PROTOTYPING A LIFECYLE MANAGEMENT APPLICATION

OPENWRT SUMMIT 2017, PRAGUE

DIRK FEYTONS

CONTENTS

Who

… am I?

- Dirk Feytons
- Software engineer at Technicolor
- Based in Edegem, Belgium
- Focus on management plane of residential/SOHO (xDSL) gateways
- One of the pioneers of our OpenWrt/LEDE-based software stack

What

… is lifecycle management (LCM)?

- Managing the lifecycle of dynamic software modules on network devices
  - Installing, updating, removing them
  - Starting, stopping them (if applicable)
  - See their state and change config
- Specified as part of TR-069 (and elaborated on in TR-157)
  - Standard remote management protocol used in telco world
  - HTTP(S) + SOAP RPCs from/to ACS server

What

… is LCM terminology?

- Execution Environment (EE)
  - Software platform on which software modules are installed
  - Examples: Linux, OSGi, Android, …
- Deployment Units (DUs) and Execution Units (EUs)
  - DU is an entity that can be deployed on an EE; it can be Installed, Updated and Uninstalled
  - DU contains zero or more EUs; these can be Started and Stopped
- RPCs
  - ChangeDUState RPC in TR-069 to manage DUs
  - {Get,Set}ParameterValues RPC on the datamodel to see information about EEs, DUs, EUs, start/stop EUs, change some config, …

Why

… this LCM prototype?

- Devices get more complex
  - Diagnosing problems remotely more difficult
  - Proactively detect and fix instead of end-users calling helpdesks
- Devices get more powerful
  - Room in flash/RAM/CPU for additional services
  - Technology to separate these services from core functionality
  - Service-based revenue model becomes viable
- Software stack consolidation
  - Linux and OpenWrt/LEDE as foundation
- Technicolor was asked by Vodafone to create a prototype

How

… did we approach this?

- Requirements outlined by Vodafone
  - Based on OpenWrt/LEDE
  - No vendor-proprietary components
  - Open-sourceable
  - Support TR-069 LCM model, but it shouldn’t be hardwired
  - Support Linux EE, but should be possible to add support for others
  - Keep security in mind
- Developed prototype meets these requirements

How

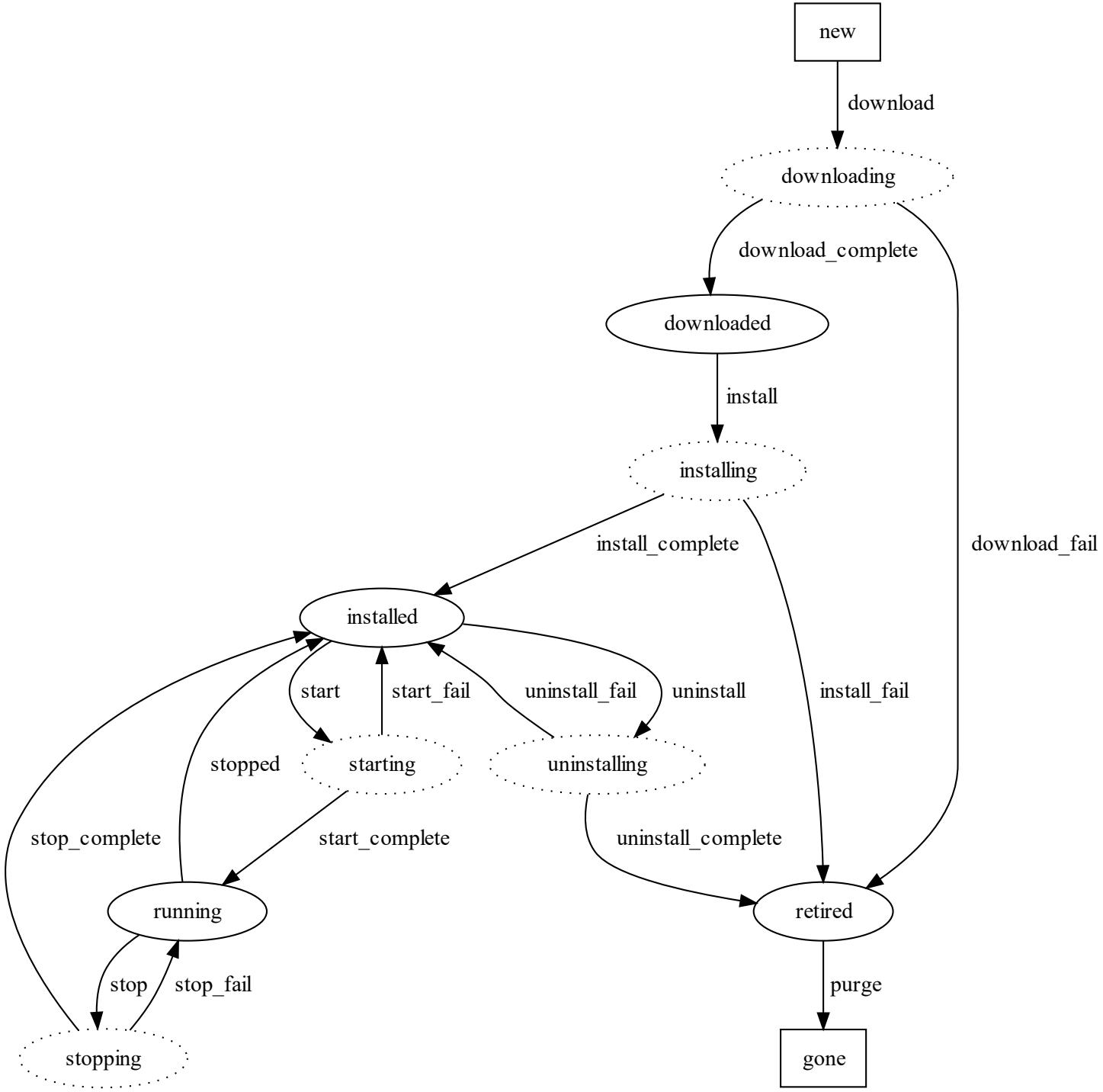
… does this LCM prototype work?

- Separate daemon: lcmd
  - Written in Lua for now
- lcmd exposes a ‘lcm’ ubus object
  - Methods:
    - `list_execenvs([name:String])`
    - `list_packages([properties:Table])`
    - `modify_package(ID:String, properties:Table)`
    - `install(URL:String, execenv:String [, username:String, password:String])`
    - `start([properties:Table])`
    - `stop([properties:Table])`
    - `uninstall([properties:Table])`
    - `delete([properties:Table])`
  - Notifications
    - `pkg.statechange`
    - `operation.complete`

How

… does this LCM prototype work?

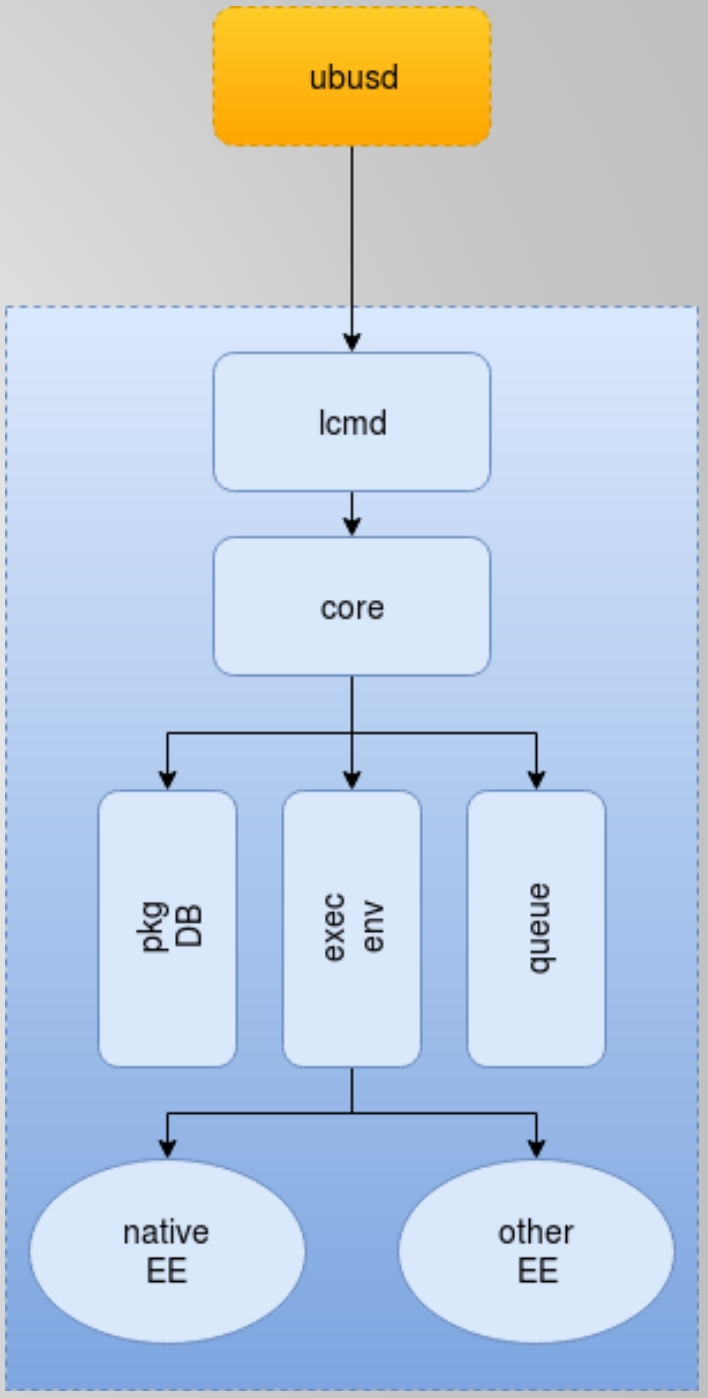
- Package state machine



How

… does this LCM prototype work?

- Interface inspired by TR-069 but kept generic
- Frontend handles ubus interfacing, maintains list of all packages, and interfaces with the EEs
- Operations received over ubus are queued and executed one by one in a child process
- Downloading is handled by the frontend using curl
- EE implementations are plugins
  - Provide implementation of `list()`, `install()`, `start()`, `stop()`, `uninstall()`



How

… does this LCM prototype work?

- Native EE implementation based on opkg
  - Readily available
  - Can separate LCM packages from system packages with `--offline-root`
- But:
  - It's a command line tool, not a library
  - Feed concept doesn't match TR-069 model of individual packages downloaded from URL
- Security
  - Existing signing support is based on the feed package list
  - We've added individual package signing support based on OpenSSL and public/private keypair
    - Add a `checksums` file to .ipk with SHA-256 hashes of `control.tar.gz` and `data.tar.gz`
    - Sign this file and include `signature.sig` in .ipk

Next

… steps for the prototype

- Code is prototype quality
- Rewrite in C?
- Persistency of state
- Native EE implementation needs to be fleshed out
  - What about opkg? What can we reuse?
  - Signing method can be kept probably?
  - Some ‘containerization’? Plain cgroups, or more?
  - Starting/stopping/monitoring
    - All info available from procd in all cases?

Thank you

technicolor

FEEL THE WONDER