LF NETWORKING

LFN Developer & Testing Forum

LF NETWORKING

LFN Developer & Testing Forum

Bring Akraino/ICN-SDEWAN into EMCO

Yao, Le
Le, Huifeng

Anti-Trust Policy Notice

- Linux Foundation meetings involve participation by industry competitors, and it is the intention of the Linux Foundation to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

- Examples of types of actions that are prohibited at Linux Foundation meetings and in connection with Linux Foundation activities are described in the Linux Foundation Antitrust Policy available at http://www.linuxfoundation.org/antitrustpolicy. If you have questions about these matters, please contact your company counsel, or if you are a member of the Linux Foundation, feel free to contact Andrew Updegrove of the firm of Gesmer Updegrove LLP, which provides legal counsel to the Linux Foundation.

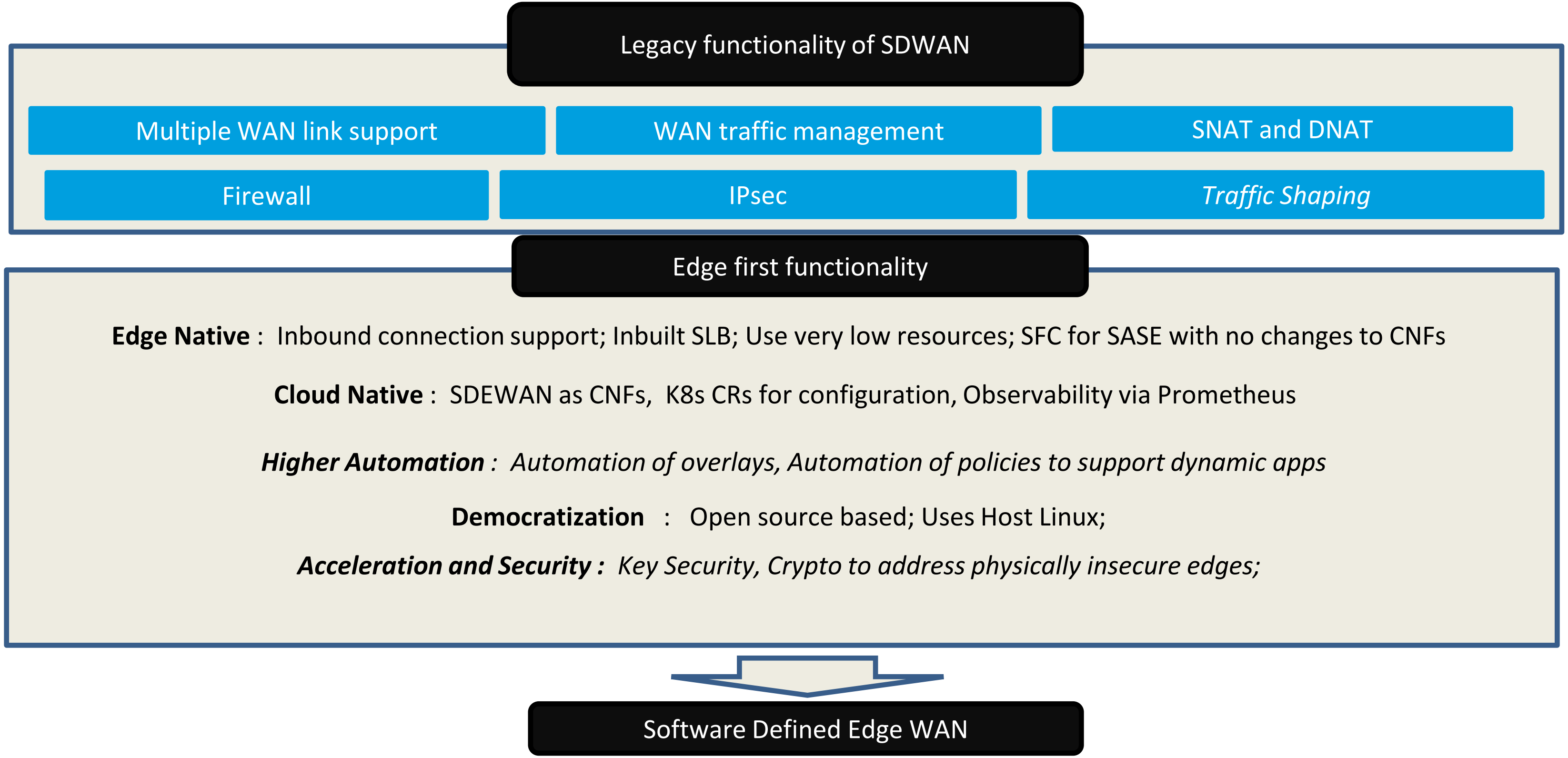
Agenda

- Motivation
- Introduce of SDEWAN
- Design and workflow
- Demo

Motivation

- EMCO is an orchestration of Application and Service, and DTC controller can only generate base traffic rules, we bring SDEWAN into EMCO to manage network communication between clusters
- SDEWAN can setup network topology between cluster for service and application, it also can deliver many other network functionalities to EMCO
- SDEWAN is a IPSEC based solution, it can enhance the network security, also, it can be accelerated by Intel QAT.

Introduce of SDEWAN



Legacy functionality of SDWAN

Multiple WAN link support

WAN traffic management

SNAT and DNAT

Firewall

IPsec

Traffic Shaping

Edge first functionality

Edge Native : Inbound connection support; Inbuilt SLB; Use very low resources; SFC for SASE with no changes to CNFs

Cloud Native : SDEWAN as CNFs, K8s CRs for configuration, Observability via Prometheus

Higher Automation *: Automation of overlays, Automation of policies to support dynamic apps*

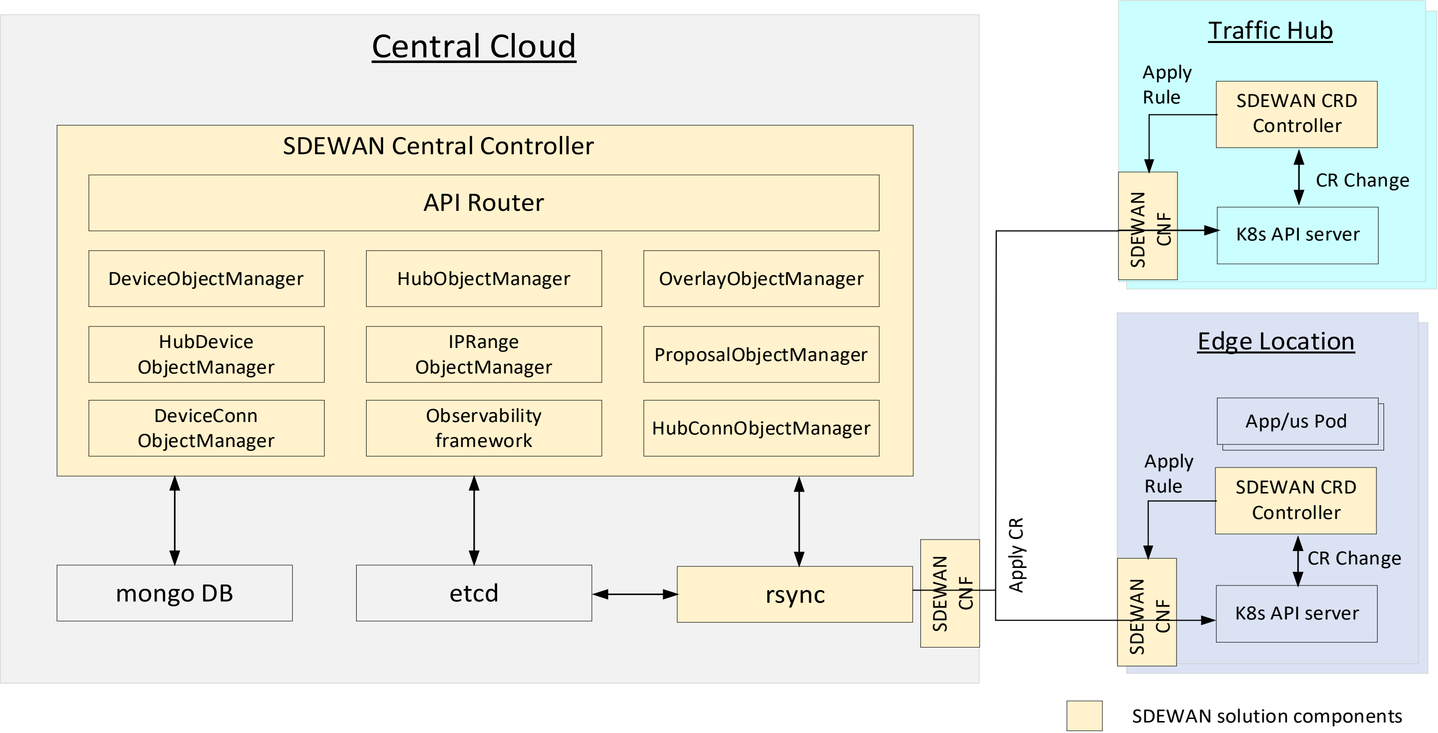
Democratization : Open source based; Uses Host Linux;

Acceleration and Security : *Key Security, Crypto to address physically insecure edges;*

Software Defined Edge WAN

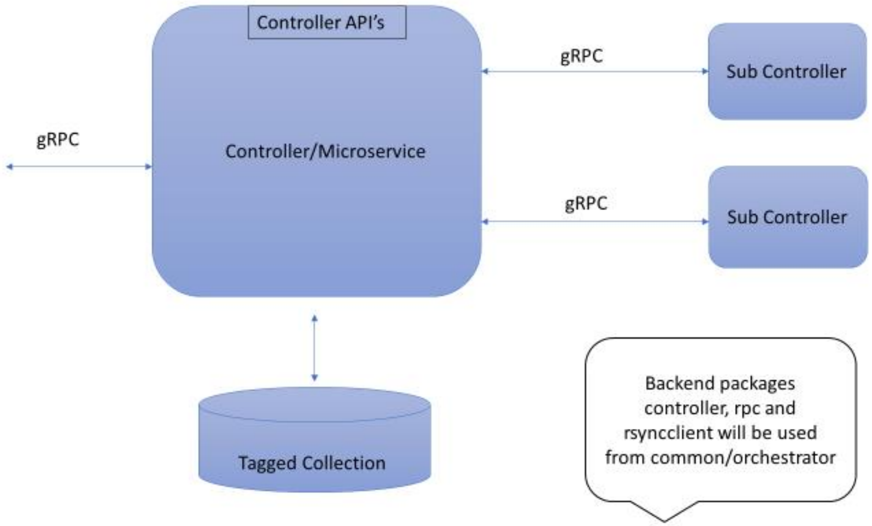
Introduce of SDEWAN

- CNF/CRD Controller: https://www.linkedin.com/pulse/software-defined-edge-wan-edges-srinivasa-addepalli/
- Central Controller: https://www.linkedin.com/pulse/software-defined-edge-wan-central-control-traffic-hub-addepalli/



EMCO DTC Overview

DTC is EMCO Distributed Traffic Controller used to generate and manage the traffic between EMCO Service and Application, users can add their own logic to DTC as a sub controller. Currently, DTC has Istio sub controller – its, service discovery – sds. The following is the architecture of the sub controller in EMCO.

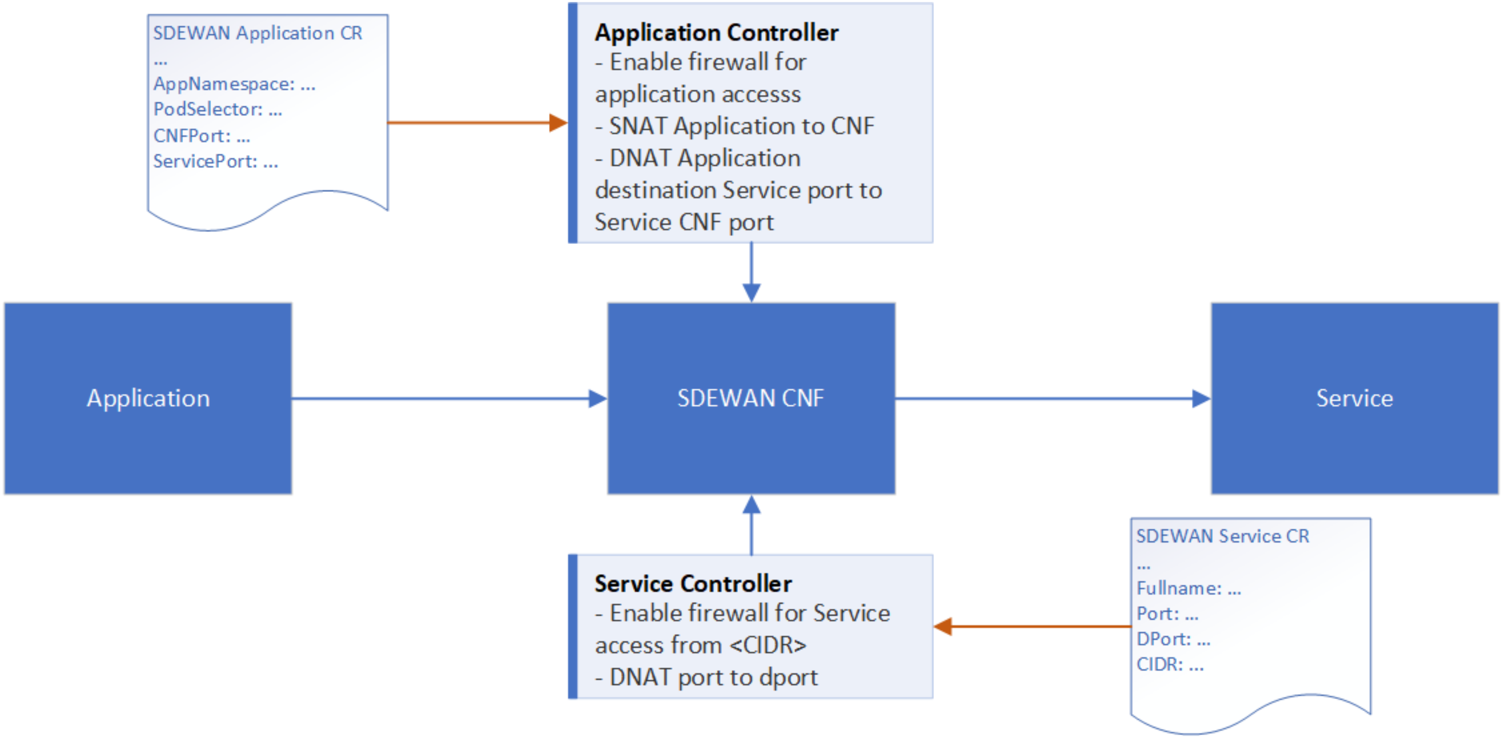


Backend packages controller, rpc and rsyncclient will be used from common/orchestrator

Design Target – SDEWAN Sub controller

- Get the cluster service and application information
- Check the SDEWAN controller status
- Generate Application CR and Service CR
  - Application CR
    - SNAT to CNF
    - Enable firewall traffic to Service
    - DNAT Service port to CNF port
  - Service CR
    - DNAT CNF port to Service port
    - Enable firewall traffic from subnet
- Store and apply the CR to EMCO application and Service

SDEWAN Service and Application



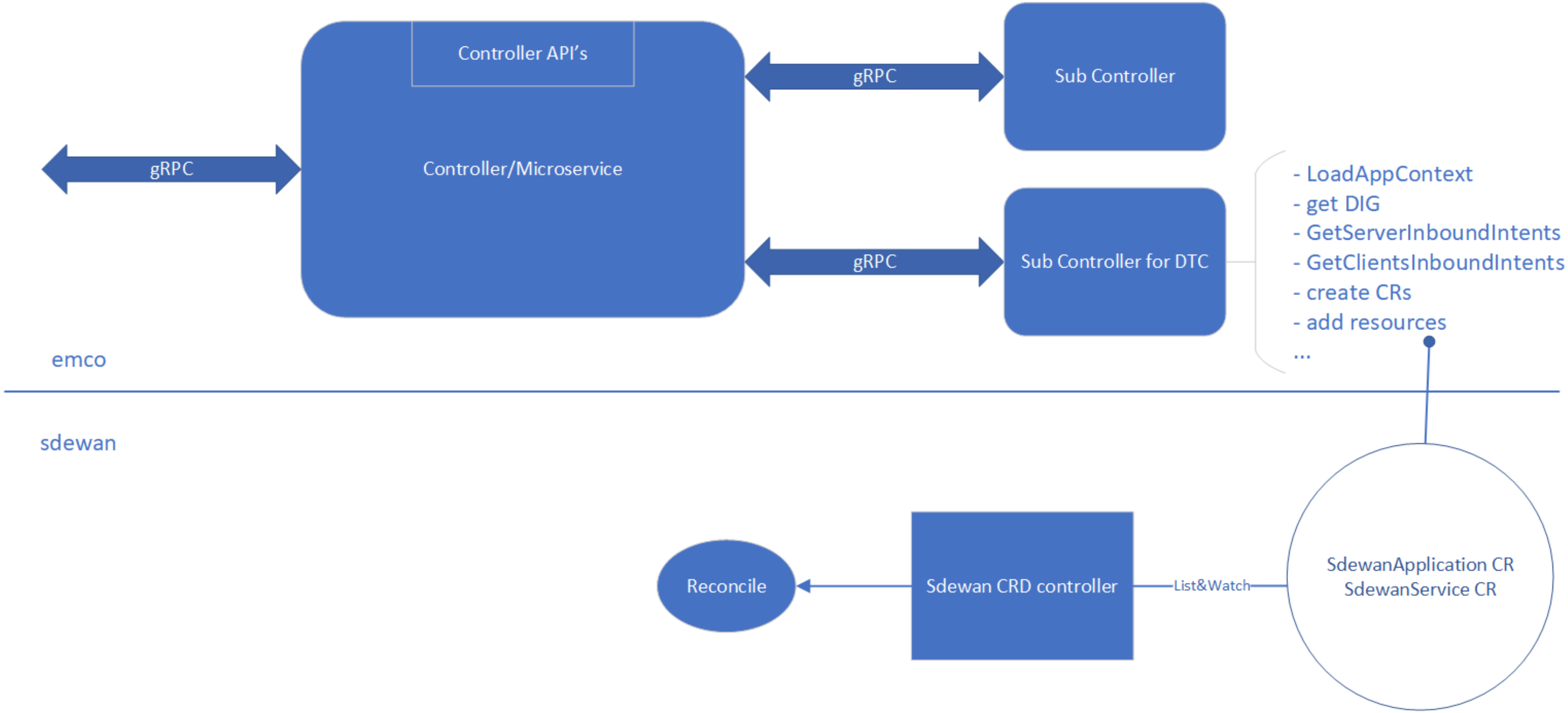
SDEWAN Application CR
...
AppNamespace: ...
PodSelector: ...
CNFPort: ...
ServicePort: ...

Application Controller
- Enable firewall for application accesss
- SNAT Application to CNF
- DNAT Application destination Service port to Service CNF port

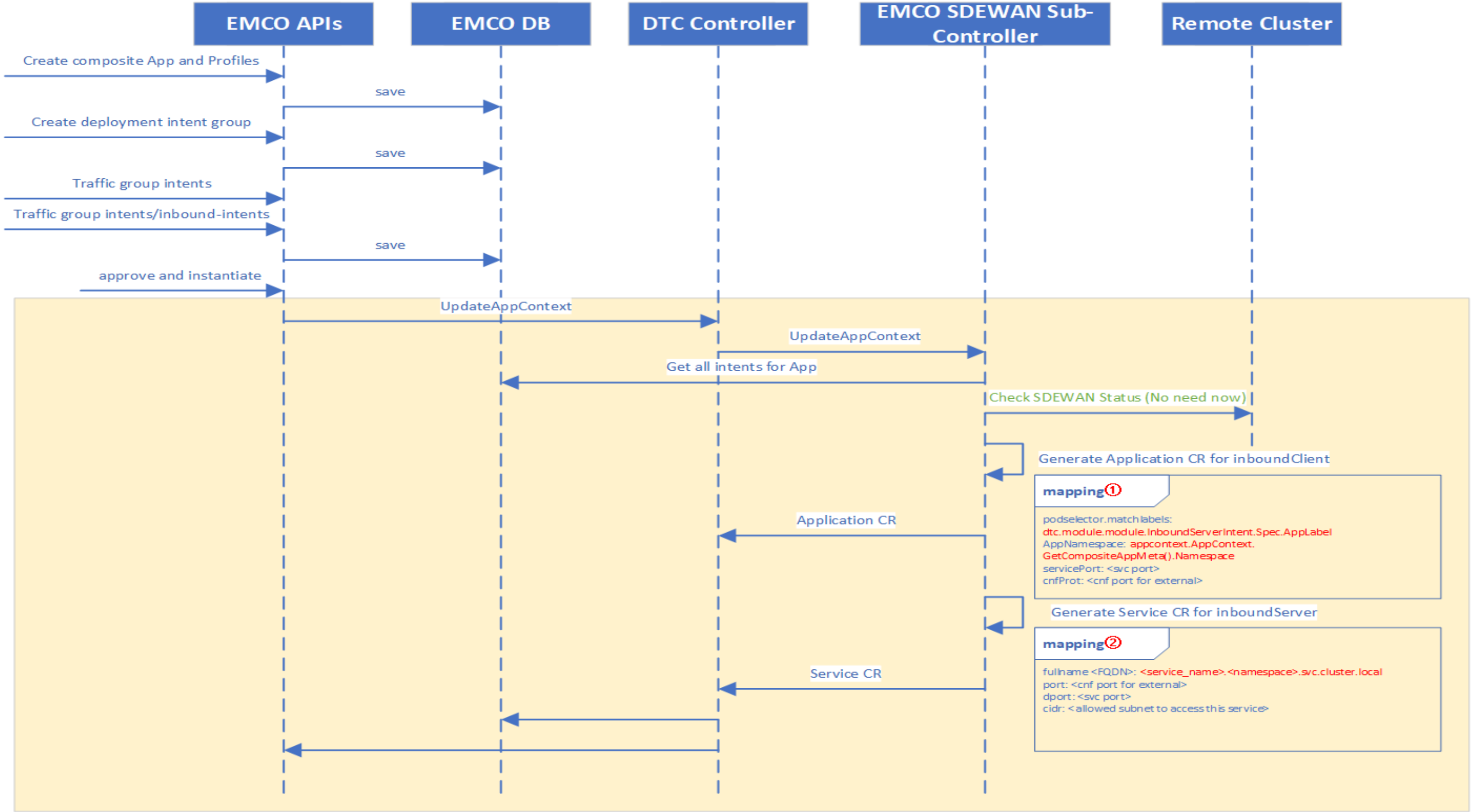
Service Controller
- Enable firewall for Service access from <CIDR>
- DNAT port to dport

SDEWAN Service CR
...
Fullname: ...
Port: ...
DPort: ...
CIDR: ...

Design diagram



Workflow



EMCO APIs

EMCO DB

DTC Controller

EMCO SDEWAN Sub-Controller

Remote Cluster

Create composite App and Profiles

save

Create deployment intent group

save

Traffic group intents

Traffic group intents/inbound-intents

save

approve and instantiate

UpdateAppContext

UpdateAppContext

Get all intents for App

Check SDEWAN Status (No need now)

Generate Application CR for inboundClient

mapping①

podselector.matchlabels:
dtc.module.module.InboundServerIntent.Spec.AppLabel
AppNamespace: appcontext.AppContext.
GetCompositeAppMeta().Namespace
servicePort: <svc port>
cnfProt: <cnf port for external>

Application CR

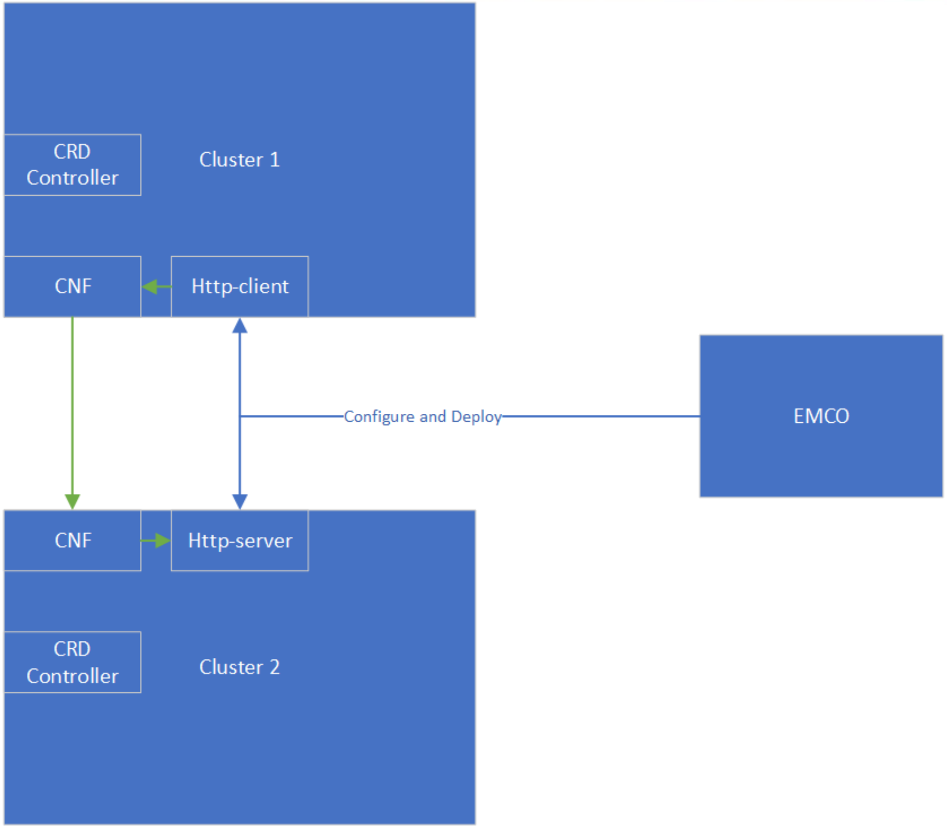
Generate Service CR for inboundServer

mapping②

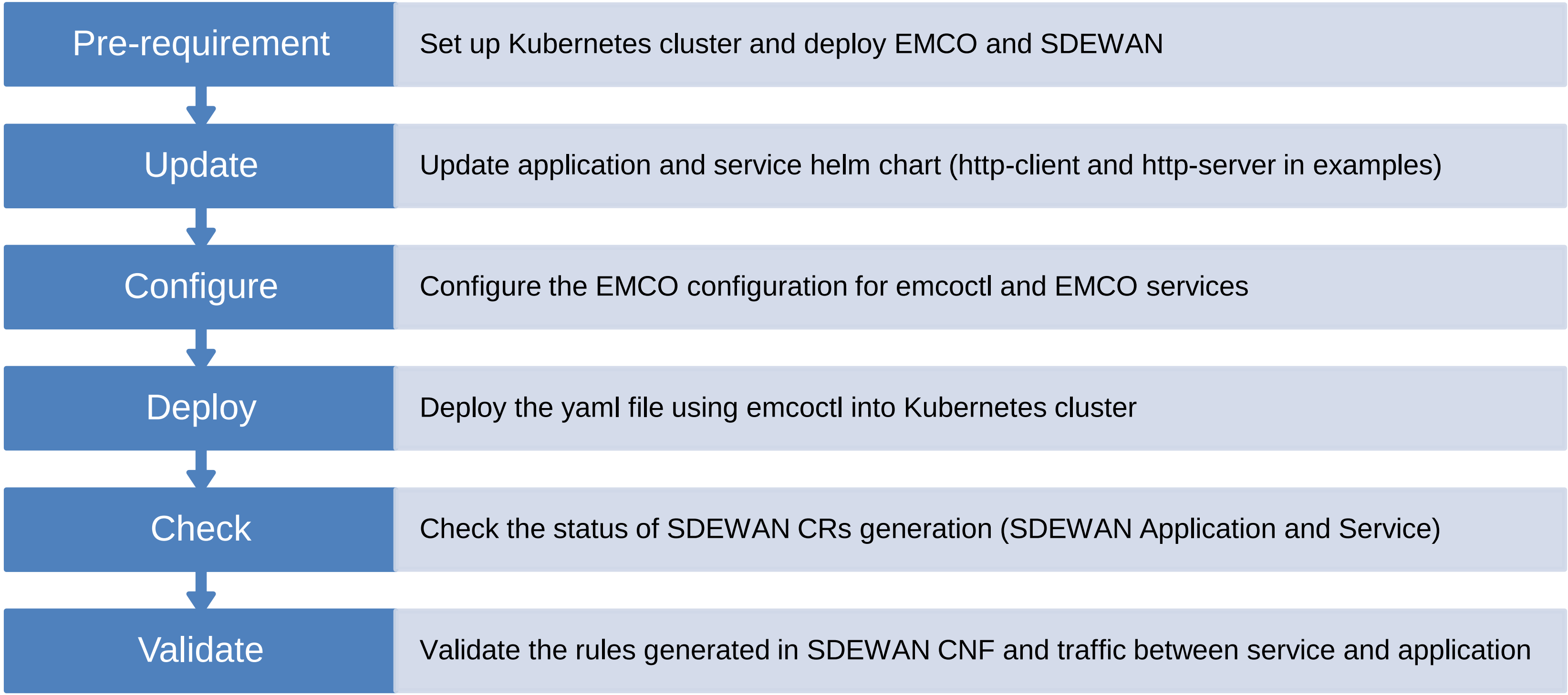
fullname <FQDN>: <service_name>.<namespace>.svc.cluster.local
port: <cnf port for external>
dport: <svc port>
cidr: <allowed subnet to access this service>

Service CR

Demo topo


CRD
Controller
Cluster 1
CNF
Http-client
Configure and Deploy
EMCO
CNF
Http-server
CRD
Controller
Cluster 2

Demo - setup



Pre-requirement: Set up Kubernetes cluster and deploy EMCO and SDEWAN

Update: Update application and service helm chart (http-client and http-server in examples)

Configure: Configure the EMCO configuration for emcoctl and EMCO services

Deploy: Deploy the yaml file using emcoctl into Kubernetes cluster

Check: Check the status of SDEWAN CRs generation (SDEWAN Application and Service)

Validate: Validate the rules generated in SDEWAN CNF and traffic between service and application

Configuration example

```
#creating controller entries
version: emco/v2
resourceContext:
  anchor: dtc-controllers
metadata :
  name: swc
spec:
  host: 192.168.121.205
  port: 30488
  type: "action"
  priority: 1
```

```
version: emco/v2
resourceContext:
  anchor: projects/proj1/composite-apps/collection-composite-app/v1/deployment-intent-groups/collection-deployment-intent-group/traffic-group-intents/testdtc/inbound-intents
metadata:
  name: serverin
  description: description of traffic intent
  userData1: user data 1
  userData2: user data 2
spec:
  app: http-server
  appLabel: app=http-server
  serviceName: http-service
  port: 3333
  dport: 4444
  cidr: 192.168.0.0/16
  protocol: TCP
  sdewanEnable: true
```

```
#add the client intent
version: emco/v2
resourceContext:
  anchor: projects/proj1/composite-apps/collection-composite-app/v1/deployment-intent-groups/collection-deployment-intent-group/traffic-group-intents/testdtc/inbound-intents/serverin/clients
metadata:
  name: client1
  description: description of traffic intent
  userData1: user data 1
  userData2: user data 2
spec:
  app: http-client
  appLabel: app=http-client
  serviceName: http-client
  namespaces: []
  cidrs: []
```

LF NETWORKING

LFN Developer & Testing Forum

Backup – Central Controller

SDEWAN Central Controller provides central control of SDEWAN overlay networks by automatically configuring the SDEWAN CNFs through SDEWAN CRD controller located in edge location clusters and hub clusters.

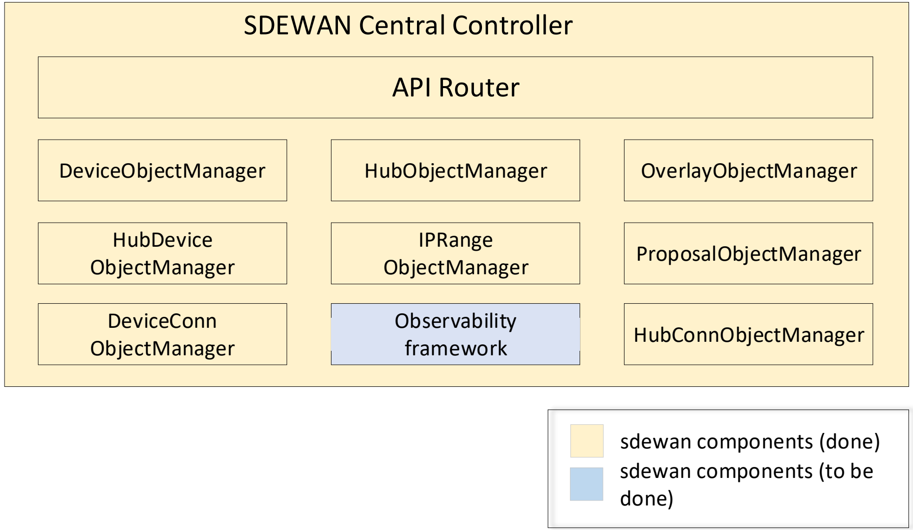
Detail Design : https://wiki.akraino.org/display/AK/SDEWAN+Central+Controller

API Definition: https://github.com/akraino-edge-stack/icn-sdwan/tree/master/central-controller/docs/scc_apis.yaml

Repo : https://github.com/akraino-edge-stack/icn-sdwan/tree/master/central-controller

Modules

- API Router: provides REST API router for SDEWAN Central Controller
- OverlayObjectManager: overlay registration, generate overlay root cert
- HubObjectManager: hub registration and setup hub connection mesh
- DeviceObjectManager: device/cluster registration and setup device connection mesh (if device has public IP)
- HubDeviceObjectManager: setup connection between hub and device
- IPRangeObjectManager: ip range registration and allocate/free overlay ip for device
- ProposalObjectManager: proposal registration
- DeviceConnManager: only support GET, query connection information of device
- HubConnObjectManager: only support GET, query connection information of hub
- Observability framework: system status monitoring, including connection status, CNF status etc.



Backup – CRD Controller

SDEWAN CRD Controller is implemented as k8s CRD Controller, it manages CRDs (e.g. Firewall related CRDs, Mwan3 related CRDs and IpSec related CRDs etc.) and internally calls SDEWAN Restful API to do CNF configuration. And a remote client (e.g. SDEWAN Central Controller) can manage SDEWAN CNF configuration through creating/updating/deleting SDEWAN CRs.

Detail Design : https://wiki.akraino.org/display/AK/Sdewan+CRD+Controller

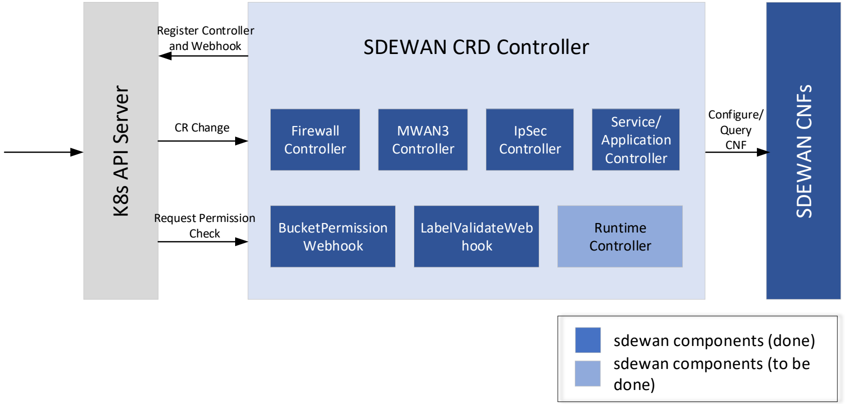
Controller and CRD definition: https://github.com/akraino-edge-stack/icn-sdwan/blob/master/platform/crd-ctrlr/examples/sdewan-controller.yaml

CR Samples: https://github.com/akraino-edge-stack/icn-sdwan/tree/master/platform/crd-ctrlr/src/config/samples

Repo : https://github.com/akraino-edge-stack/icn-sdwan/tree/master/platform/crd-ctrlr

Modules

- MWAN3 Controller: monitor mwan3 related CR change then do mwan3 configuration in SDEWAN CNF
- Firewall Controller: monitor firewall related CR change then do firewall configuration in SDEWAN CNF
- IpSec Controller: monitor ipsec related CR change then do ipsec configuration in SDEWAN CNF
- Service/Application Controller: configure firewall/NAT rule for in-cluster service and application
- Runtime controller: collect runtime information of CNF include IPSec, IKE, firewall/NAT connections, DHCP leases, DNS entries, ARP entries etc..
- BucketPerssion/LabelValidateWebhook: do sdewan CR request permission check based on CR label and user



Backup – CNF

SDEWAN CNF is implemented based on OpenWRT, it enhances OpenWRT Luci web interface with SDEWAN controllers to provide Restful API for network functions' configuration and control.

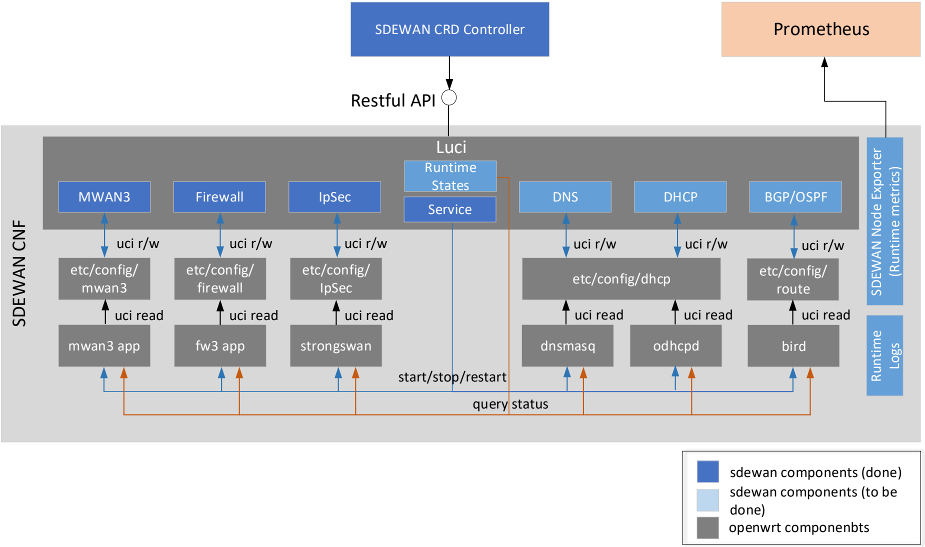
Detail Design / Rest API : https://wiki.akraino.org/display/AK/SDEWAN+CNF

API Samples: https://github.com/akraino-edge-stack/icn-sdwan/blob/master/platform/test/e2e-test/edge-scripts/sdwan_verifier.sh

Repo : https://github.com/akraino-edge-stack/icn-sdwan/tree/master/platform/cnf

Modules

- MWAN3: mwan3 configuration for multiple WAN links’ management
- Firewall: fw3 configuration for firewall rule, NAT rule.
- IpSec: strongswan configuration to setup security tunnel between CNFs
- DNS/DHCP: dnsmasq configuration for DNS and DHCP (ip4) or odhcpd configuration for DHCP (ip6)
- BGP/OSPF: bird configuration for BGP/OSPF auto routing
- Runtime States: query network function applications to collect runtime information such as IPSec, IKE, firewall/NAT connections, DHCP leases, DNS entries, ARP entries etc.
- Service: manage (e.g. start, stop, restart etc.) lifecycle of network function applications (e.g. mwan3, fw3, strongswan etc.)
- Node exporter: Prometheus exporter to export CNF runtime metrics to Prometheus
- Runtime Logs: exports system log for debugging


SDEWAN CRD Controller
Prometheus
Restful API
SDEWAN CNF
Luci
Runtime States
Service
MWAN3
Firewall
IpSec
DNS
DHCP
BGP/OSPF
uci r/w
uci r/w
uci r/w
uci r/w
uci r/w
uci r/w
etc/config/ mwan3
etc/config/ firewall
etc/config/ IpSec
etc/config/dhcp
etc/config/ route
uci read
uci read
uci read
uci read
uci read
uci read
mwan3 app
fw3 app
strongswan
dnsmasq
odhcpd
bird
start/stop/restart
query status
SDEWAN Node Exporter (Runtime metrics)
Runtime Logs
sdewan components (done)
sdewan components (to be done)
openwrt componenbts