Network configuration management at CERN

Arkadiy Shevrikuko
Stefan Stancu

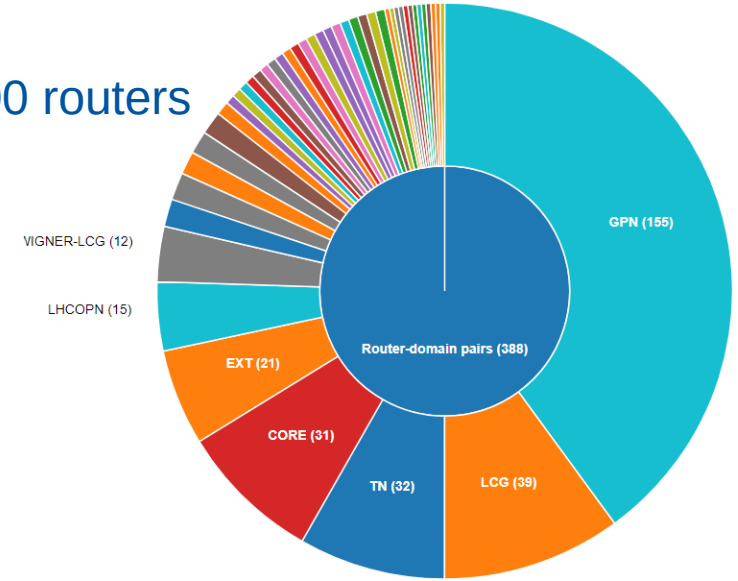
Outline

- Network overview
- Current solution: cfmgr
- Overview of open-source platforms
- Evolution plan

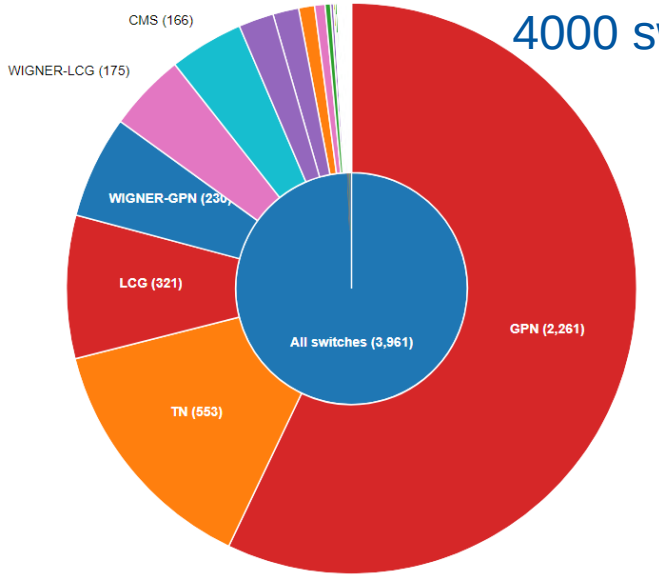
Network at CERN

- Around 4,4 k network devices
- Multi-vendor network (CERN procures equipment with open call for tenders)
- Specific configuration for each environment

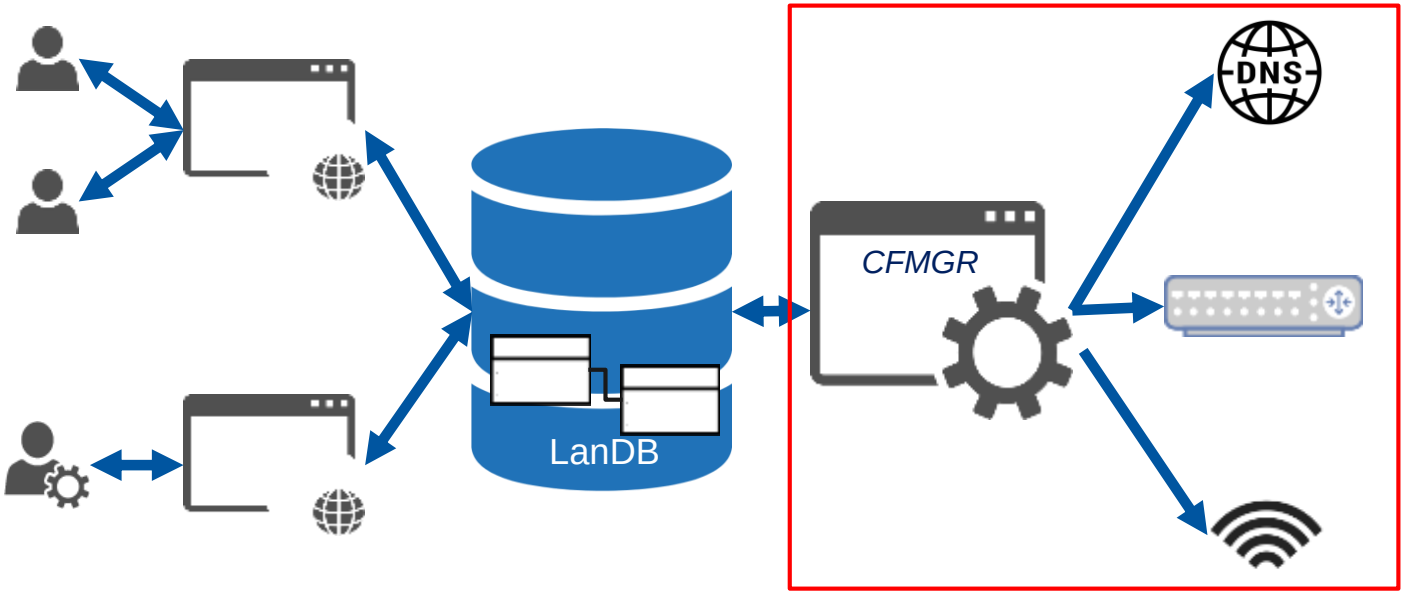
400 routers



4000 switches



Network configuration automation workflow at CERN

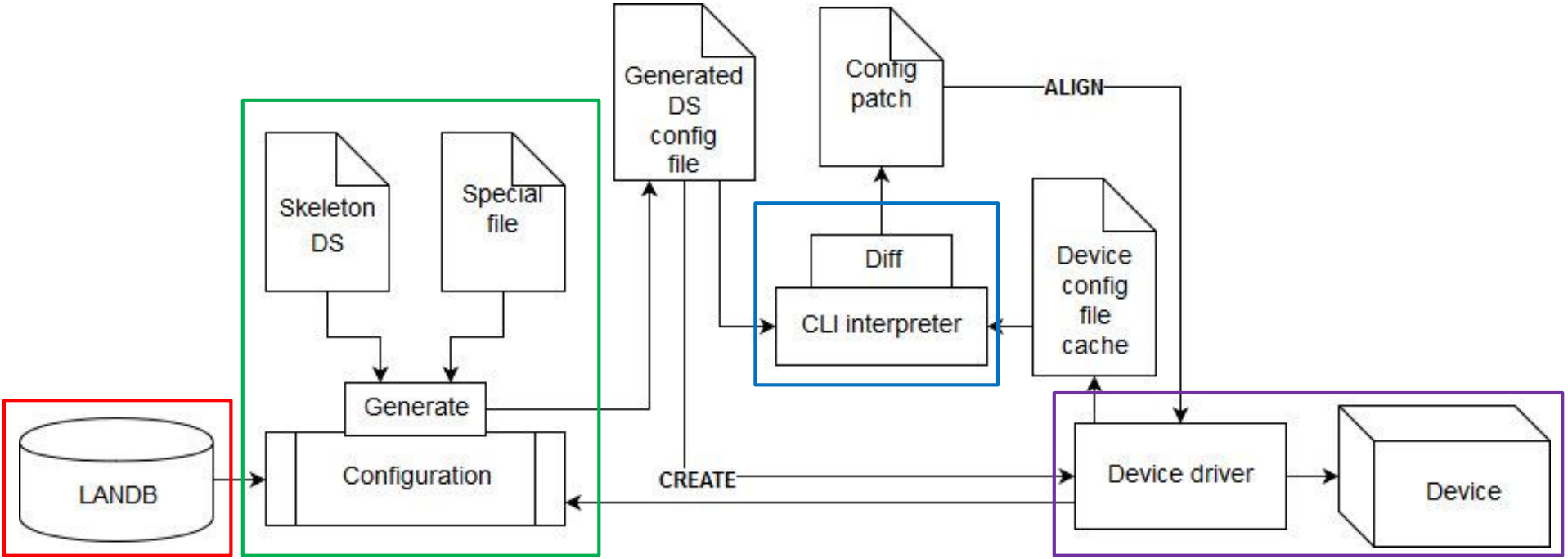


CFMGR – configuration management tool

What is cfmgr?

- Perl-based command-line tool
- Helps to ensure consistent configuration for the network
  - Uses central DB as a source of truth
- Features:
  - Multi-vendor support
  - Supports multiple operations
    - Basic configuration management (load, align)
    - Automatic configuration of ACLs and PoE
    - Large scale firmware management
    - etc…
  - Integration with Spectrum CA
    - Sends cron job status notifications

CFMGR architecture

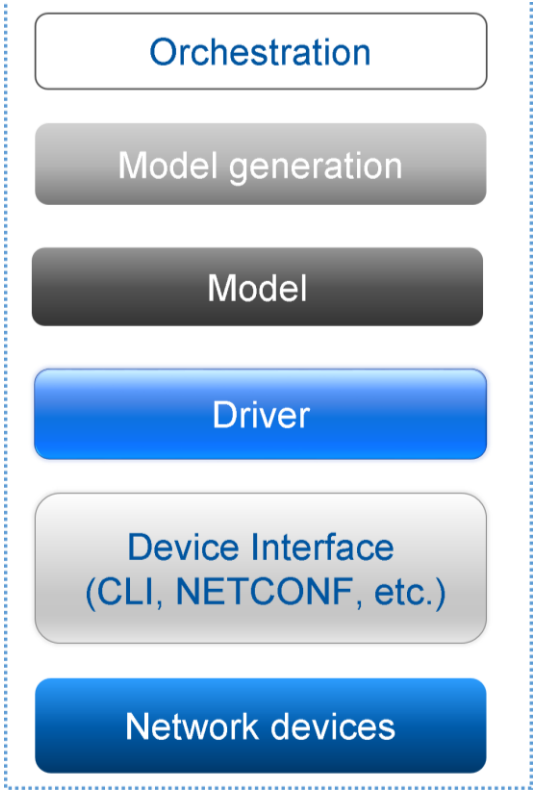


Evolution motivation

- Need to support a new router platform
  - different configuration workflow
- Decrease in popularity of Perl
  - Most network automation libraries (vendor or open-source) use Python
- Large code base with non-uniform/outdated coding practices
- No clear separation between different modules of the system

Open-source automation solutions

Network automation stack



- A network automation stack comprises multiple layers:
  - Orchestration: triggering the action of reconfiguring network devices
  - Model: device independent network configuration data
  - Driver and device Interface: translate the model to the appropriate device specific format and enforce it on the device

N.A.P.A.L.M.

- N.A.P.A.L.M. (Network Automation and Programmability Abstraction Layer with Multivendor support)
  - Python library that implements a set of functions to interact with different network device Operating Systems using a unified API.
- N.A.P.A.L.M. API covers two aspects:
  - Configuration management

Configuration support matrix

_	EOS	JunOS	IOS-XR	FortiOS	NXOS	IOS	Pluribus	PANOS	MikroTik	VyOS
Config. replace	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes
Config. merge	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes
Compare config	Yes	Yes	Yes [2]	Yes [2]	Yes [5]	Yes	No	Yes	No	Yes
Atomic Changes	Yes	Yes	Yes	No [3]	Yes/No [6]	Yes	Yes	Yes/No [6]	No	Yes
Rollback	Yes [3]	Yes	Yes	Yes	Yes/No [6]	Yes	No	Yes	No	Yes

- Python-based tool
- All orchestration is based on playbooks: programs, written in human readable language
- Generates configuration using template language (jinja2) and yaml data files
- No need to install additional software on the network devices

- For every device type, vendor, generation separate set of playbooks should be written
- Only one node controls whole network, which can cause concurrency issues during multiprocessing operations
- Simple set of operations supported

- Generation of the configuration using templates
- Python-based
- Reactive behaviour
- Scalable architecture
- Supports ACL generation out of the box

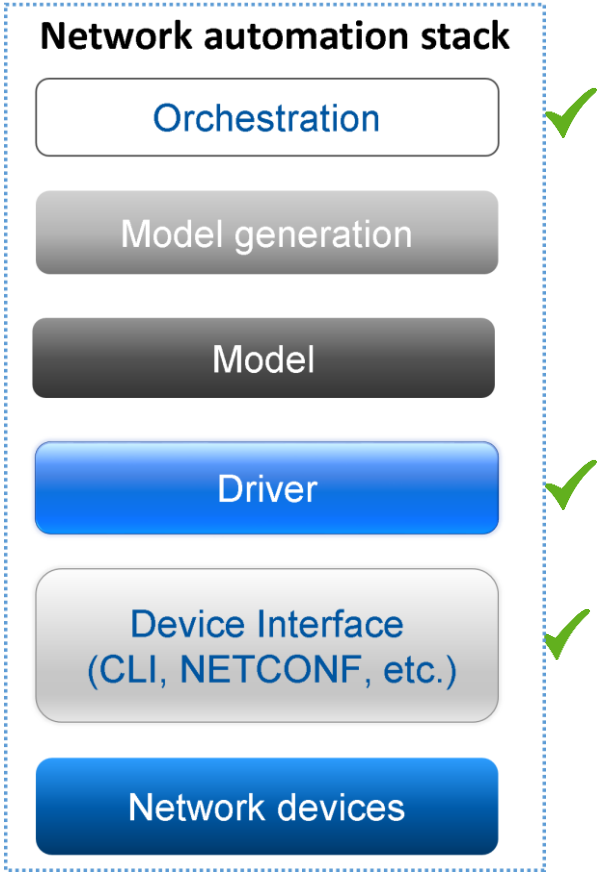
- Master-slave architecture
- Works over SSH in “headless” mode
- Provide layer of abstraction over N.A.P.A.L.M.

- Python based.
- StackStorm is a IFTTT(if-this-then-that) orchestration platform
- There are basic workflows for network configuration based on N.A.P.A.L.M.

- No multi-vendor support
- Was created to orchestrate servers, but not real network devices.
- N.A.P.A.L.M. is the main source of network device automation

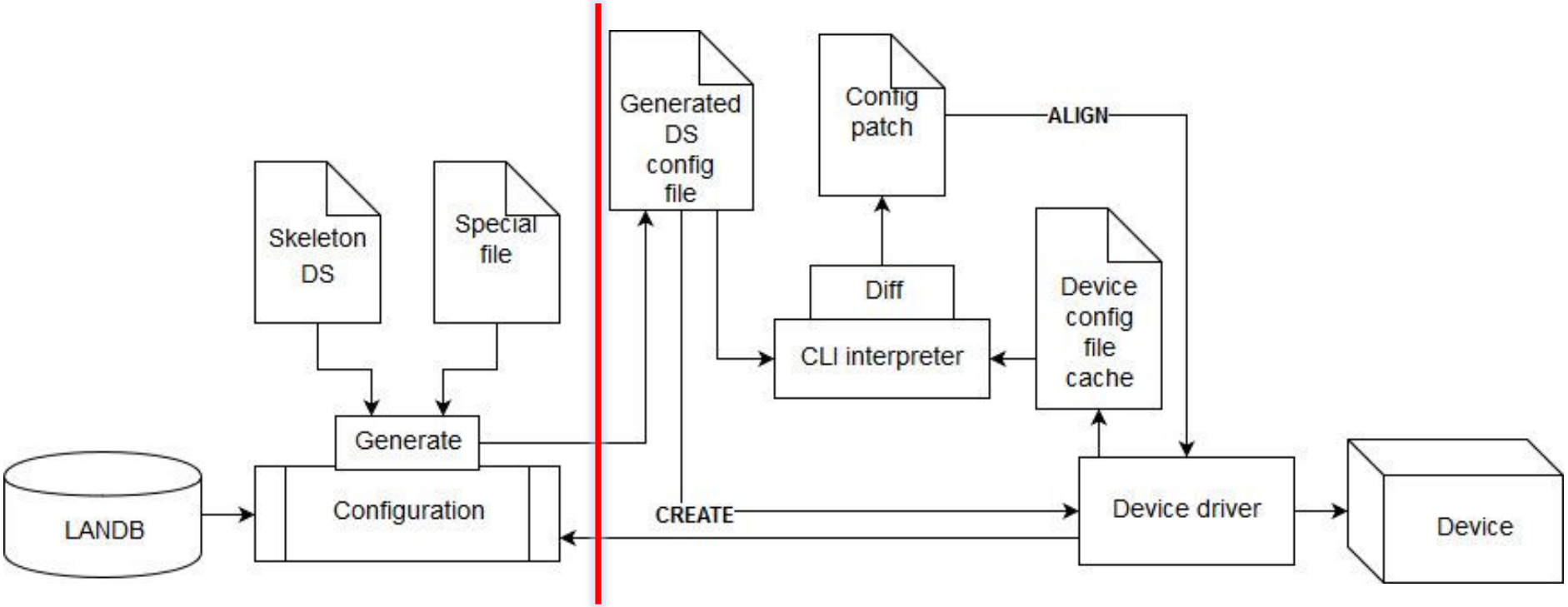
Open-source platform applicability

- There is no perfect automation platform which can cover full network automation stack.
  - All network configuration capabilities are build on top of N.A.P.A.L.M.
  - Some of them have scalability issues
  - Lack of multi-vendor support
  - Generation of the configuration is still our duty
    - Configuration models are vendor specific
    - Partial cover of configuration generation with OpenConfig

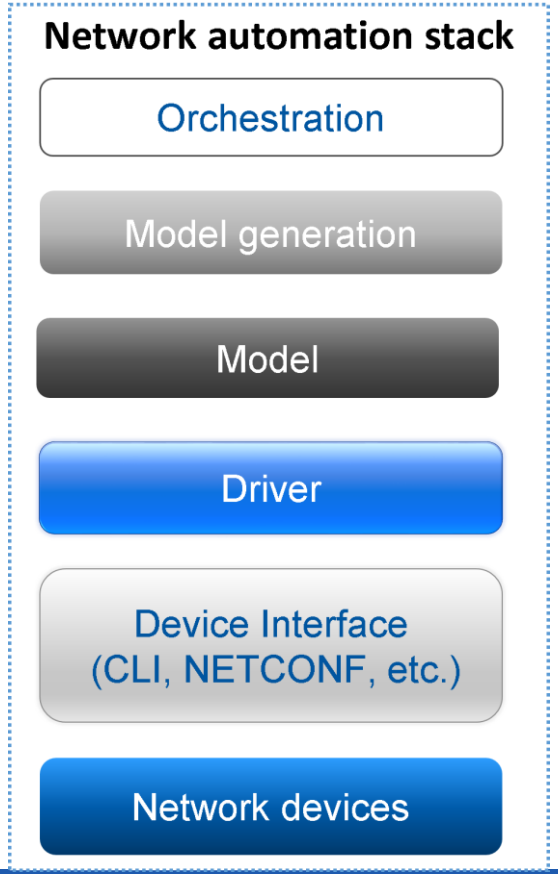


Evolution plan

CFMGR – current state



Evolution plan (1)

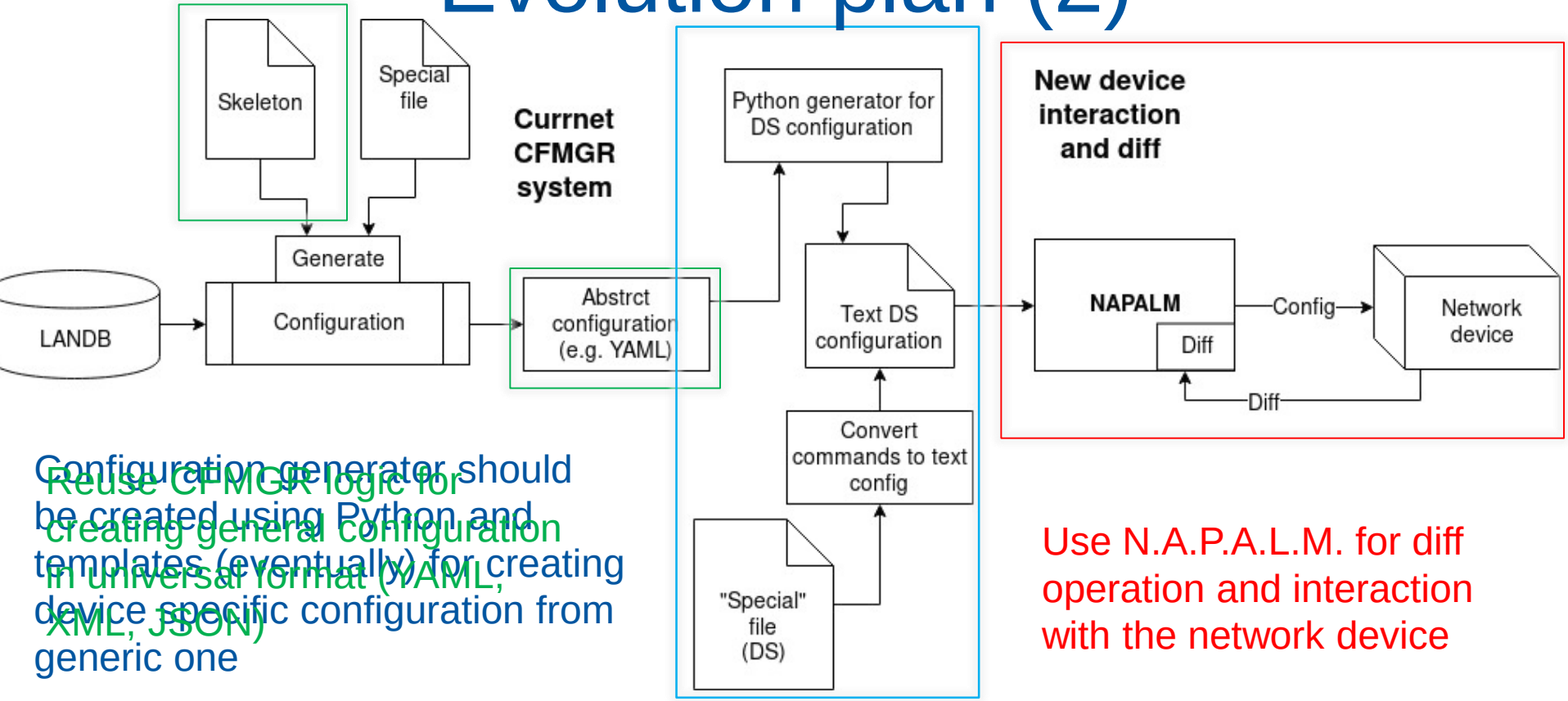


Not covered by any automation platform. Should be developed

N.A.P.A.L.M.

Python libraries, provided by vendors

Evolution plan (2)



Configuration generator should be created using Python and templates (eventually) for creating device specific configuration from generic one

Reuse CFMGR logic for creating general configuration in universal format (YAML, XML, JSON)

Use N.A.P.A.L.M. for diff operation and interaction with the network device

Evolution plan (3)

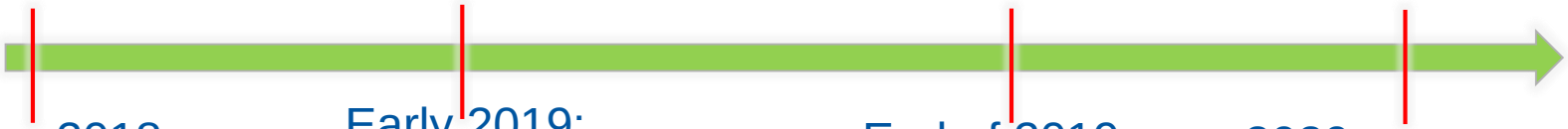
- Decouple concerns:
  - Configuration generation: in-house
    - Generate vendor independent configuration (Perl → Python)
      - Gradually use REST interface for retrieving database information
    - Generate vendor specific configuration (use XSD/YANG schemas if available)
  - Configuration enforcement
    - Use N.A.P.A.L.M. and vendor libraries (Python)

Evolution plan (4)

- Transition phase, need Perl – Python interaction
  - Use ‘system’ to call full programs
    - Structured input/output data (JSON and/or YAML)
  - Proxy server for communicating to devices:
    - Expose methods via XML-RPC

- Start with new vendor support
  - Back-port to vendors still in production once the platform becomes complete

Timeline



Autumn 2018:
Beginning of the development.

Early 2019:
NAPALM driver and vendor specific configuration generator. Support of Juniper configuration workflow

End of 2019:
Python implementation of core vendor independent configuration elements

2020+
Full Python-based configuration generation
Additional features (e.g. VLANs)
Integrate other vendors

Summary

- Network configuration automation is a must
  - Due to the scale and diversity of CERN's networks
- Today there is no commercial or open-source product, capable of replacing cfmgr
- We are evolving the tool in order to leverage open-source tools with multi-vendor support:
  - Faster and easier integration of the new vendors
  - Easier maintenance of the tool
  - Faster implementation of new features
- New calls for tender will most likely require programmatic configuration APIs

Questions?


CERN

Things to improve

- Make use of open-source libraries for automation and integrate them to the existing system
- Switch to modern programming language
- Integrate vendor specific APIs for generation of the configuration
- Clear border between different modules of the cfmgr

Update plan

- Different parts of cfmgr will be replaced with the new ones
  - Python-based modules which are going to interact with Perl cfmgr using XML RPC mechanism
- Initial plan is to have support only for new vendors (Juniper)
  - Once architecture is clear and robust – implement support for other vendors

Update plan

- Use NAPALM to organize communication between cfmgr and network device
  - Using RPC calls from perl to python
- Generation of the configuration
  - Using XSD or yang schemas provided by vendors, to generate appropriate classes
  - Convert class based configuration to text version
- Replacement of the platform, which communicates with LANDB
  - Rest API is available for deriving required data from LANDB