An introduction to OpenSDN source code

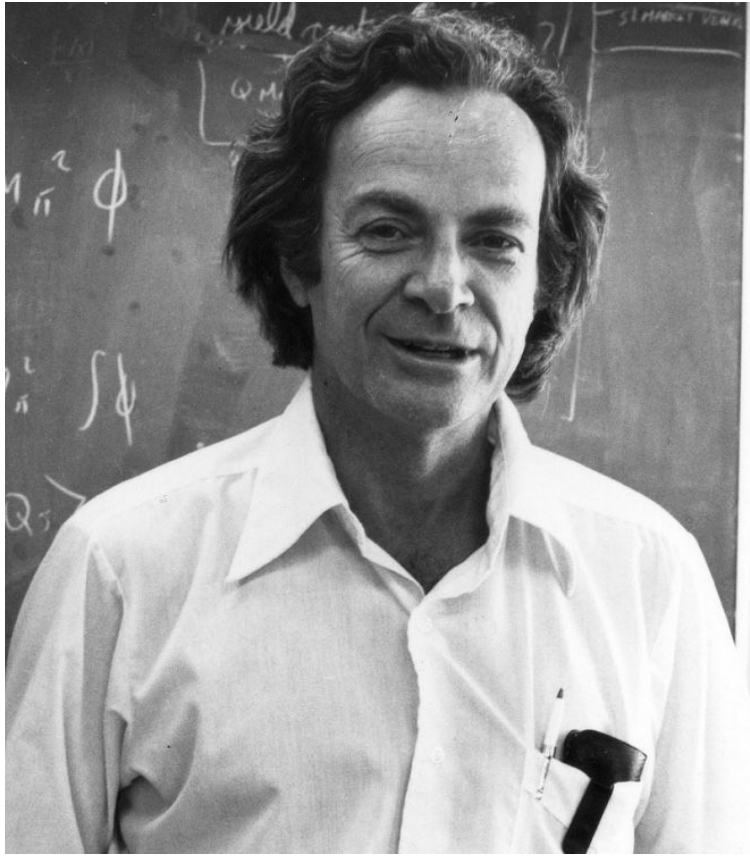
By Matvey Kraposhin

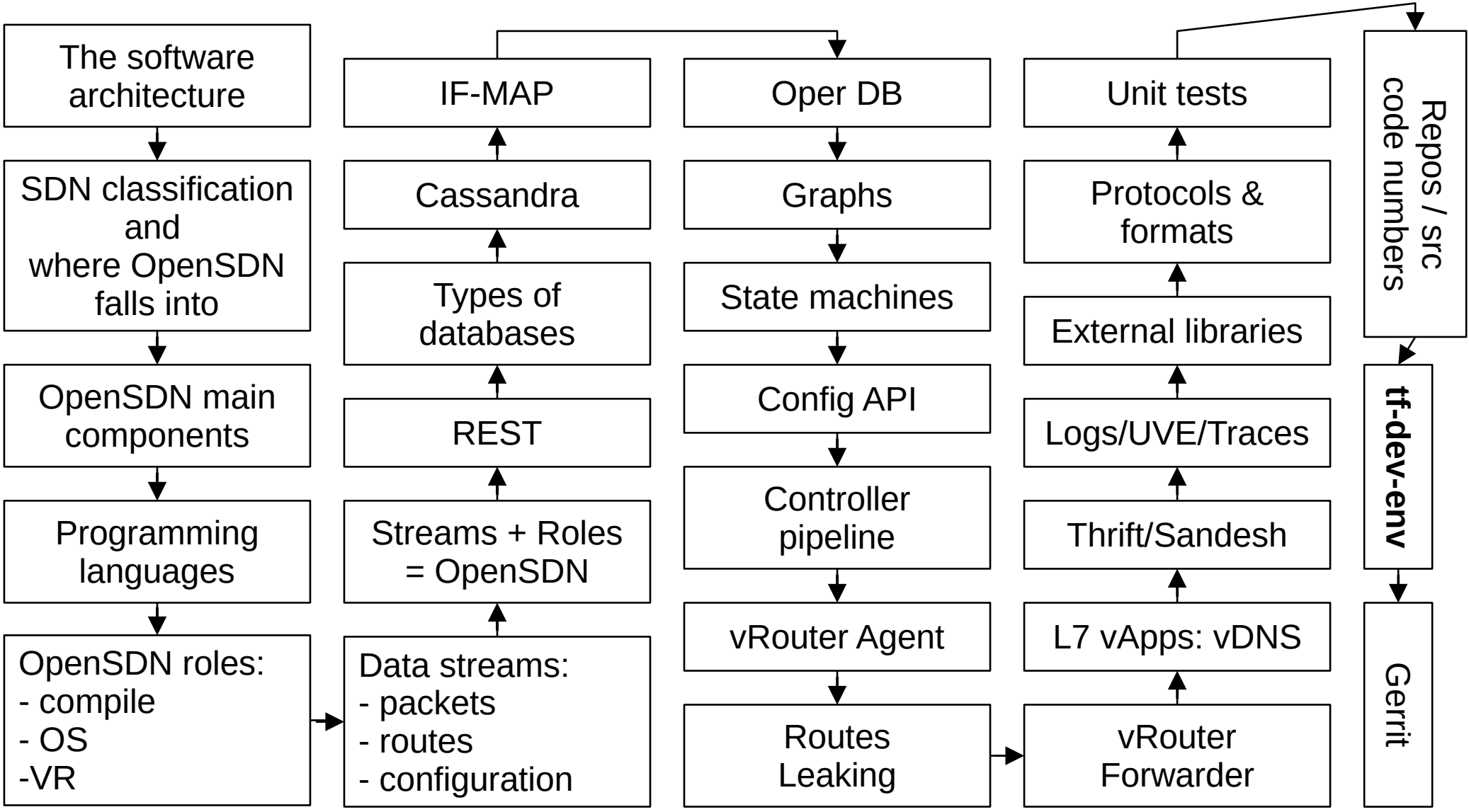
kraposhin.online

nipa.cloud OpenSDN BootCamp 18 - 20 June 2025

Instead of foreword

- Even the material might sound too complicated, it can be useful in the future or for others
- Low level details are actually as a multiplication table: we hate learning it, but we appreciate it later
- If something is not clear to you, then probably it is not totally clear for me




The software architecture
SDN classification and where OpenSDN falls into
OpenSDN main components
Programming languages
OpenSDN roles:
- compile
- OS
-VR
Data streams:
- packets
- routes
- configuration
Streams + Roles = OpenSDN
REST
Types of databases
Cassandra
IF-MAP
Routes Leaking
vRouter Agent
Controller pipeline
Config API
State machines
Graphs
Oper DB
vRouter Forwarder
L7 vApps: vDNS
Thrift/Sandesh
Logs/UVE/Traces
External libraries
Protocols & formats
Unit tests
Gerrit
tf-dev-env
Repos / src code numbers

Why does architecture is needed?

- Architecture as a logically connected compound of verified technical decisions and solutions is bounded by:
  - Functional requirements, e.g.:
    - Network virtualization
    - Switch and router virtualization
    - Network Function Virtualization
  - Functional properties and indicators, e.g.:
    - scalability, performance, user friendliness etc

An SDN or an SDN controller?

- The terms are different, while sometimes they are used interchangeably
- The SDN is a technology, while an SDN controller is a part of some SDN solution (platform)
- Many references use “SDN controller”, so do we

SDN controller properties: references

1. Scalability, Consistency, Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers by S. Ahmad and A. Hussain Mir

2. Reliability Challenges in Software Defined Networking by V. Netes and M. Kusakina

3. Performance Evaluation of Different SDN Controllers: A Review by S. Askar and F. Keti

4. OpenContrail as SDN controller for NFV infrastructure in AT&T network by A. Gorbunov

5.High-Performance Packet Routing Acceleration for Cloud Systems Using High Bandwidth Memory by T. Shimuzu et all.

6. Accelerating Contrail vRouter / Netronome report

SDN controller properties

- **Scalability** is an ability of an SDN controller to response to growing demands from the network (requests, flows, etc)
- **Consistency** means having a stable and updated network wide view
- **Reliability** is an SDN controller’s degree of resilience to failures in the network
- **Security** is an ability to withstand malicious attacks
- **Performance** is a quantity that shows how many requests an SDN controller can handle per some period of time

Types of SDN controllers

- Centralized:
  - problems with scalability, reliability and security
  - small enterprises, campus networks, domain specific networking in small-scale data centers and edge networks
  - example: NOX
- Distributed:
  - addresses issues of a centralized type, but at the cost of consistency, performance
  - tradeoff between performance and scalability might result in degradation of reliability
  - Break into: 1) both logically and physically distributed and 2) logically centralized, physically distributed
  - example: ONOS, ODL ,TF/OpenSDN
- Hybrid
  - Include centralized and distributed (for specific vendors)

SDN controller landscape

- Controllers:
  - OpenSDN/Tungsten Fabric/OpenContrail
  - OVN/OVS
  - OpenDayLight, ONOS
  - RunSDN (Russian fast SDN controller written in C++)
  - NOX/POX
  - Ryu
- References
  - https://tech.ginkos.in/2019/04/list-of-openflow-controllers-for-sdn.html
  - https://www.oreilly.com/library/view/sdn-software-defined/9781449342425/ch04.html
  - https://en.wikipedia.org/wiki/List_of_SDN_controller_software

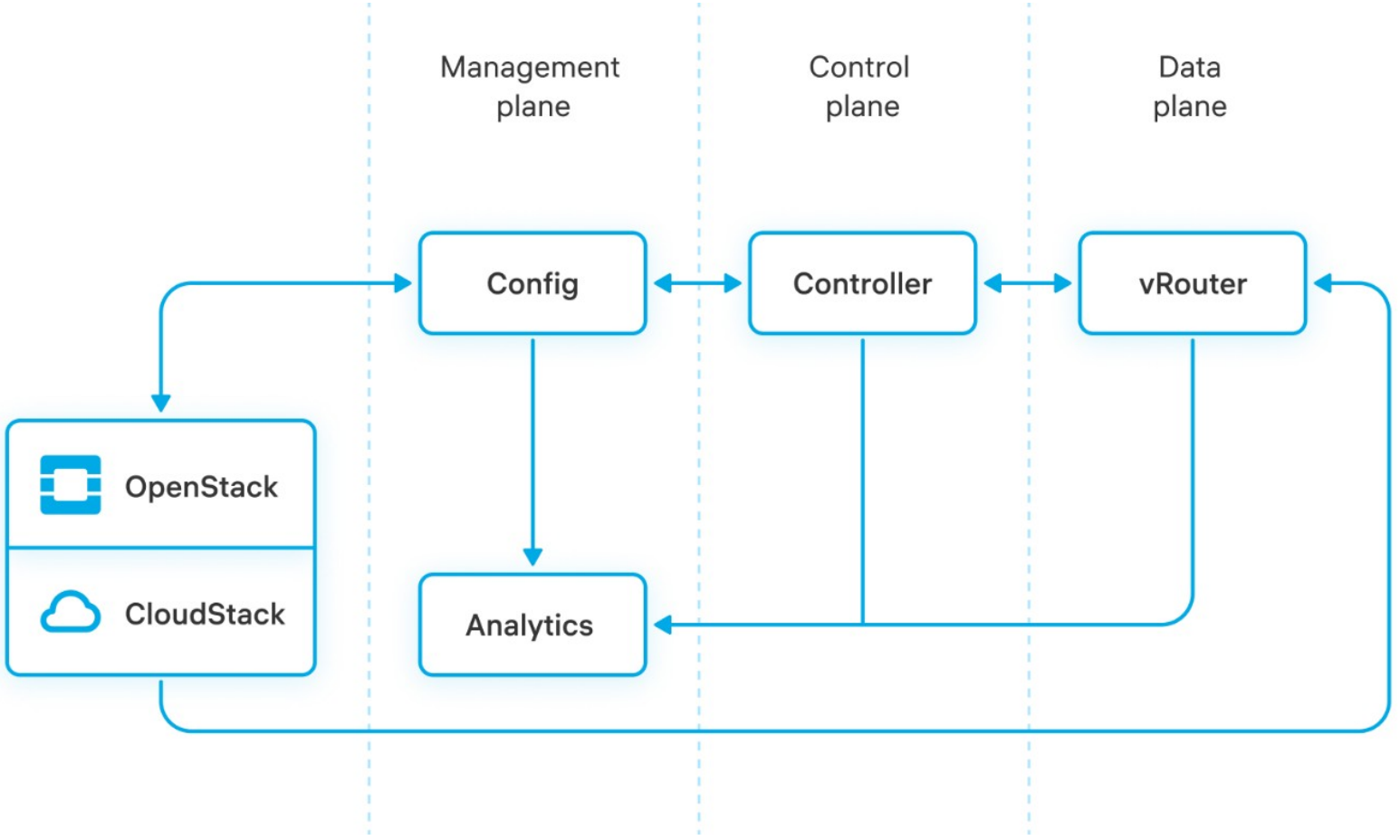
Where OpenSDN falls into

- It is physically distributed and logically centralized SDN controller
- It is partially hybrid because it can program Juniper devices
- It is more than a controller, since it includes virtual switch (unlike ODL, for example) of different types: kernel module, DPDK, user space application

Some OpenSDN controller features

- Logically centralized, physically distributed type
- Flow-based granularity: a trade-off between scalability and control over a network
- Extension of limited control logic back to forwarding devices [1] to speed-up responses for flow requests (not possible for OpenFlow![1])
- Delegates state storage, replication and management of state information to external data store (a database)

OpenSDN architecture

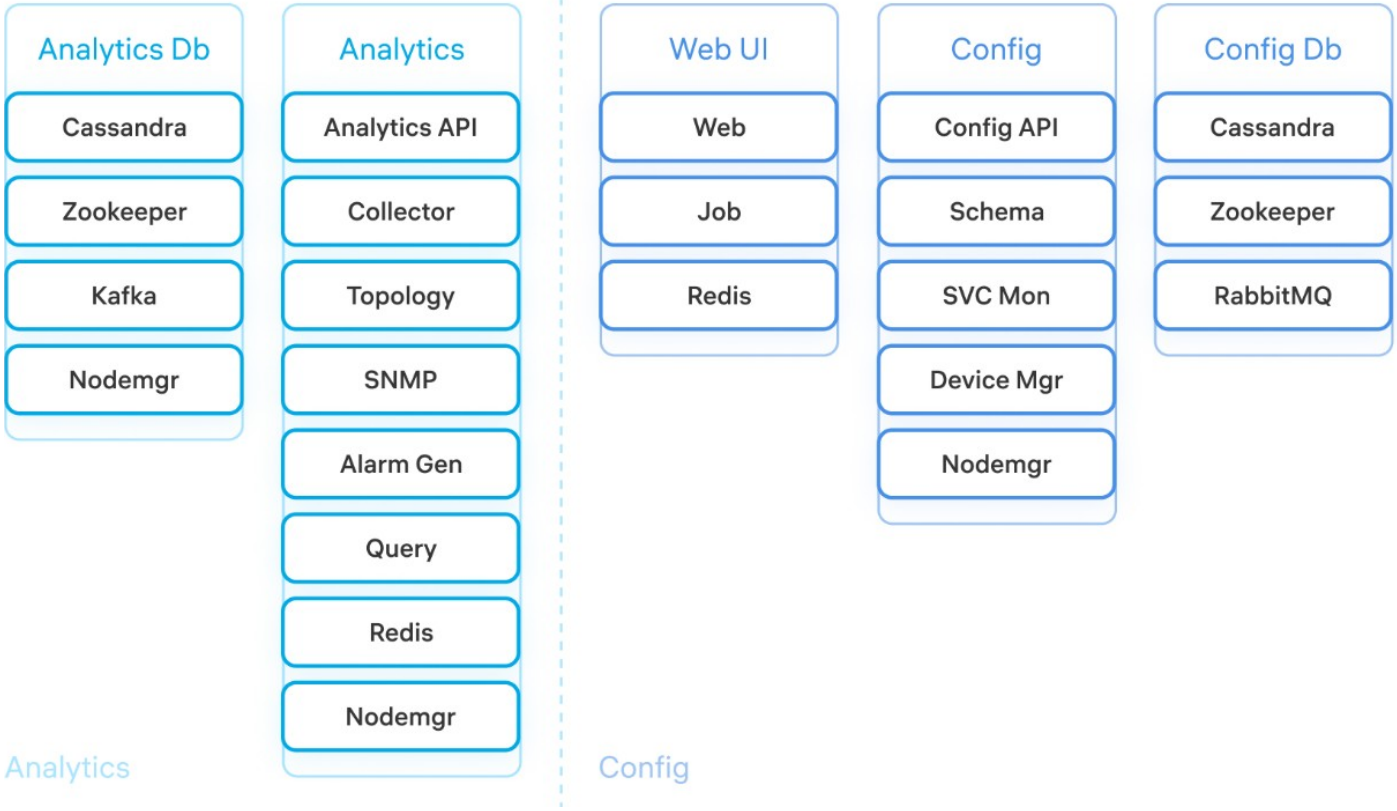


OpenSDN: a closed management cycle

- Almost all modern programs know how to forward and route packets (OVS/OVN, Calico, OpenDayLight, Cilium etc)
- However, a larger problem still persists (Zucht und Ordnung): how to keep order in computer networks generating vast number of events and characterized by huge diversity of protocols and environment and billions of connections?
- At the same time, an average person can process only 7±2 words at once
- How can we find the root reason of troubles and failures?
- Possible answer can be in an architecture with closed cycle: Config → Controller → Hypervisor (Agent /Forwarder) → Analytics → Config

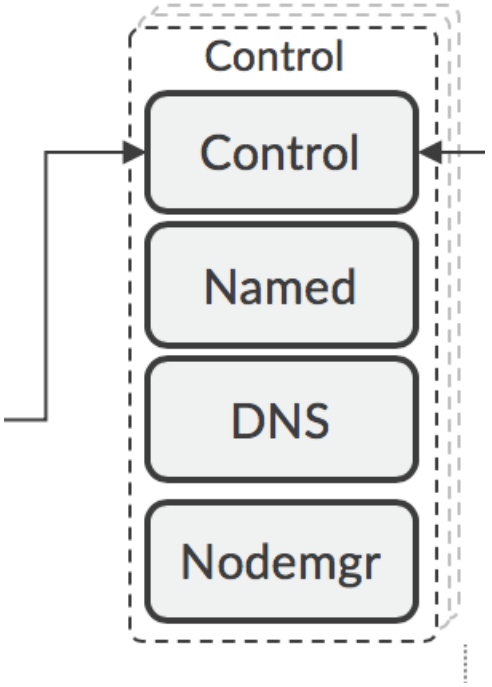
OpenSDN management plane

- Manages a virtual network configuration
- Analyzes the current state of a virtual network
- Provides a UI for the management of virtual networks



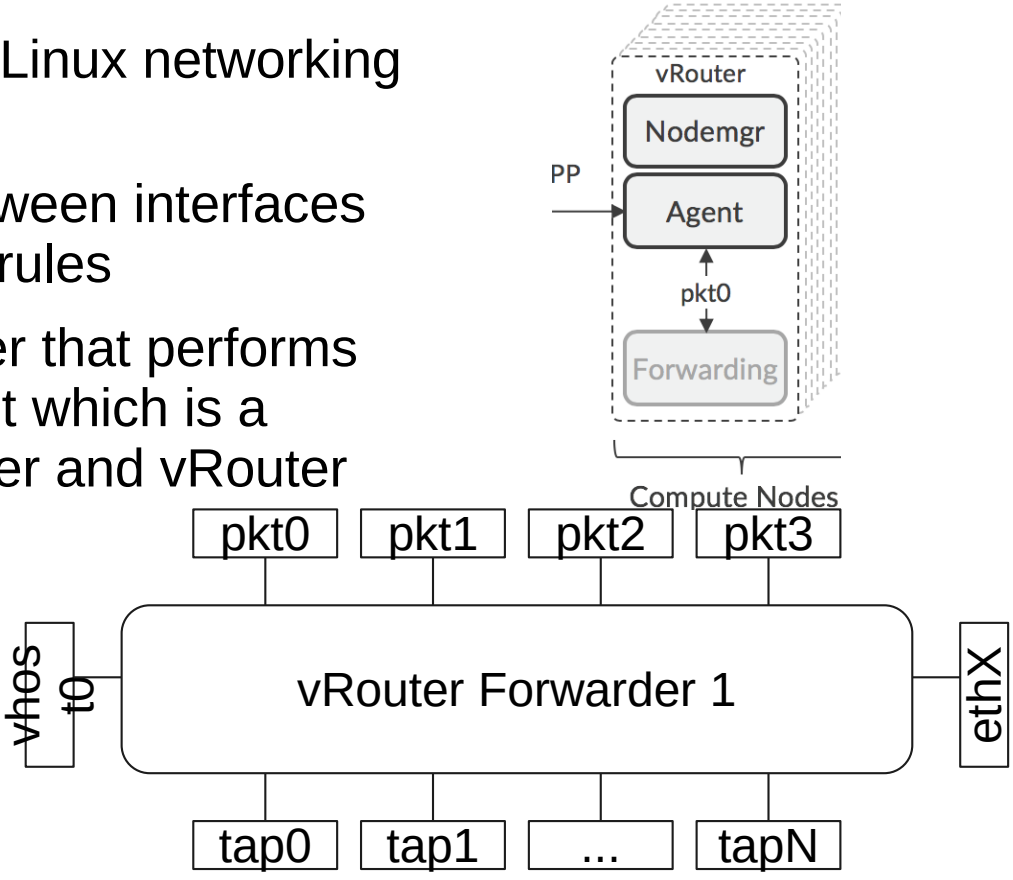
OpenSDN control plane

- Routes management
  - Interaction with external BGP routers
  - Synchronization of routes from different sources
  - Routes transfer to vRouters
- Virtual networks configuration management
  - Creation of a virtual networks representation from persistent database (Config)
  - Distribution of the representation to hypervisors (vRouter)



OpenSDN data plane

- OpenSDN vRouter replaces a part of Linux networking system (bridges and ip tables)
- The component forwards packets between interfaces (virtual machines) according to given rules
- vRouter consists of vRouter Forwarder that performs packets forwarding and vRouter Agent which is a mediator between OpenSDN Controller and vRouter Forwarder



Coding challenge

- 3 layers
- 5 components
- Dozens of applications
- How to implement all this complexity?

OpenSDN programming languages

- JS: a GUI
- Python: input data management
- C++: operative database management
- C: packets forwarding
- Apache Thrift (Sandesh): interface definition language
- And others

C vs C++ in OpenSDN

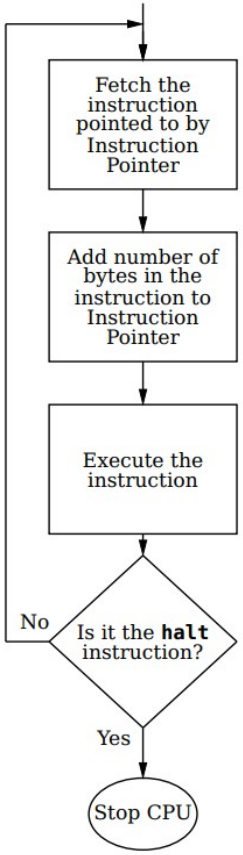
- OpenSDN uses C and C++ languages for forwarding, routing and operative configuration management layers
- Responsibility zones are distributed as followed:
  - Where high performance is demanded and simple logic is possible (this is data plane usually), then C language is used
  - When algorithms are complicated and requirements to performance can be reduced then C++ language is performed (control plane)

Advantages provided by different programming languages

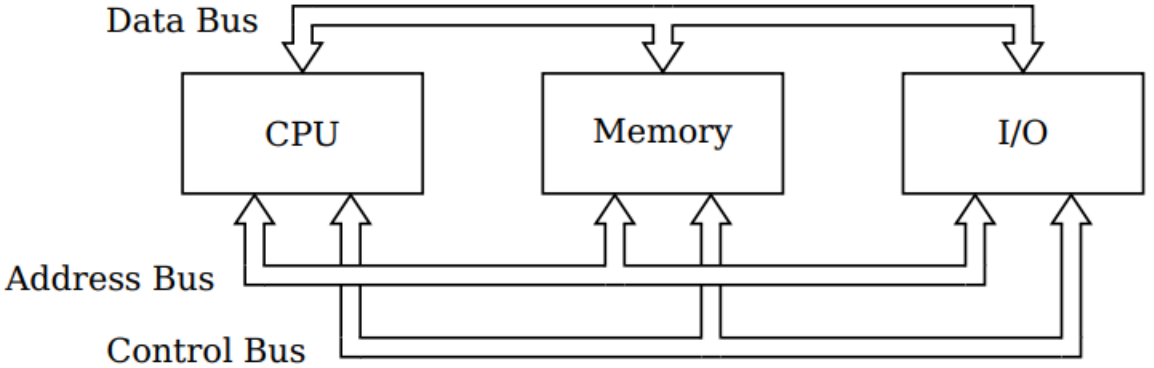
- asm: high performance and memory thrift
- C: portability and proximity to asm/hardware
- C++: extensibility and relative proximity to asm/hardware
- Java: processor architecture independence
- Python: usage flexibility, simple and concise syntax

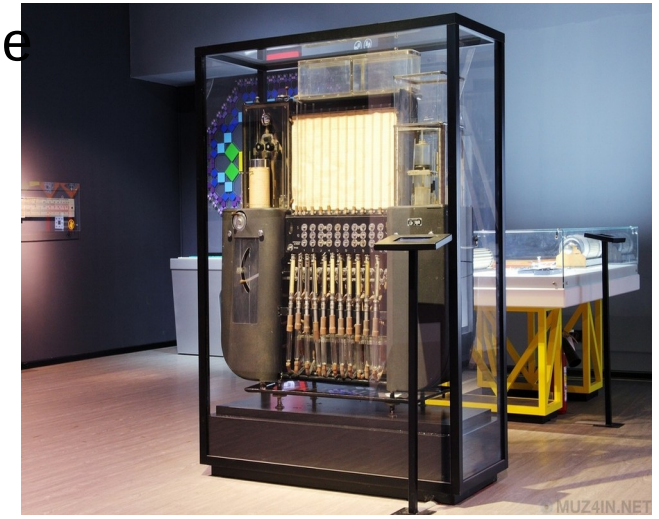
Performance vs convenience

↓ Not always ↓



- Usually, the closer a language to hardware, the higher the performance is, but this increases risk of an error and severity of consequences
- However, when a language is close to a specific domain, then programming is easier, but performance is lower





Lukyanov hydraulic integrator
(Buckingham π theorem)

https://habr.com/ru/articles/228283/

Robert G. Plantz: Introduction to Computer Organization with x86-64 Assembly Language & GNU/Linux

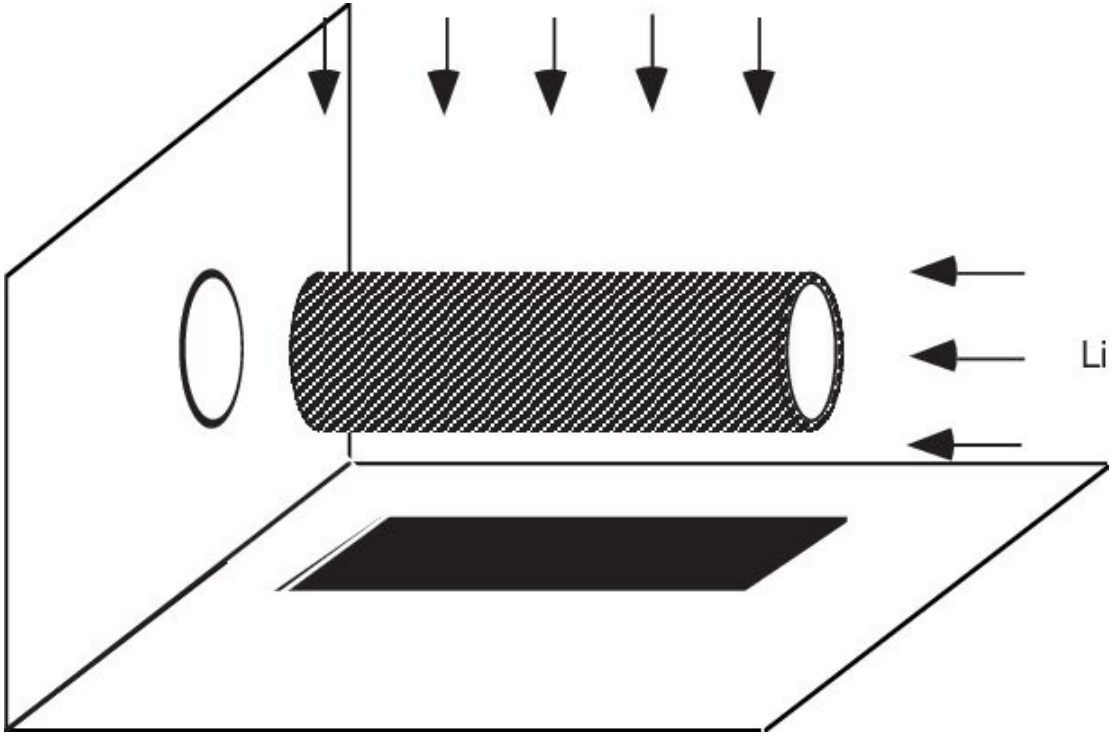
Programming languages types

- Machine code
- Assembler
- General purpose
  - Compiled
    - C/C++/Go/Rust/Java/Fortran
  - Interpreters
    - Python, bash, Julia
- Domain-specific:
  - RELAP5/MOD3, LaTeX
- Declarative:
  - Logic programming (Prolog)
  - Functional programming (Lisp, C++)
  - DSL (LaTeX, SQL, C++)
- Imperative:
  - Procedural (Fortran, C++)
  - Object-oriented (C++, Java)

OpenSDN roles

- A compiler
- A network operating system
- A virtual router

All these roles are actually levels



DOI: http://dx.doi.org/10.1080/1080703989128482O

OpenSDN as a compiler

- According to Contrail Architecture, Chapter “Scale-Out Architecture and High Availability”
- This SDN can be deemed as a compiler
- That compiles a high level data model into the low-level one

Typical compilation stages

- Pre-processor → modified code without (# directives)
- Compiler (translation) → asm code
- Assembler → machine code (object files)
- Linker (composer) → a program (machine code with all links)

Data model

- A data model consists of a set of objects *definitions*, their capabilities, and the relationships between them
- A data model can be: **high level** (external input) and **low level** (generated internally)
- A data model yields a state, which can be of **configuration** or **operative*** type

*While Contrail documentation uses term “operational”, it seems that “operative” is closer to the proper meaning according to the Oxford dictionary

States

- **A configuration state** is a set of objects and relations between the the described according to a **high level data model** (intended state, what we want)
- **An operative state** is a set of objects corresponding to the **low level data model** (what we have now in the virtual network)
- OOP analogies: a data model → a class, a state → an instance

Traits of OpenSDN as a programming language

- Allegedly, it is a declarative programming language
- Most likely it is an interpreter
- Provides instruments to define virtual networks (so, it is a DSL)
- Programs virtual switchers
- Provides tools for failures diagnostics

OpenSDN as a network OS

- NOS provides a uniform and **centralized programmatic interface** to the entire network without managing the network itself
- The network is managed by special programs written in programs are written in **terms of high-level abstractions**
- Natasha Gude et al, NOX: Towards an operating system for networks // ACM SIGCOMM Computer Communication Review, 38(3):105-110, 2008
- More than 1k citations

NOS: geeksforgeeks.org

- https://www.geeksforgeeks.org/what-is-a-network-operating-system/

Functions of the NOS (Network Operating System)

The following are the main functions of NOS:

- Creating and managing user accounts on the network.
- Controlling access to resources on the network.
- Provide communication services between the devices on the network.
- Monitor and troubleshoot the network.
- Configuring and Managing the resources on the network.

What does OpenSDN provide as a NOS

- Allocation and destruction of network resources (IP, MAC, etc)
- Support for network addressing and routing
- Access management
- Packets forwarding
- Provides a platform for network function virtualization
- Provides tools for state analysis

if it looks like a duck, walks like a duck, and quacks like a duck, then it's probably a duck

OpenSDN as a virtual router

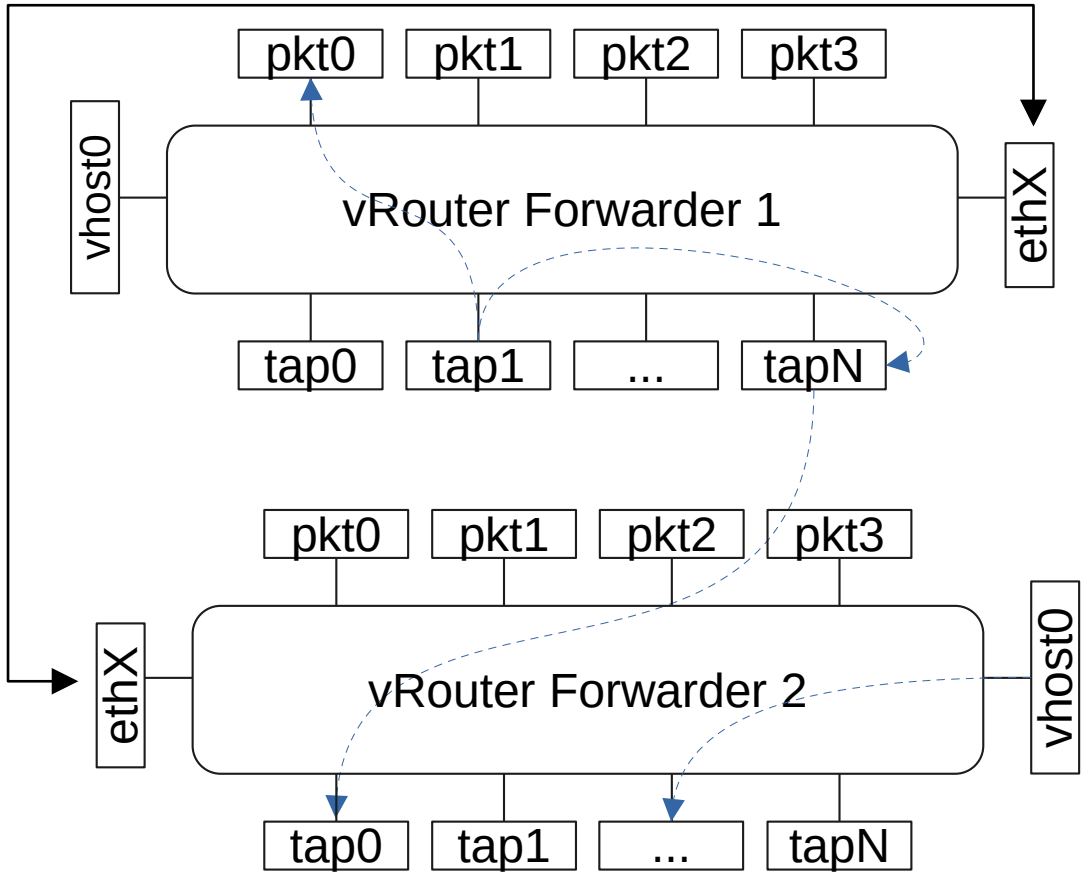
- OpenSDN has vRouter Forwarder with L2 and L3 packets switching
- It has a BGP router to exchange routes with other routers
- Essentially, it is a virtual router on top of a NOS

Main data streams in OpenSDN

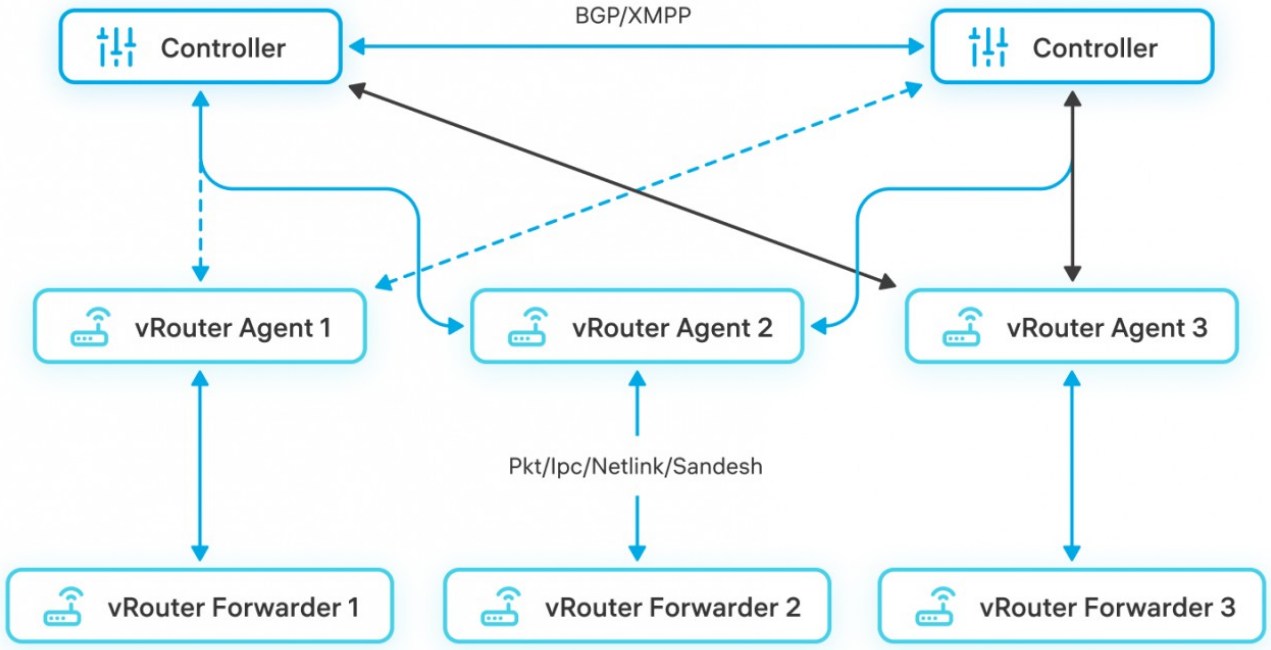
- A packets data stream
- A routes data stream
- A configuration data stream

The packets data stream

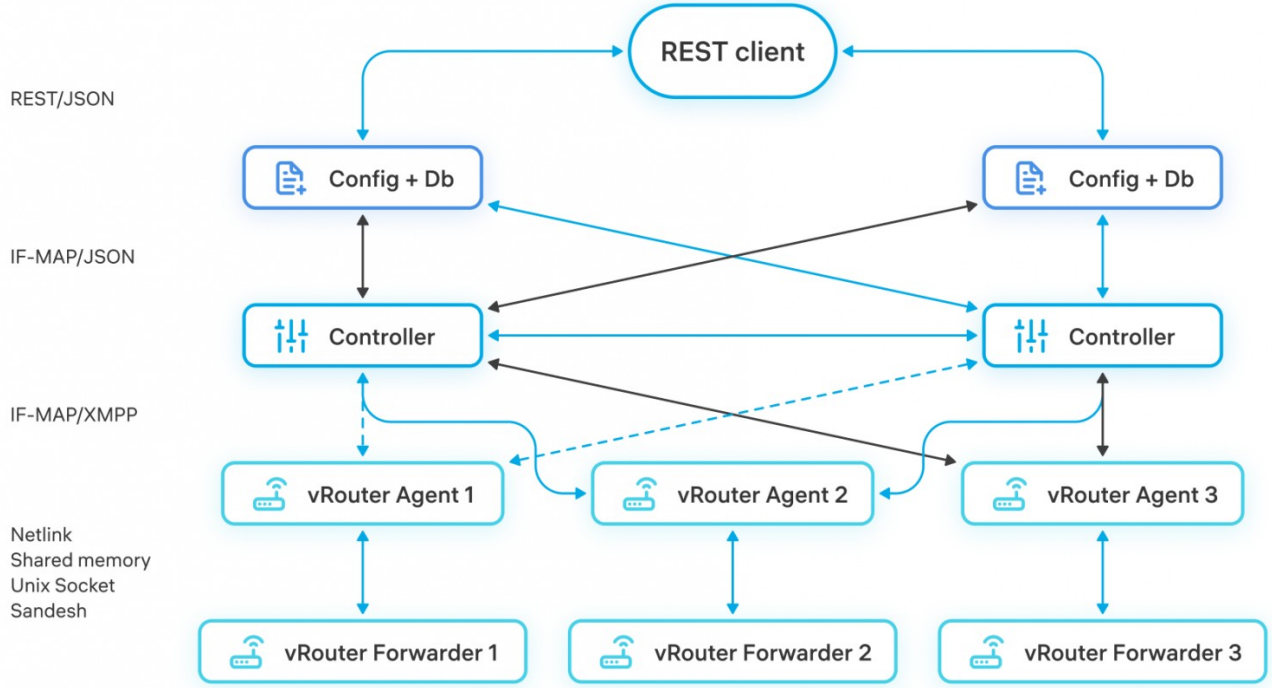
- Packets go between TAP-interfaces of a hypervisor, between hypervisors (via eth), between host OS and TAP-interfaces, between TAP-interfaces and pkt interfaces



The routes data stream

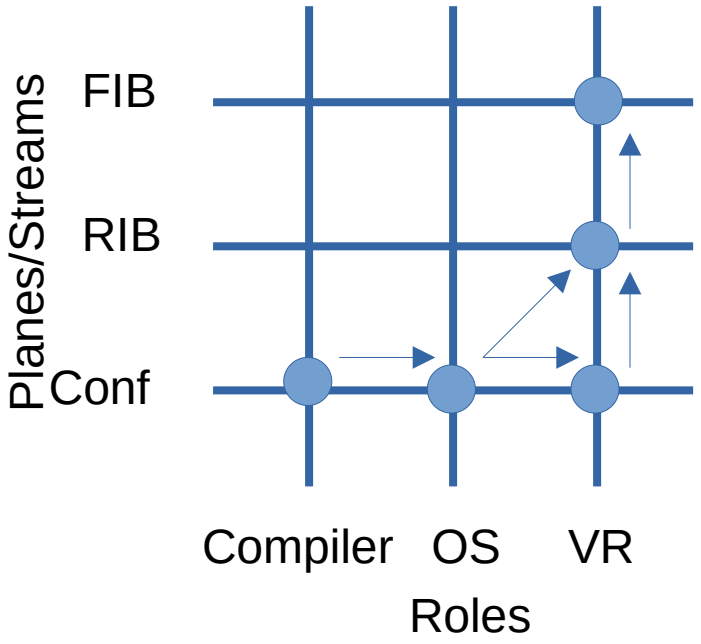


The configuration data stream



Roles + Data Streams = OpenSDN

- Compiler + high level data model => virtual networks resources (low level data model)
- OS + low level data model => virtual networks resources management + RIB
- Virtual router + low level data model => RIB
- Virtual router + RIB => packets forwarding



How these tools are linked and implemented?

- What technologies are used in OpenSDN?
- What concepts, data structures and algorithms are employed?
- How different parts are linked together?

Types of REST messages

- Create
  - Creates a new element, e.g. a virtual network, a virtual machine interface, etc
- Read
  - Reads the state of a specified element
- Update
  - Updates the state of a specified element
- Delete
  - Deletes a specified element
- Link
  - Creates a link between two elements (for example, between a virtual machine interface and an IP instance)

Representational State Transfer

http://controller-ip:8082/documentation/index.html

REST request structure

- Command: usually curl
- Type is one of: POST/GET/PUT/DELETE
- Header
- Request body
- URL

<Command> <Type> <Header> <Request body> <URL>

Where requests are stored?

- Persistent DB
  - Employed for long-term storage of a virtual network state (the relationships graph of network entities
  - Cassandra DB (OpenSDN, Netflix, Instagram)
- Operative storage or DB
  - Contains the instant state of a virtual networks and it's participants, an is a derivative data w.r.t. the persistent DB

Logically monolithic application

- But physically distributed
  - Controller
  - DNS server
  - Config API
  - And perhaps others
- This implies that operative DB synchronizes not only between threads, but even between processes located on physically different computational nodes

Cassandra

- Key – value database
- Uses Cassandra Query Language (similar to SQL)
- Persistent storage: writes are fast, read are slow

```
cqlsh:config_db_uuid> SELECT blobAsText(key) AS key, blobAsText(column1) AS value FROM config_db_uuid.obj_fq_name_table WHERE

 key         | value
-------------+-----------------------------------------------------------------------------------------------------
 floating_ip | default-domain:admin:V-net:default:ff5d39b0-1f2d-4986-9343-b20116070fc9:ff5d39b0-1f2d-4986-9343-b20116070fc9
```

```
cqlsh:config_db_uuid> SELECT blobAsText(key) as key, blobAsText(column1) as value FROM obj_uuid_table WHERE k

 key                                  | value
--------------------------------------+-----------------------------------------------------------------
 ff5d39b0-1f2d-4986-9343-b20116070fc9 |                                                  META:latest_col_ts
 ff5d39b0-1f2d-4986-9343-b20116070fc9 |                                                            fq_name
 ff5d39b0-1f2d-4986-9343-b20116070fc9 |       parent:floating_ip_pool:bca266ba-8ff8-4779-a7b6-e2551f4c18bf
 ff5d39b0-1f2d-4986-9343-b20116070fc9 |                                                       parent_type
 ff5d39b0-1f2d-4986-9343-b20116070fc9 |                                                  prop:display_name
```

IF MAP DB

- IF MAP DB is an instant representation of low-level data model instance (operative state)
- Applications listen to Cassandra DB
- When a change happens, it is processed as a JSON and stored in IF-MAP Graph DB:
  - Objects as vertices
  - References as edges

https://trustedcomputinggroup.org/resource/overview-of-trusted-network-connect-tnc-if-map/

Operative DB (OperDB)

- OperDB is stored in RAM and created from low-level data model
- Consists of:
  - Tables store elements of a same type (e.g., ports, flows)
  - Partitions are used to separate a table between CPUs
  - Entries represent single units of information stored in tables
  - States are additional information added optionally to entries
  - Requests are objects to manipulate entries

DB Graph

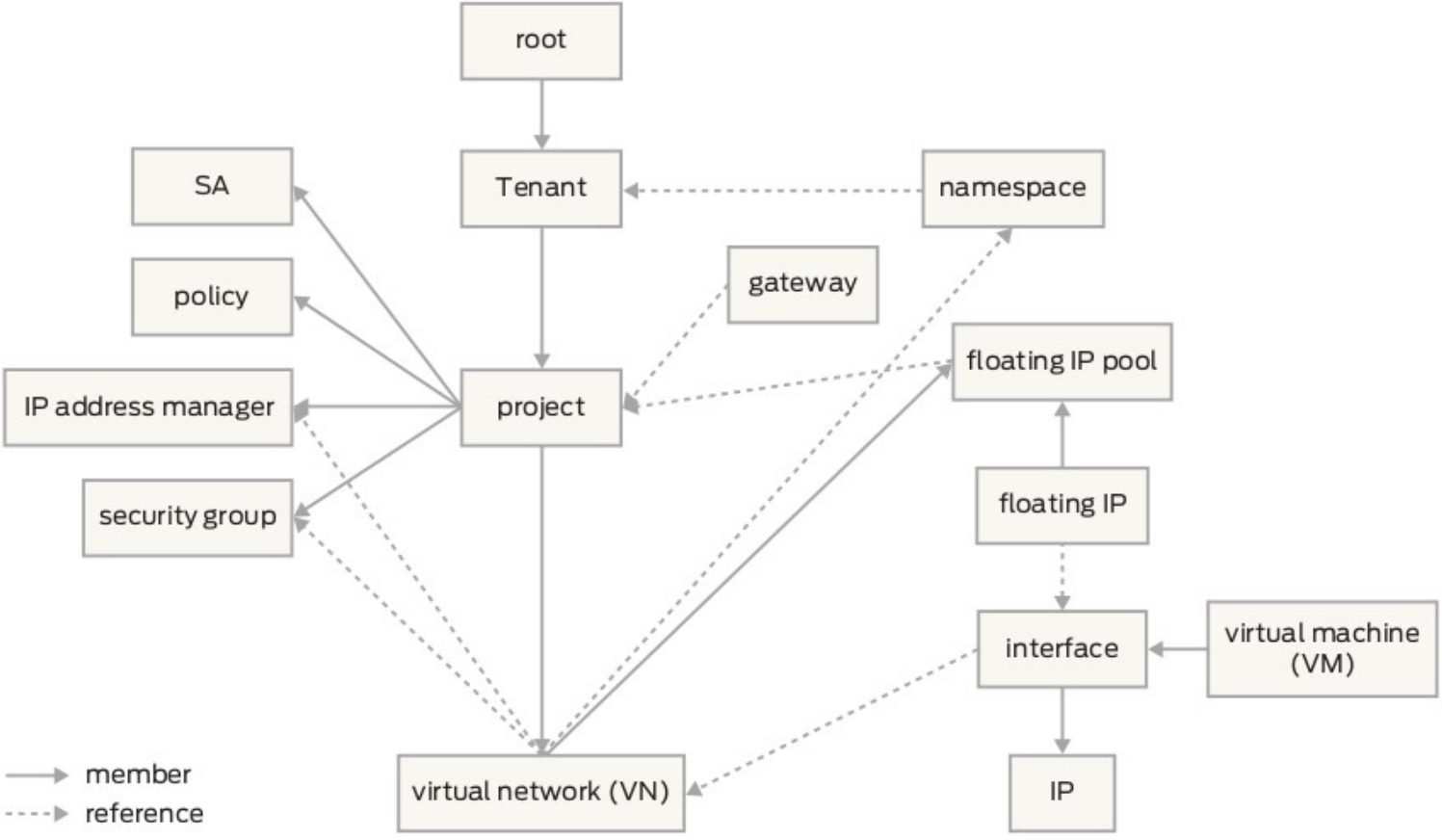
- Represents the operative state as a graph
- Relates IF-MAP records to vertices and edges
- Handles deletion and addition of new vertices and edges, search and walks

Graphs in OpenSDN

- The data model graph
- An objects relationships graph
- Configuration propagation graphs
- Operative DB graph (red-black trees), tries, etc
- State machines

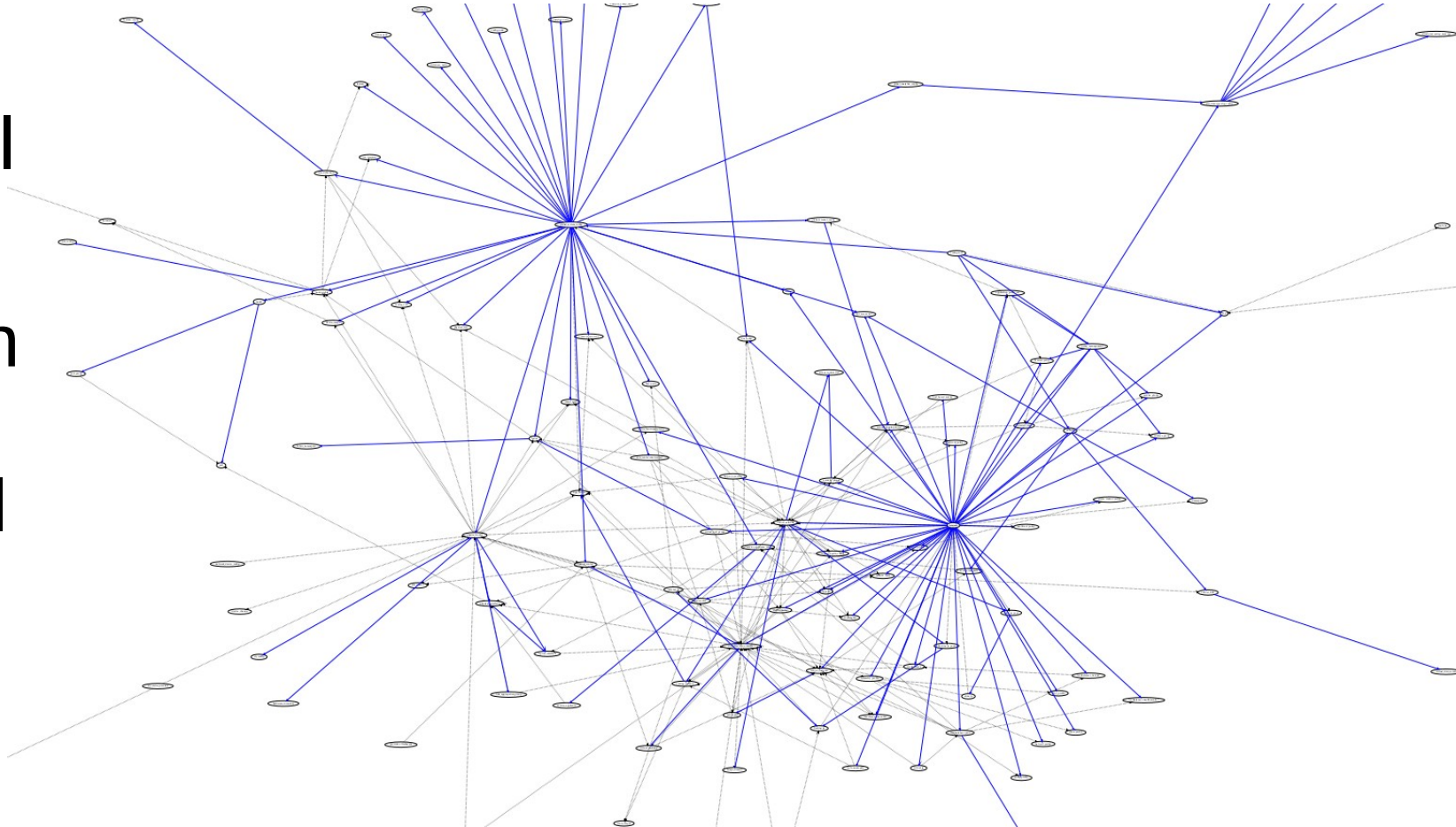
Data model graph

- Directed Acyclic Graph (polytree)
- Simplified version



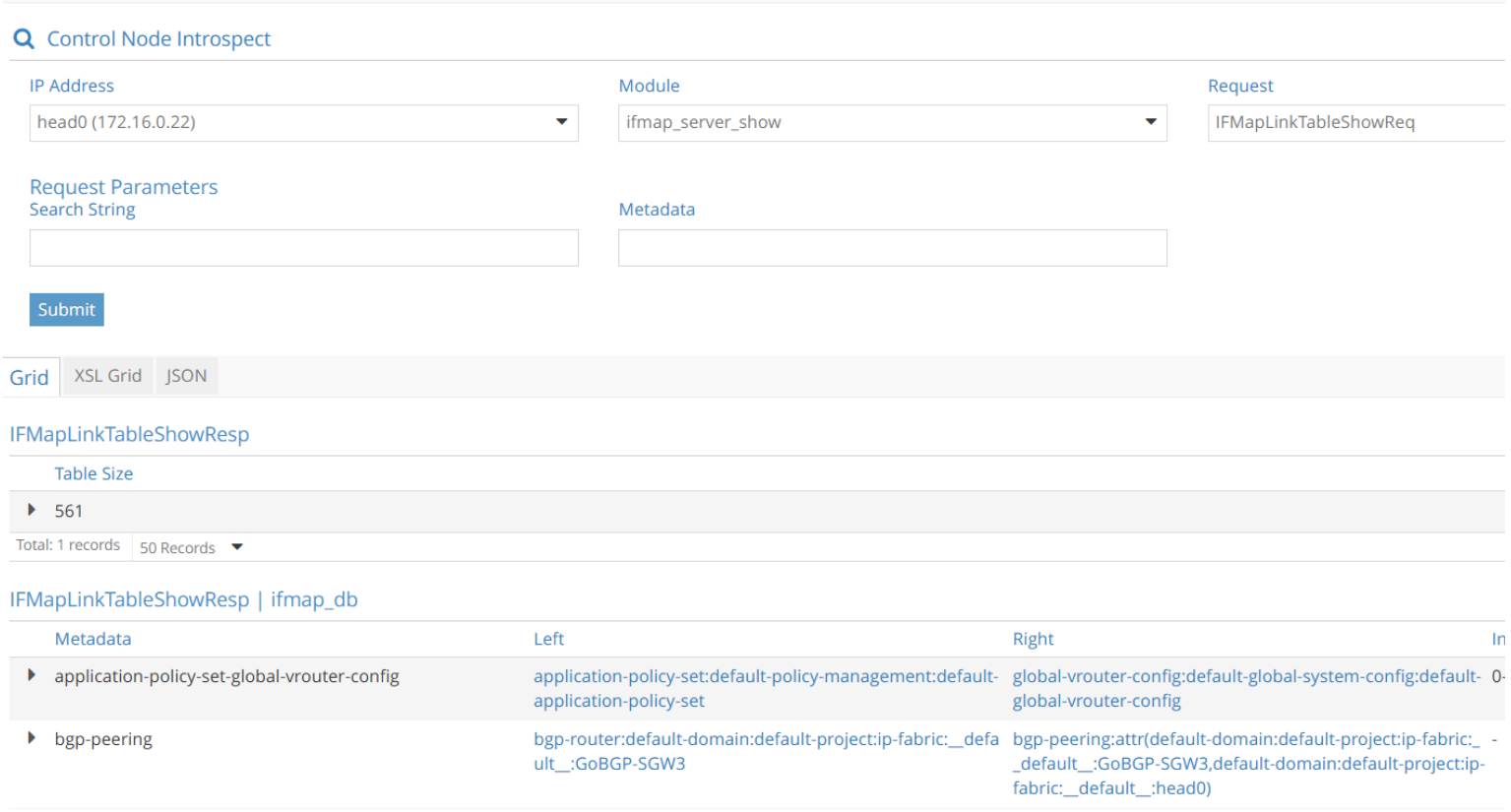
Data model graph (full)

- Includes almost all nodes
- Has been built from .yml files



Objects relationships graph

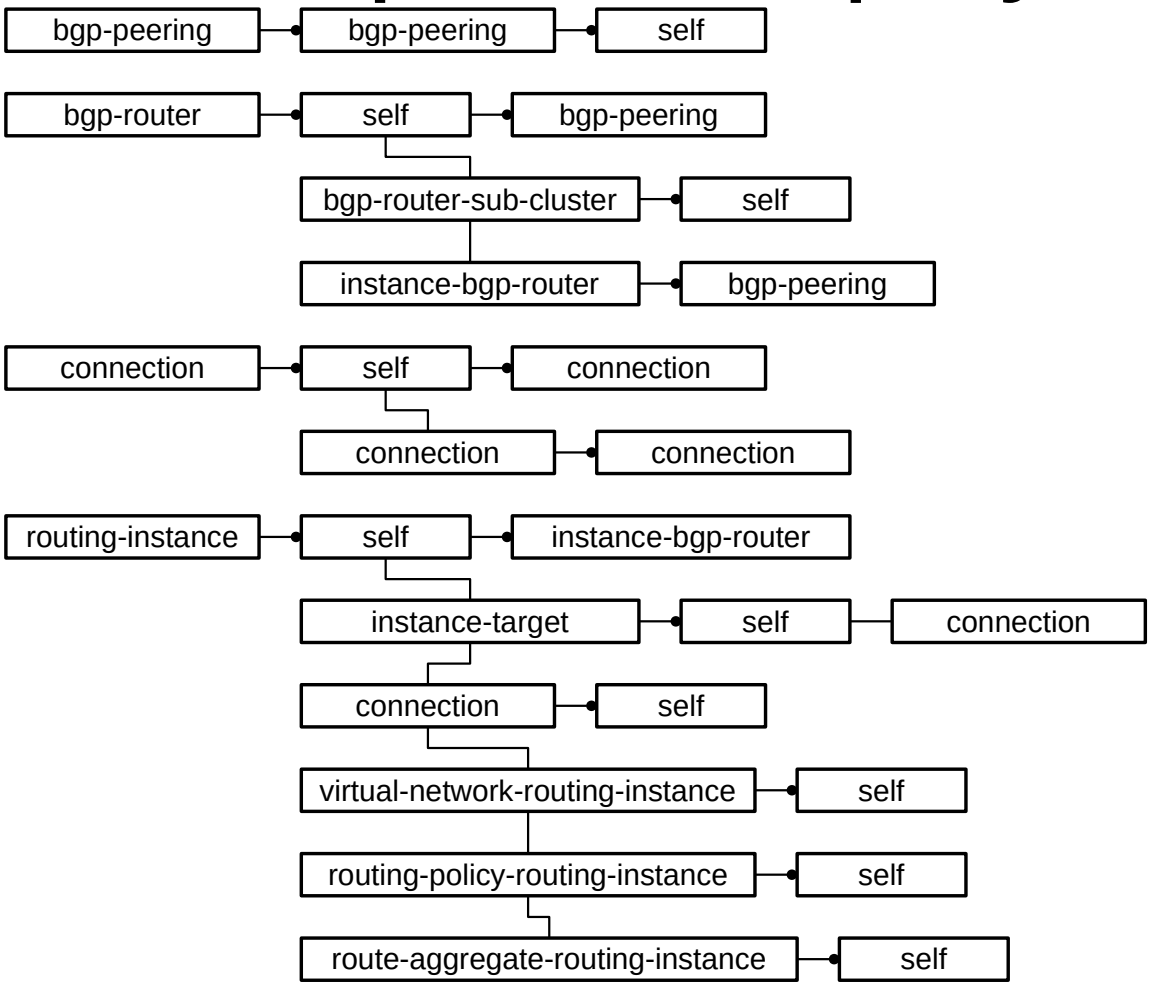
- DAG (polytree)
- Can be viewed via Introspect UI



Configuration propagation graph(s)

- Propagation of changes (e.g., inside controllers)
- Graph cuts construction (e.g., controller to agent)
- Disconnected acyclic graph (polyforest)

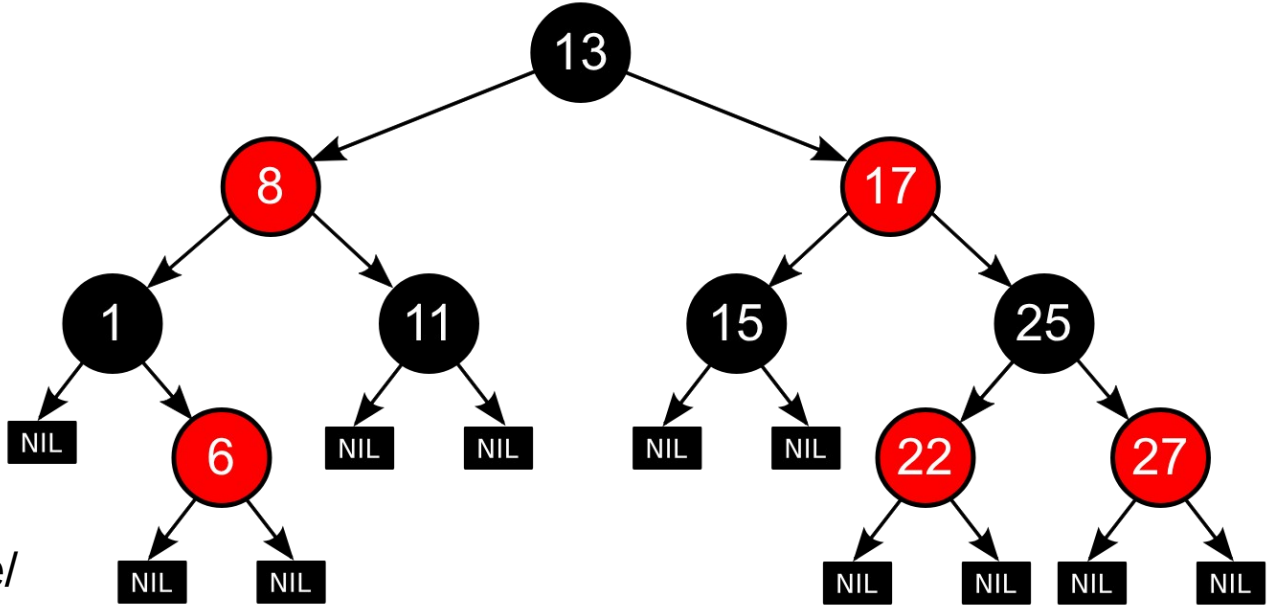
OpenSDN polyforest example



- Propagation of changes polyforest

Red-black tree

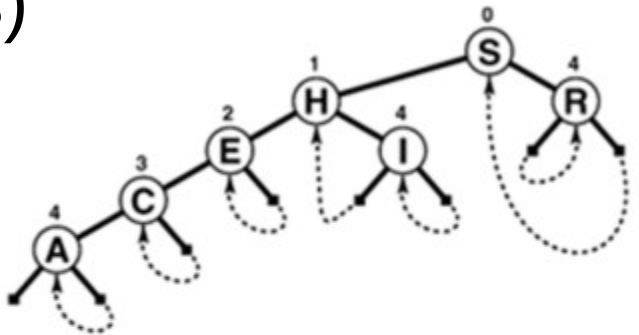
- Is used to store various object in OpenSDN tables (DBTable class), boost::intrusive::set, incl. RIB in controller, IF-MAP, etc

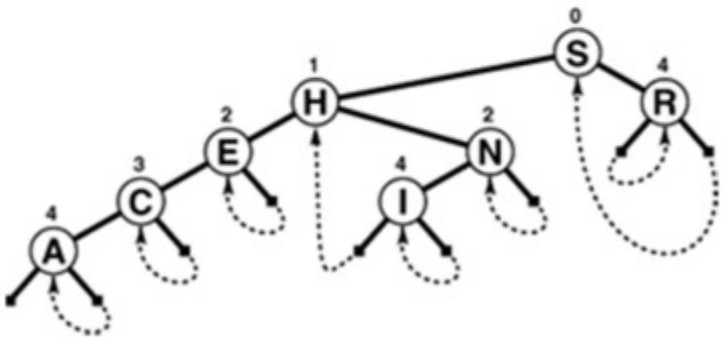


https://brilliant.org/wiki/red-black-tree/

PATRICIA Trie

- Is used to store INET Unicast routes tree in an Agent process (due to it's compactness)
- Performs LPM



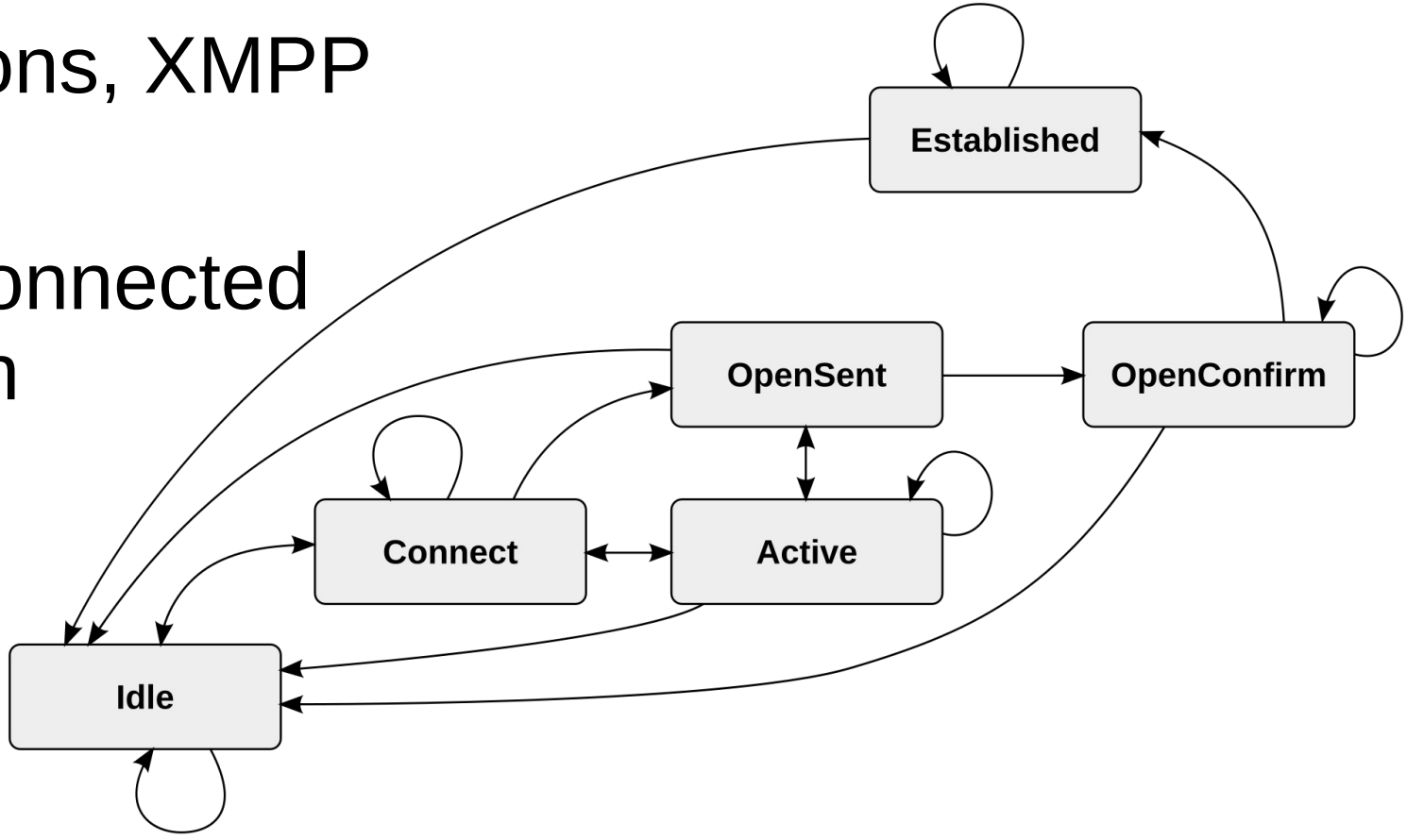


Donald R. Morrison: PATRICIA—Practical Algorithm To Retrieve Information Coded in Alphanumeric, Journal of the ACM, Volume 15 Issue 4, Oct. 1968, pp 514-534.

https://cw.fel.cvut.cz/b181/_media/courses/b4m33pal/paska13trie.pdf

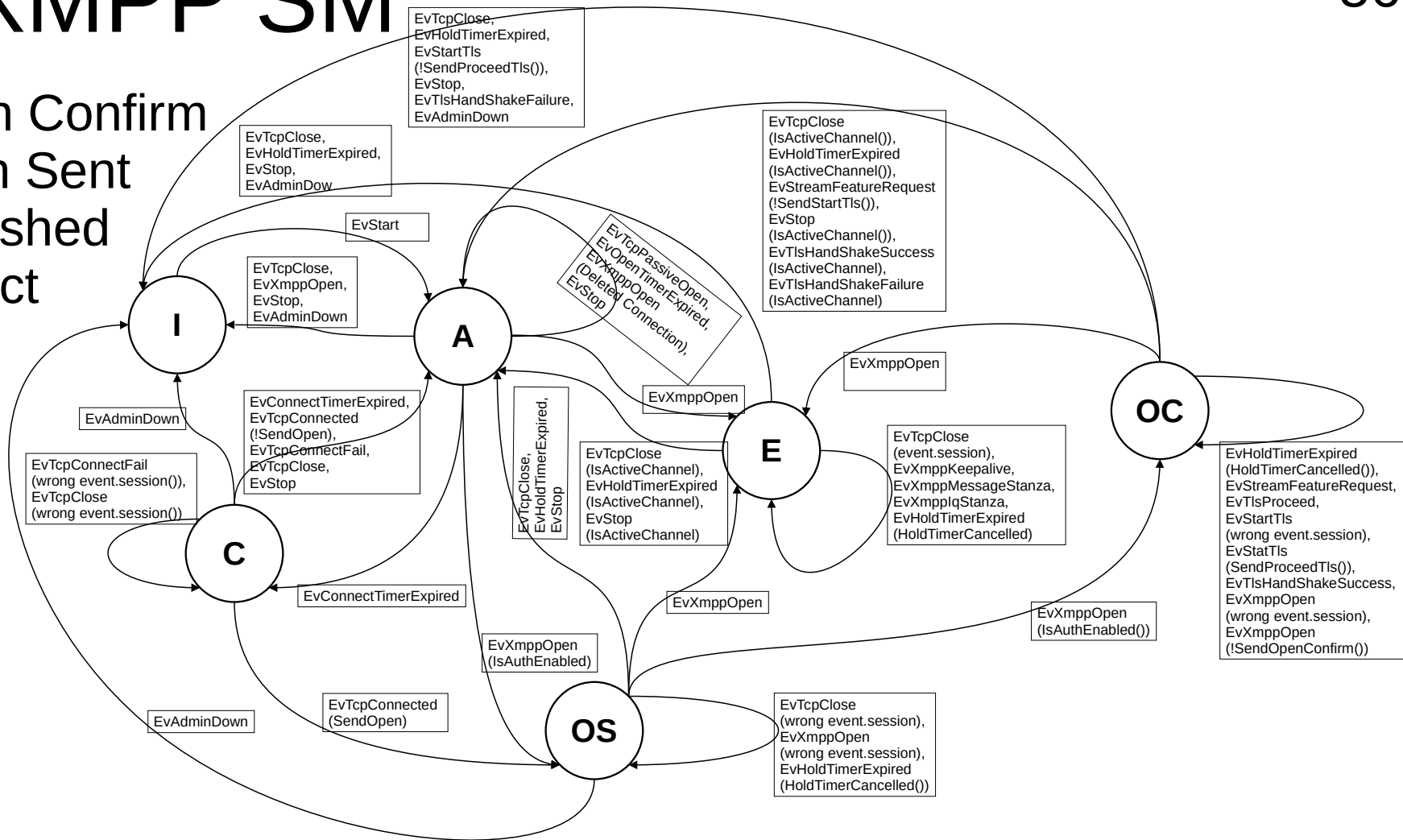
State machines

- BGP sessions, XMPP sessions
- Directed, connected cyclic graph



Full XMPP SM

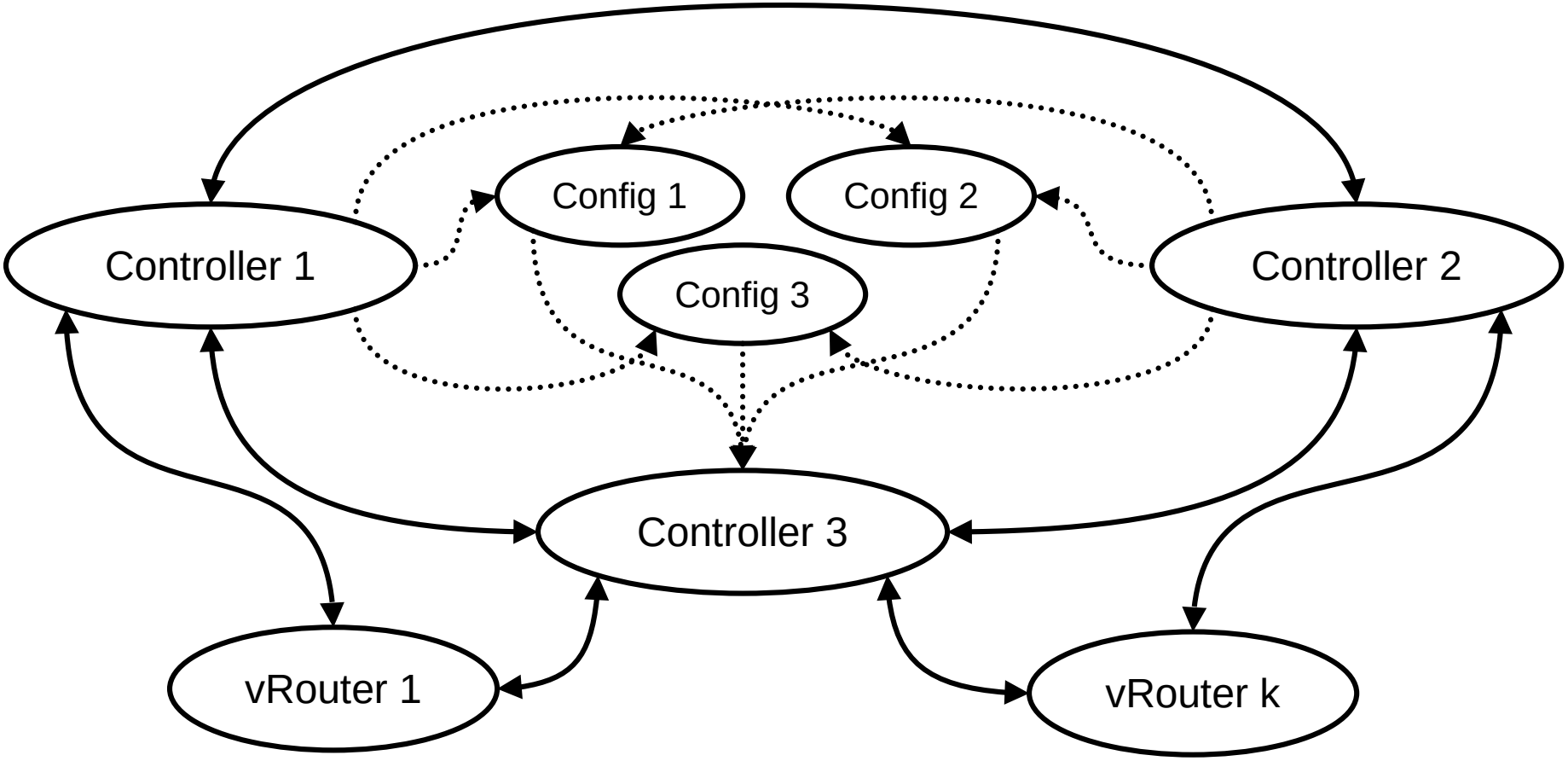
OC — Open Confirm
OS — Open Sent
E — Established
C — Connect
A — Active
I — Idle



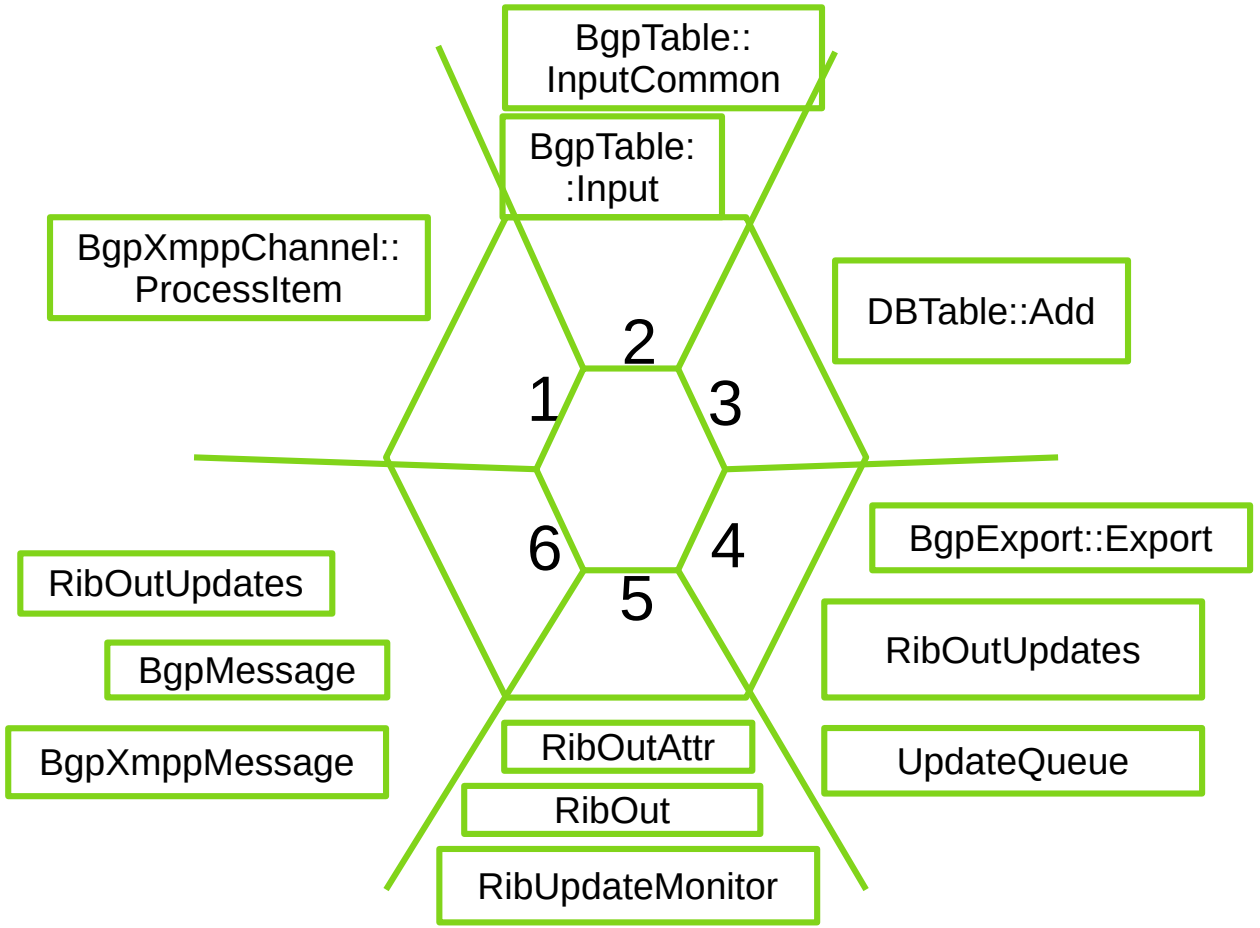
Controller

- Central point for routing information base (RIB)
- Central point of IF-MAP operative data (low level data model) for OpenSDN vRouters
- Logically centralized and physically distributed application
- Works as a pipeline: reads persistent DB and creates/updates the operative state

Connections with other modules



The pipeline stages and RIB classes



vRouter Agent main parts

- Databases (tables and graphs): IF-MAP + operative
- Servers: Metadata proxy, Introspect, etc
- Services (DNS, DHCP, ICMP, etc)
- vRouter Forwarder synchronization

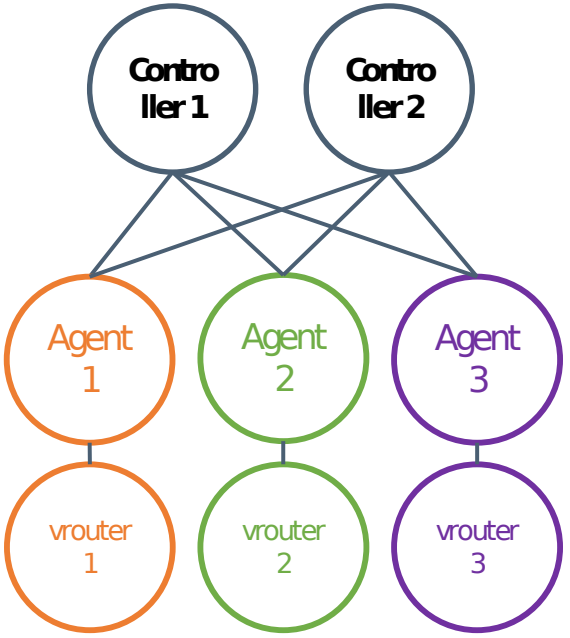
OpenSDN RIB elements

- Virtual Network (VN)
- Virtual Routing & Forwarding (VRF)
- Route tables (EVPN & INET)
- Route
- Prefix:
  - Prefix address – L3 (IP) and L2 (MAC)
  - Prefix length
- Path
  - Nexthop
  - Peer

- A **peer** together with a **nexthop** make up the **path**.
- List of **path**s make up a **route**. Combination of a **peer** type and a nexthop is unique within a **route**.
- **Route**s make up a **route table**.
- **Route table**s make up a **VRF instance**.
- **VRF instance** with **Virtual Network** makes up representation of a network

OpenSDN RIB exchange

Each Agent is responsible for storing of the part of the Controller's table. There is a part corresponding to local VM interfaces. The remaining part is stored as tunnels.



Routes leaking

- Routes leaking is a procedure of routes synchronization between two tables according to some predefined rules:
  - Introduction of new routes
  - Modification of existing routes
  - Deletion of obsolete routes
- This procedure is a key concept of OpenSDN VxLAN control plane implementation

Routes leaking example

Table1

Prefix1

Path1: Nexthop1 Peer1
Path2: Nexthop2 Peer2

Prefix3

Path3: Nexthop3 Peer1
Path4: Nexthop4 Peer2

Table2

Prefix5

Path5: Nexthop5 Peer1
Path6: Nexthop6 Peer2

Prefix7

Path7: Nexthop7 Peer1
Path8: Nexthop8 Peer2

Table3

Prefix1

Path1: Nexthop1 Peer1

Prefix3

Path3: Nexthop3 Peer1

Prefix5

Path5: Nexthop5 Peer1

Prefix7

Path7: Nexthop7 Peer1

Routes can be synchronized between tables in one VRF instance or between VRF instances.

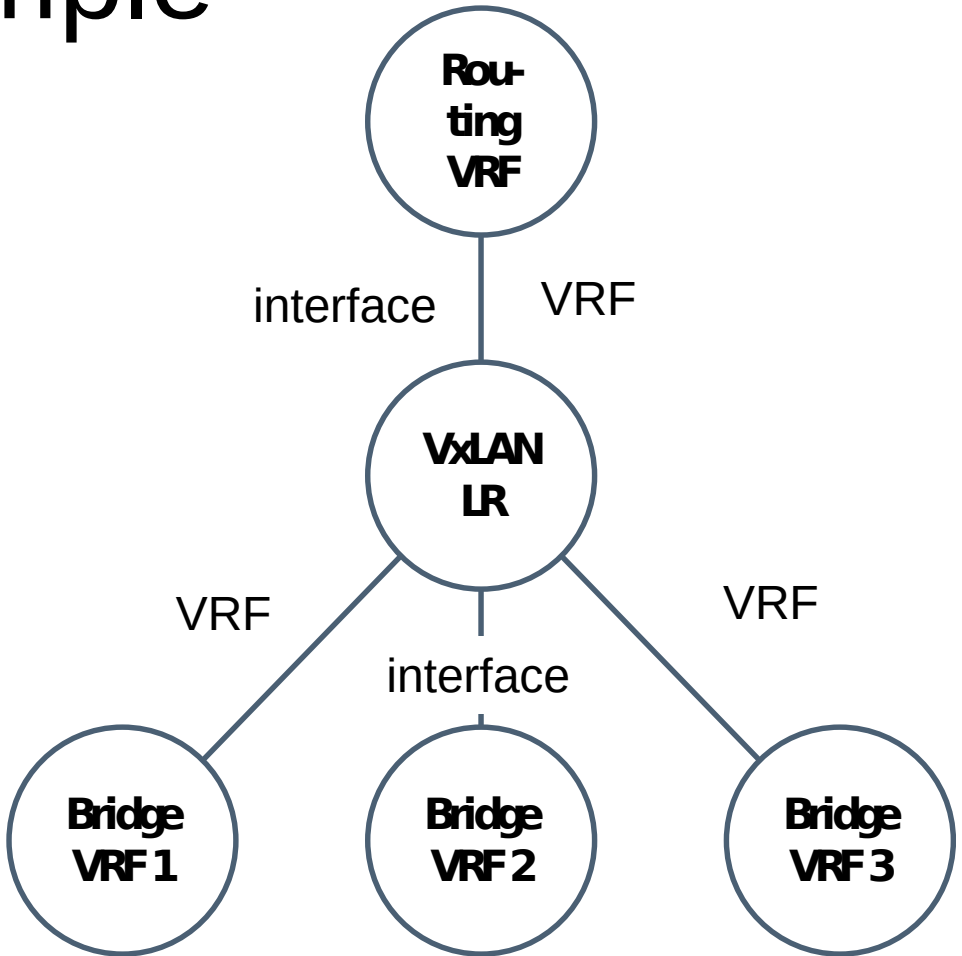
VxLAN: an example of routes leaking

- VxLAN (RFC 7348):
  - Data plane: VxLAN header in UDP
  - Control plane:flood-and-learn, MP BGP, etc
- OpenSDN VxLAN: a special type of logical router (VxLAN LR), which may be similar to “central authority/BGP” approach from RFC 7348

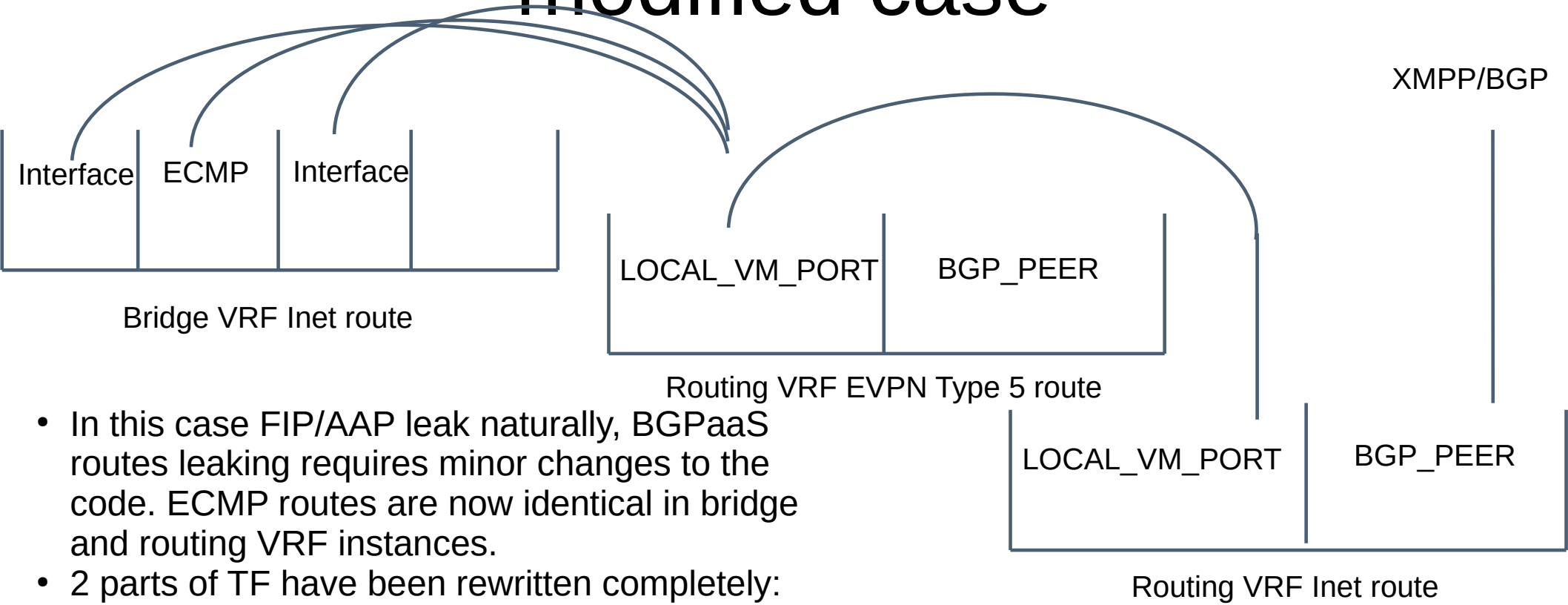
https://datatracker.ietf.org/doc/html/rfc7348

VxLAN routes leaking in OpenSDN: an example

- There are 3 routing-bridge pairs of VRF instances:
  - bridge1-routing
  - bridge2-routing
  - bridge3-routing
- When routes are copied from each bridge VRF into the routing VRF, this creates special nexthops (VRF type) in other bridge VRF instances: the references to leaked routes

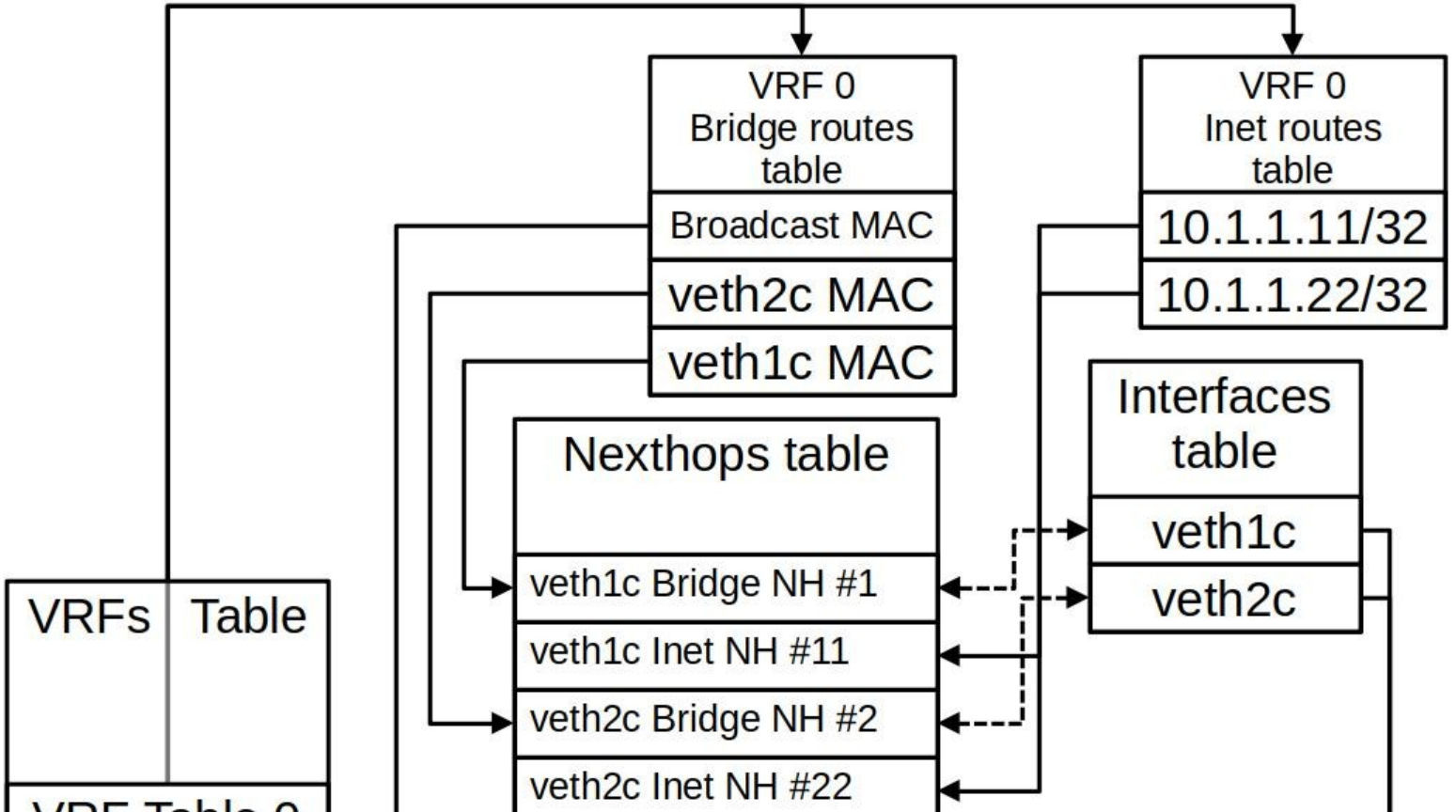


Analogy with baskets for the modified case



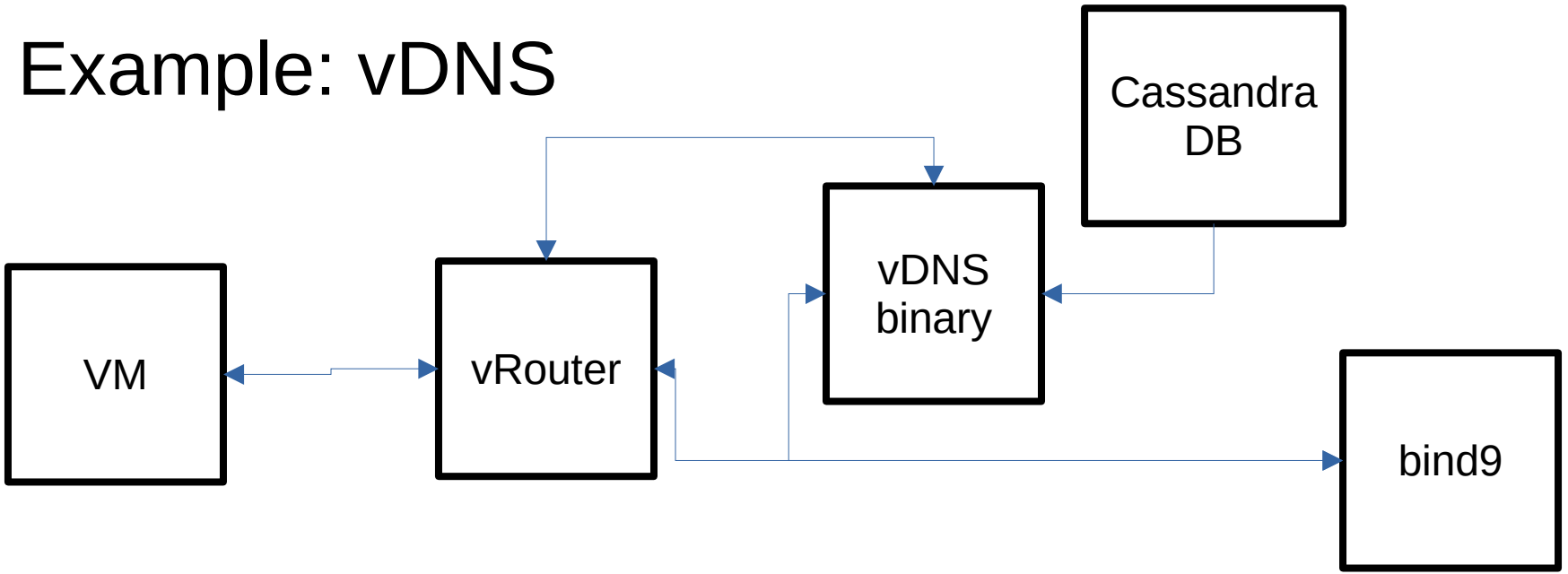
- In this case FIP/AAP leak naturally, BGPaaS routes leaking requires minor changes to the code. ECMP routes are now identical in bridge and routing VRF instances.
- 2 parts of TF have been rewritten completely:
  - Leaking of LOCAL_VM_PORT routes
  - BGP/XMPP routes input

vRouter Forwarder FIB



Virtual Networks Apps: vDNS

- There are other applications providing network services for tenants
- Example: vDNS



Sandesh/Apache Thrift: How everything is glued

- Uses and extends Apache Thrift
- Uses Flex and Bison
- Creates declarations, definitions and auxiliary implementations from IDL definitions
- OpenSDN uses Sandesh to communicate between modules (vRouter Agent and vRouter Forwarder, contrail-tools, etc), to create and dispatch introspection information, for UVE, debug and

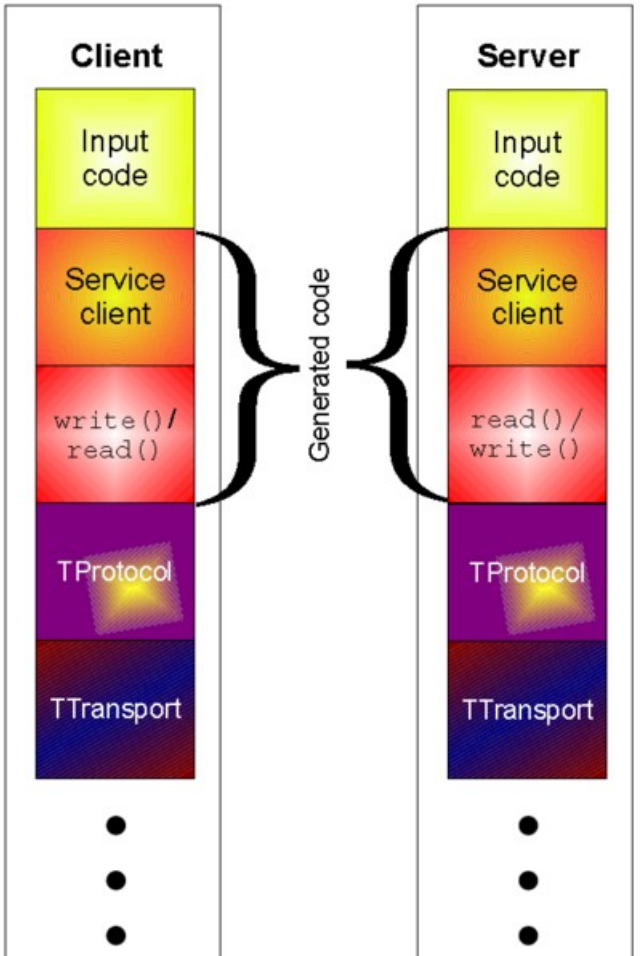
Apache Thrift

https://thrift-tutorial.readthedocs.io/en/latest/intro.html

- Supports C/C++/python/Java/Go and other languages
- Literature:
  - Programmers guide to Apache Thrift by R. Abernethy
  - readthedocs

Apache Thrift Basics

- A user defines interfaces of interaction between components (IDL file with extension **.thrift**)
- Selects:
  - Socket type
  - Transport type
  - Protocol type
- Writes implementation code for



Some Apache Thrift terms

- Socket – a marker which allows to identify remote part of a conversation, i.e. identifies the receiver and the sender of single communication
- Transport – a mechanism of communication between parts of the conversation (memory, TCP, file, etc)
- Protocol – how messages of the conversation are stored (binary, JSON, etc)

The example of a thrift file

- https://github.com/apache/thrift/blob/master/tutorial/tutorial.thrift
- **struct** – defines user type that contains several primitive types
- **exception-** is a structure that is returned in case of the error
- **service** – named definition of remote functions that are available for parts of conversation
- Other keywords are available as

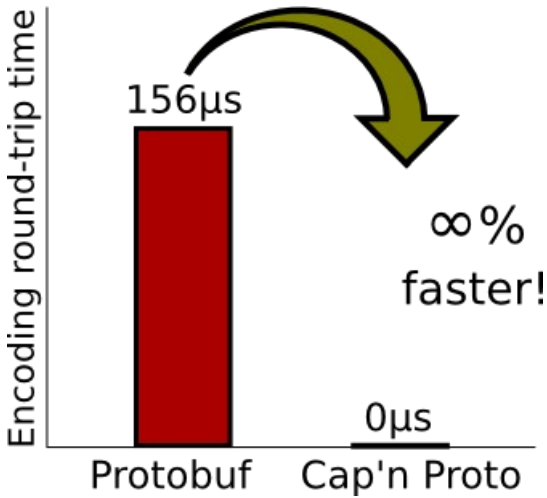
```
struct Work {
  1: i32 num1 = 0,
  2: i32 num2,
  3: Operation op,
  4: optional string comment,
}

/**
 * Structs can also be exceptions, if they are nasty.
 */
exception InvalidOperation {
  1: i32 whatOp,
  2: string why
}

/**
 * Ahh, now onto the cool part, defining a service. Services just need a name
 * and can optionally inherit from another service using the extends keyword.
 */
```

Apache Thrift alternative

- Capnproto – serialization and RPC library
- https://github.com/capnproto/capnproto
- Supports C++,C,python,Go,Rust



Sandesh technology

- Extends Apache Thrift
- Provides new tools for data collection and exchange (UVE, tracing, introspection, etc)
- New keywords are introduced: request, response, sandesh, traceobject, objectlog, uve

Sandesh types

- sandesh – indicates that type belongs to Sandesh system
- request – a request specification to the module (sub-system)
- response – a response specification for the module (sub-system)
- traceobject – trace record
- objectlog – log record

Collector / Generator roles

- Participants of conversations in Sandesh can have two main roles:
  - Collector (the one who collects messages/information)
  - Generator
- Each participant (application) of a conversation should initialize Sandesh context using InitCollector or InitGerator functions depending on its role
- ConnectToCollector – connects to a collector when it is created (actually, TCP/IP socket specification)

```
struct SandeshRole {
    enum type {
        Invalid,
        Generator,
        Collector,
        Test,
    };
};
```

Sandesh introspection ports

src/sandesh/common/vns.sandesh

"contrail-vrouter-nodemgr" : 8102

"contrail-vrouter-agent" : 8085

"contrail-control" : 8083

"contrail-collector" : 8089

"contrail-query-engine" : 8091

"contrail-analytics-api" : 8090

"contrail-dns" : 8092

"contrail-api" : 8084

"contrail-api:0" : 8084

"contrail-schema" : 8087

"contrail-svc-monitor" : 8088

"contrail-control-nodemgr" : 8101

"contrail-database-nodemgr" : 8103

"contrail-storage-stats" : 8105

"contrail-ipmi-stats" : 8106

"contrail-inventory-agent" : 8107

"contrail-alarm-gen" : 5995

"contrail-alarm-gen:0" : 5995

"contrail-snmp-collector" : 5920

"contrail-topology" : 5921

Sandesh, UVE, Logging

- Sandesh is an RPC and serialization tool for communicating between program components
- UVE is a tool to gather information about state of OpenSDN entities (virtual networks, VRF instances, etc)
- Logging is a tool to record some meaningful events of OpenSDN into a separate file
- Tracing is similar to Logging, but with storing messages in memory of modules

External libraries

- Intel TBB: management of parallel tasks inside one application
- Boost: an extended C++ containers library
- log4cplus: a logging library
- bind9: a DNS server
- scons: a python-based build management system
- Apache Thrift: an IDL and RPC library
- DPDK: a library for packets processing (a substitution for kernel network drivers)
- and others

Protocols and main formats

- REST – northbound interface
- IF-MAP – objects and data model
- BGP – RIB exchange w/ peers
- Netlink – southbound interface
- Shared memory – connection between vRouter modules
- JSON – conf storage
- XMPP – RIB exchange w/ VRs
- BGP – RIB exchange w/ peers
- Sandesh – IDL and analytics
- XML – data model storage

Unit tests frameworks

- TF uses next testing frameworks:
  - gtest for contrail-common, controller, vRouter Agent and other libs & apps written in C++;
  - pytest for vRouter Forwarder, uses:
    - DPDK to run vRouter Forwarder in user space
    - vtest program to send information into vRouter Forwarder mock
    - pylib to call vtest from python
    - sandesh to make communication between vtest and vRouter Forwarder

Only gtests are considered in next slides

Main repositories

- https://github.com/OpenSDN-io/community
- https://github.com/OpenSDN-io/docs
- https://github.com/OpenSDN-io/tf-vrouter
- https://github.com/OpenSDN-io/tf-controller

OpenSDN source code in numbers

- Not accounted parts of the project
  - Sandesh interface declarations
  - auto-generated code
  - Web UI
- Total count of lines of code for 4 languages: 1 497 376, amongth them:
  - C++: 786 799
  - Go: 350 164
  - Python: 286 965
  - C: 73 448
- With average efficiency of 70 lines per day we get approximately 92 worker-years

tf-dev-env

- A container for manual compilation and packaging OpenSDN components
- URL: https://github.com/OpenSDN-io/tf-dev-env
- Run on: Rocky 9.5 definetely, Ubuntu 22 (to be checked)

tf-dev-env first steps

- Use README.md provided by the repository
- Container requires root privileges or passwordless sudo
- Download the repository:
  git clone http://github.com/opensdn-io/tf-dev-env
- Initialize the repository:
  tf-dev-env/run.sh none

tf-dev-env available options

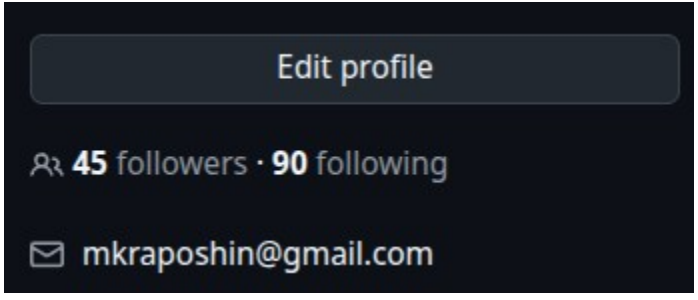

```
There are stages available to run ./run.sh <stage>:
  build     - perform sequence of stages: fetch, configure, compile, package
              (if stage was run previously it be skipped)
  fetch     - sync TF git repos
  configure - fetch third party packages and install dependencies
  compile   - build TF binaries
  package   - package TF into docker containers (you can specify target container to build like container-vrouter)
  test      - run unittests
  freeze    - prepare tf-dev-env for pushing to container registry for future reuse by compressing contrail directory
  upload    - push tf-dev-env to container registry
  none      - create the tf-dev-env container empty
  frozen    - fetch frozen tf-dev-env from Ci registry, you still have to use run.sh or fetch/configure to get sources
  doxygen   - builds doxygen documentation for the project
For advanced usage You can now connect to the sandbox container by using:
  sudo docker exec -it tf-dev-sandbox bash
```

Sending changes to the project

- All changes must be submitted via the Gerrit system: https://gerrit.opensdn.io/
- Contribution guide:
  - https://github.com/OpenSDN-io/docs/blob/master/contributing-to-opensdn/getting-started/getting-started-with-opensdn-development.rst
  - https://docs.opensdn.io/contributing-to-opensdn/getting-started/getting-started-with-opensdn-development.html

Main contribution steps

1) Prepare a GitHub account with publicly visible e-mail



2) Install Git and obtain an SSH key

3) Link your Gerrit account with the GitHub account

4) Install “git-review” on your local machine

Ask questions if they arise

Why OpenSDN

- Smaller community – easier to converse with members
- More chances for a commit to be accepted
- Less distances to core members
- It is mature, but young:
  - Great start for your career
  - Open to innovations and proposal