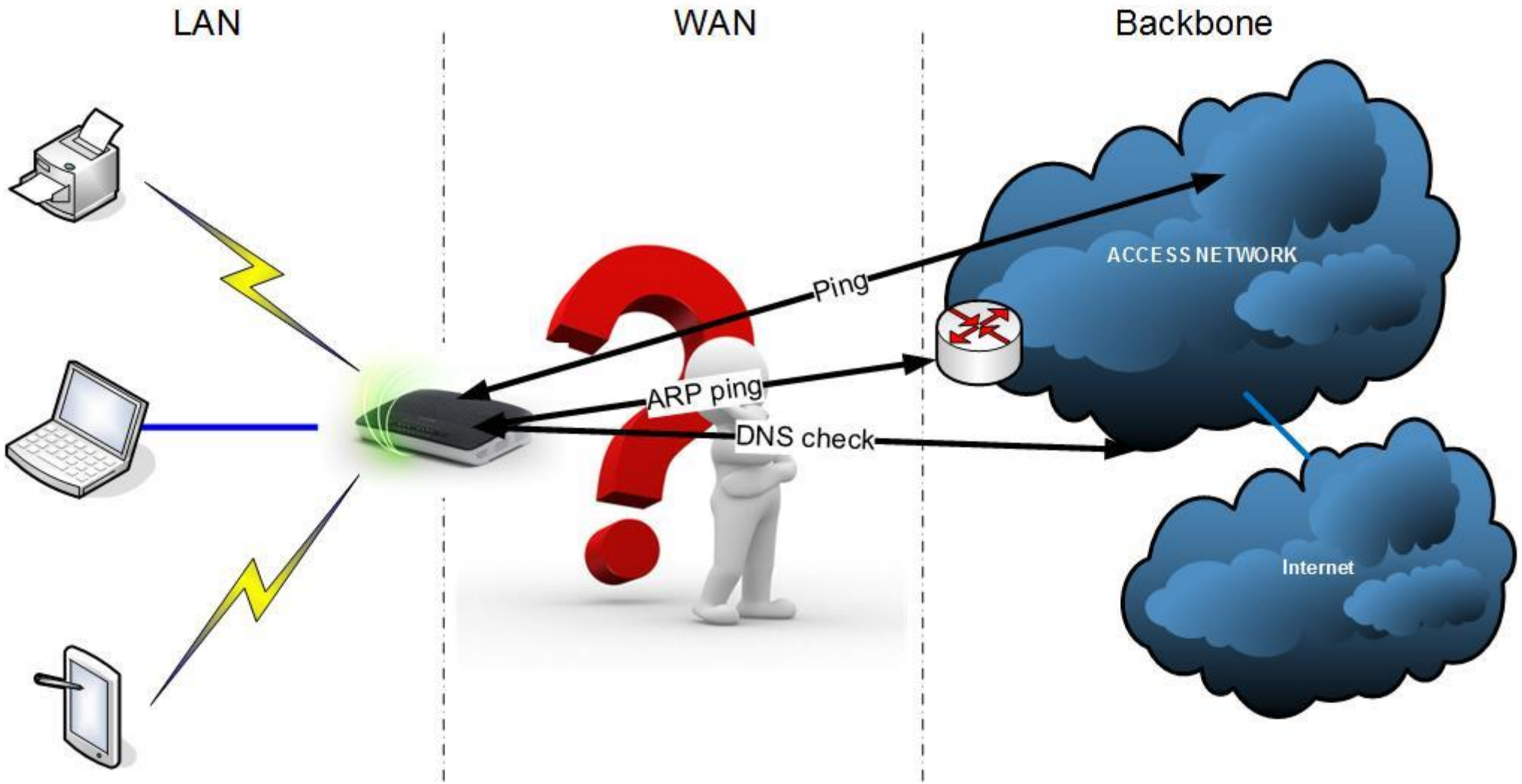


technicolor
FEEL THE WONDER

Problem statement for network operators

- Different connectivity checks on top of IPoE:
  - Ping a host in the network
  - Check correct functioning of a DNS server via resolving
  - Arp ping the next hop
- Are using large lease times on their IPoE interfaces
  - Still want to react fast on link failures in the access segment
- No connectivity check needed for PPP (LCP echo embedded in protocol)
  - UCI configuration to configure LCP echo behavior available by default

Problem statement



technicolor
FEEL THE WONDER

Problem statement

- Can’t we use remote management to configure the customer ?
  - No
  - Chicken and egg problem
  - Correct network stacking needed to enable remote management capabilities.

Problem statement

- In order to scale, the same build should handle the variety of connectivity models

- Having different kinds of builds is *unmanageable*:
  - For the operator
    - Software upgrade process needs to be in function of user
    - Each kind of build needs a customer acceptance test
  - For R&D:
    - Each kind of build needs his own quality assurance cycle.

Agenda

1. Problem statement
2. **Technicolor solution**
3. Next steps
4. Feedback

Solution

- Create a highly customizable daemon which
  - Senses the used access technology
  - Senses the used connectivity model
  - Configures the network stack in function of the sensed connectivity model
  - Reconfigures the network stack on the fly if the connectivity model is changed
    - Eg 1: transition from ppp towards ipoe
    - Eg 2: customer moves from one region to another

Solution

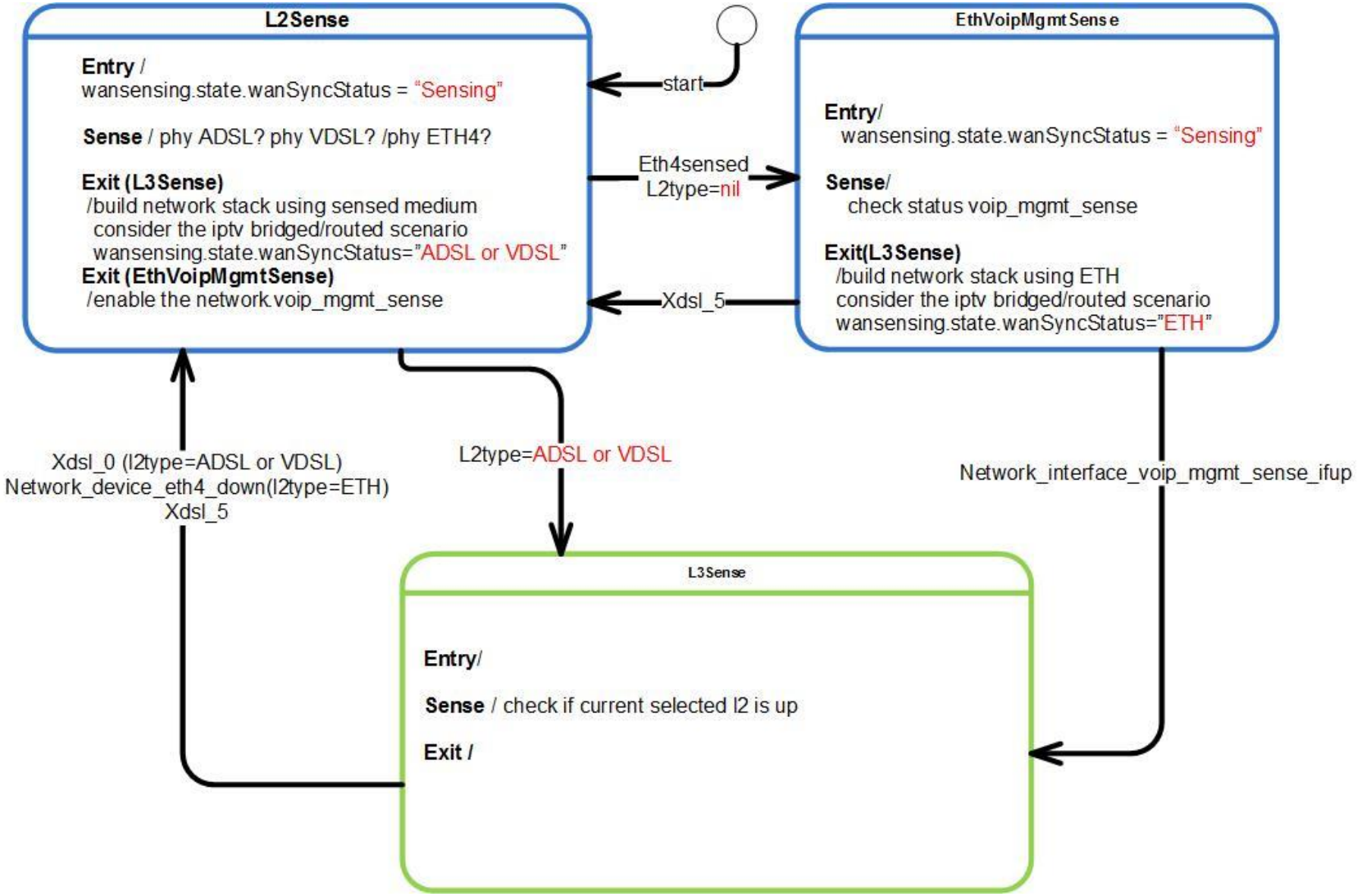
- Lua
  - Time to market
  - Not executing time critical code

- The core of the daemon is a configurable state machine
  - Distinguishes L2State and L3State types

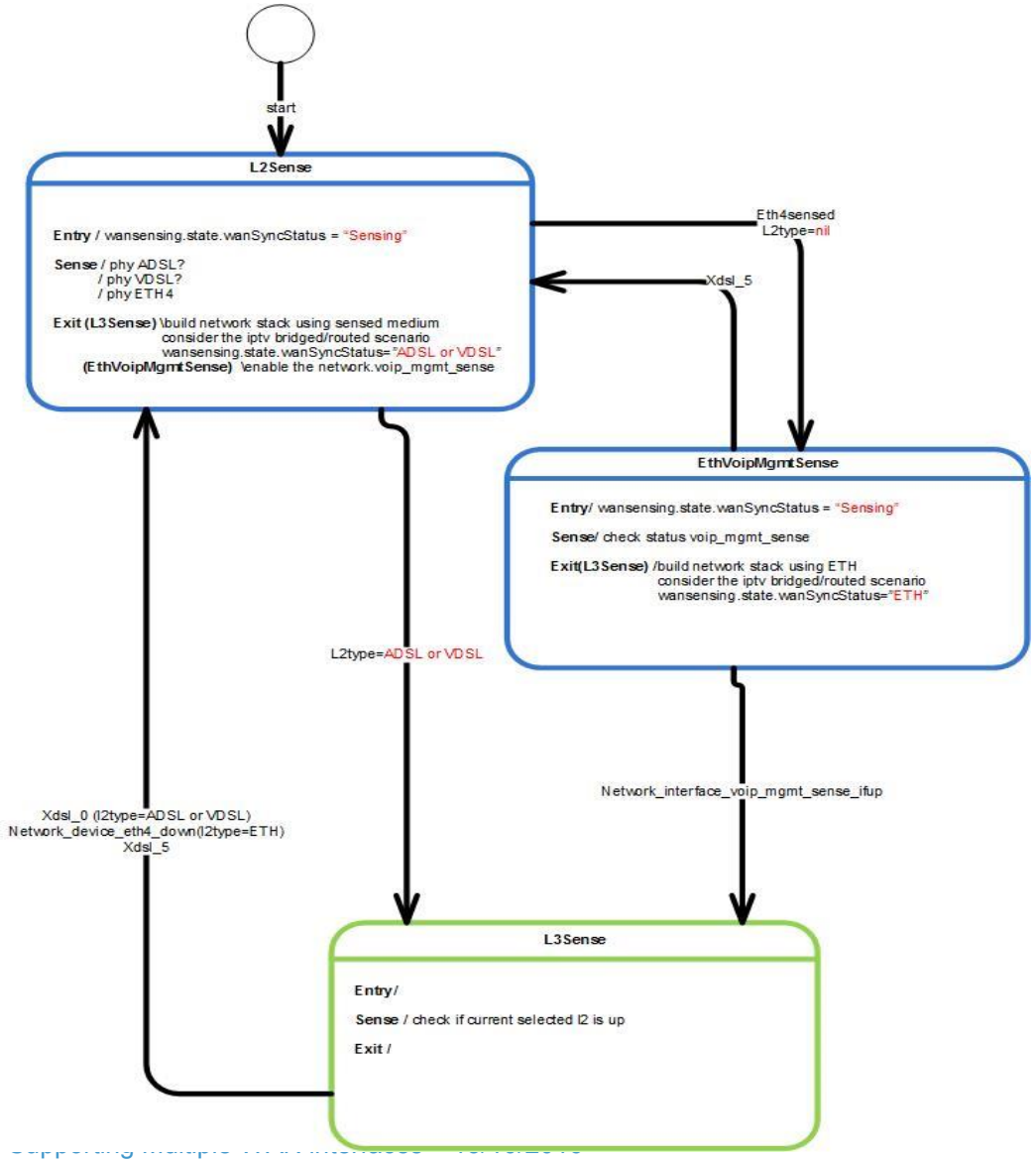
Solution

- The core of the daemon is a configurable state machine
  - Number of states are configurable via UCI (see /etc/config/wansensing)
  - Customization scripts do define:
    - Entry/Exit/Periodic check function
    - State transitions
    - Events it acts upon
  - Supported event types
    - Ubus interface up/down events
    - Netlink lower layer up/down events
    - Timeout events

Solution (Example)



Solution (Example – state chart)




```
config wansensing 'global'
        option enable '1'
        option initmode 'L2Sense'

config L2State
        option name 'L2Sense'
        option entryexits 'L2EntryExit'
        option mains 'L2SenseMain'
        option timeout '5'

config L2State
        option name 'EthVoipMgmtSense'
        option entryexits 'L2EntryExit'
        option mains 'EthVoipMgmtSenseMain'
        option timeout '5'

config L3State
        option name 'L3Sense'
        option entryexits 'L3SenseEntryExit'
        option mains 'L3SenseMain'
        option timeout '3600'
```


technicolor
FEEL THE WONDER

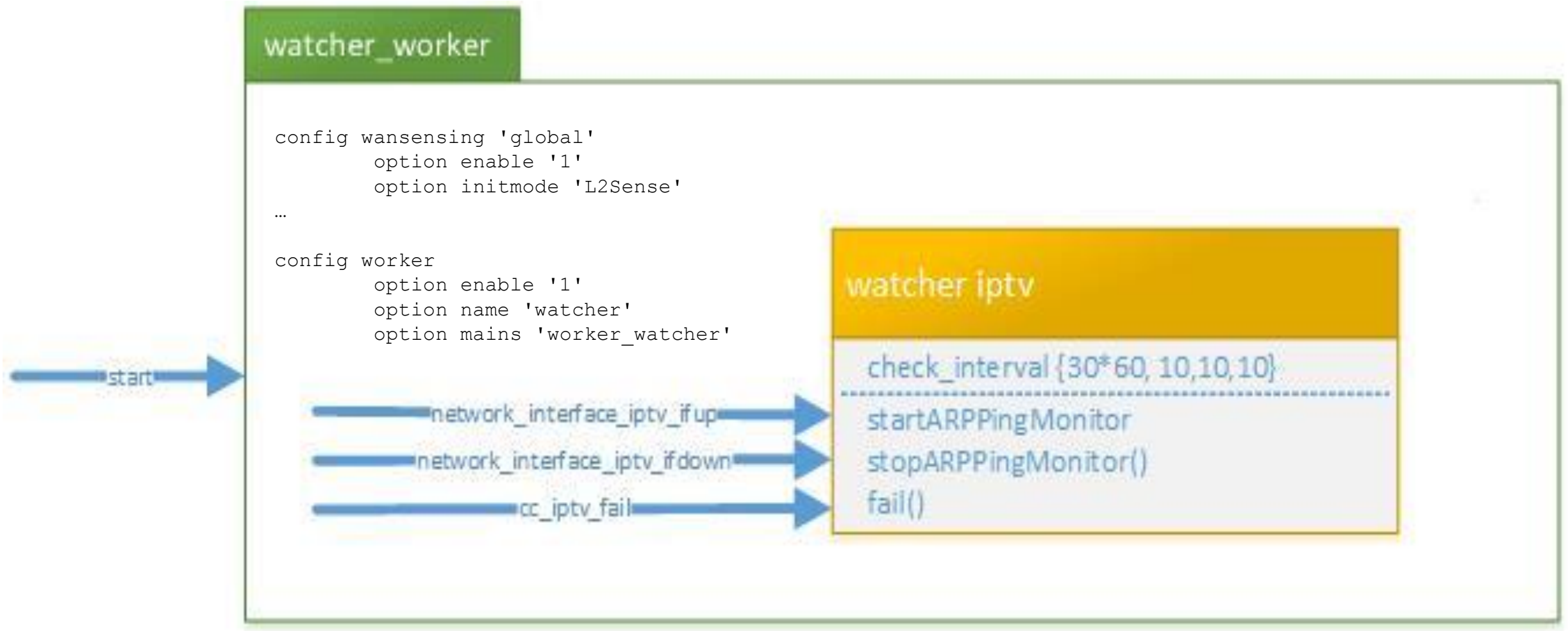
Solution (Example – script)

```
local M = {}
M.SenseEventSet = {
    'xdsl_5', --xdsl showtime
}

function M.check(runtime, event)
  local scripthelpers = runtime.scripth
  local uci = runtime.uci

  if  event == "xdsl_5" or event == "timeout" then
      -- check if xDSL is up
      local mode = xdslctl.infoValue("tpstc")
      if mode then
         if match(mode, "ATM") then
            return "L3Sense", "ADSL"
         elseif match(mode, "PTM") then
            return "L3Sense", "VDSL"
         end
      end
   end

  -- check if wan ethernet port is up
  if scripthelpers.l2HasCarrier("eth4") then
     return "EthVoipMgmtSense"
   end
   return
end

return M
```

Solution (Example – Event Registration)

```
-- Advanced Event Registration (availabe for version >= 1.0)
--
--   List of events can be changed during runtime
--   By default NOT registered for 'timeout' event
--   Support implemented for :
--       a) network interface state changes coded as `network_interface_xxx_yyy` (xxx= OpenWRT interface name, yyy =
ifup/ifdown)
--       b) dslevents (xdsl_1(=idle)/xdsl_2/xdsl_3/xdsl_4/xdsl_5(=show time))
--       c) network device state changes coded as 'network_device_xxx_yyy' (xxx = linux netdev, yyy = up/down)
--       d) add/delete events raised by the neighbour daemon
--            scripthelper function available to create the event strings,see
'scripthelpers.formatNetworkNeighborEventName(l2intf,add,neighbour)'
--       e) timeout event
--
-- @SenseEventSet [parent=#M] #table SenseEventSet
M.SenseEventSet = {
    ['timeout'] = true,
    ['xdsl_0'] = true,
    ['xdsl_5'] = true,
    ['network_device_eth4_down'] = true,
    ['network_interface_voip_mgmt_sense_ifup‘] = true
}
```

Solution (Example - connectivity check)



Solution (Example – script example)

```
local function IPoEWatch(runtime, intf, event)
   local conn = runtime.ubus
   local uci = runtime.uci

   if event=="start" then
      start_watcher(runtime,intf)
   elseif event=="fail" then
      conn:call("network.interface."..intf, "down", { })
      conn:call("network.interface."..intf, "up", { })
   elseif event=="ifup" then
      start_watcher(runtime,intf)
   elseif event=="ifdown" then
      stop_watcher(runtime,intf)
   end
end
function M.check(runtime, event)
   if event == "start" then
      IPoEWatch(runtime,"iptv",event)
      return
   elseif event == "network_interface_iptv_ifup"  then
      IPoEWatch(runtime,"iptv",'ifup')
      return
   elseif event == "network_interface_iptv_ifdown" then
      IPoEWatch(runtime,"iptv",'ifdown')
      return
   elseif event=="cc_iptv_fail" then
      IPoEWatch(runtime, "iptv", "fail")
      return
   end
   return
end
```

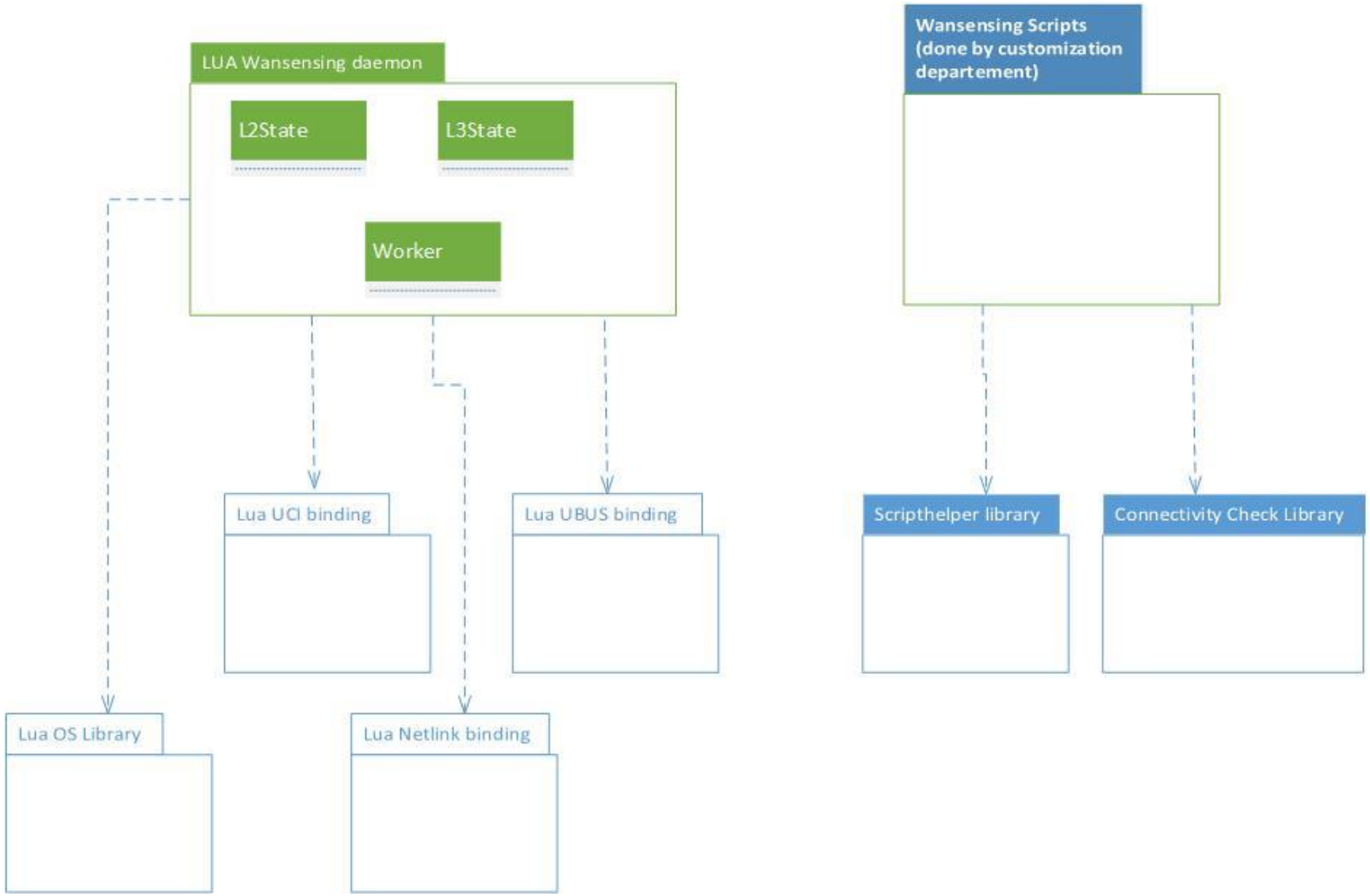
Solution (maturity level)

- Daemon used in shipment towards all our big customer
- Present in ten of millions of homes
- Present since first release of OpenWRT based gateways

Agenda

1. Problem statement
2. Technicolor solution
3. **Next steps**
4. Feedback

Next steps



Next steps

- Setting wheels in motion for a contribution of the Wan Sensing framework
  = Green packages in previous slide

- The scripts and scripts libraries will be kept in-house
  = Blue packages in previous slide

Agenda

1. Problem statement
2. Technicolor solution
3. Next steps
4. **Feedback**

Feedback

- Confronted with the same problems?
- How do you solve this problem?
- Open to hear alternative solutions

QUESTIONS ?

THANK YOU

technicolor

FEEL THE WONDER