

SDEWAN Central Controller

- [System Architecture](#)
- [System Design](#)
 - [Assumption](#)
 - [Environment Setup \(Pre-condition\)](#)
 - [Restful API definition and Back-End flow](#)
 - [Error handling](#)
- [DB Schema](#)
- [Module Design](#)
- [Task Breakdowns](#)

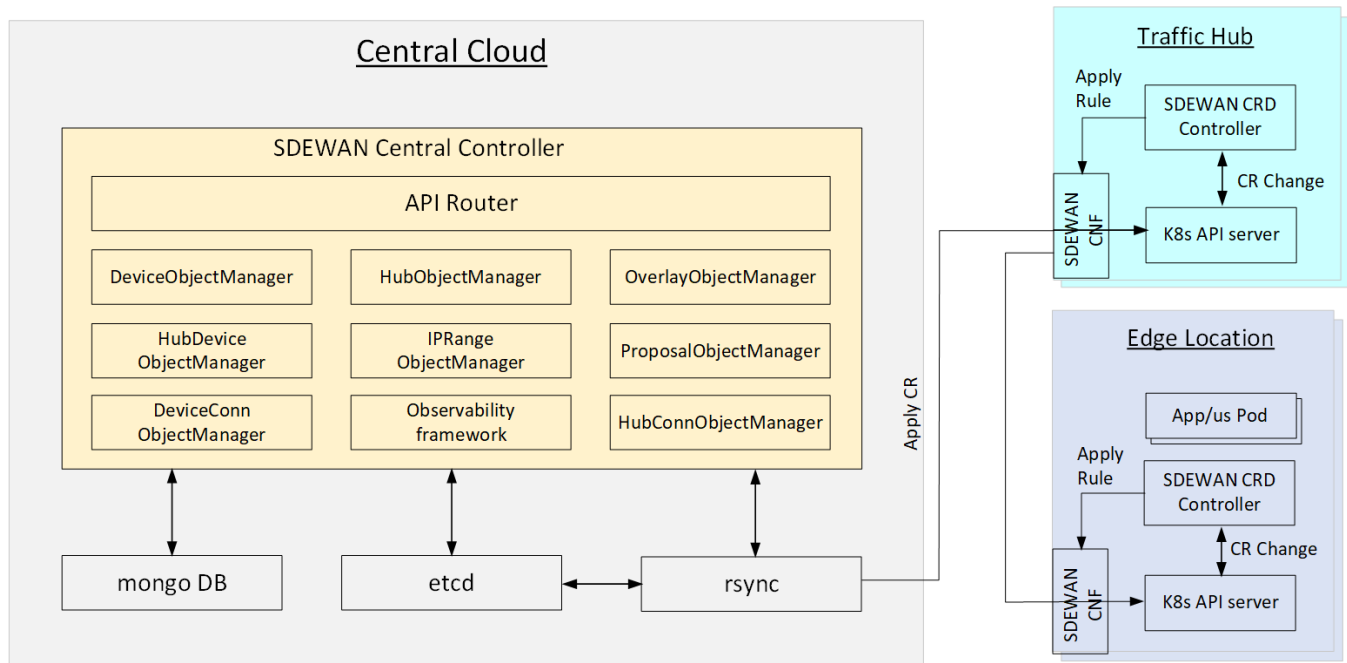
SDEWAN central controller provides central control of SDEWAN overlay networks by automatically configuring the SDEWAN CNFs located in edge location clusters and hub clusters:

- To create secure overlays where each overlay connects application and hub clusters together.
- To allow application connectivity with external entities and entities of other clusters.

System Architecture

The system includes the following micro-services as showed in below diagram:

- SDEWAN Central Controller:
 - 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 status of device
 - HubConnObjectManager: only support GET, query connection status of hub
 - Observability framework: system status monitoring, including connection status, CNF status etc.
- Rsync: a daemon service which accepts request from SDEWAN Central Controller (through RPC) then deploy relevant K8s CRs of SD-EWAN CNFs of various hubs and edges to establish the tunnels.
- Mongo DB: a database to store information such as edge clusters, hubs, overlays, ip addresses, application/services etc.
- Etc: a metadata database to exchange configuration information between SDEWAN Central Controller and Rsync



System Design

Assumption

IP

- Central Cloud has public IP as CIP
- Traffic Hub has public IP as HIP1 HIP2, ...
- Edge Location (Device) may have public IP in one edge node as EIP1, ... or don't have public IP (behind a gateway as EGIP1, ...)

Connection for control plane (e.g. central cloud to k8s API server):

- Central Cloud to Traffic Hub: Direct connection through Hub's public IP
- Central Cloud to Edge Location:
 - Edge location has public IP: Direct connection through Edge Location's public IP
 - Edge location does not have public IP: Using Edge Location owned hub's SDEWAN CNF as proxy

IPSec Tunnel mode for data plane (for data traffic)

- Edge to Edge: Host to host
- Edge to Hub: Host (edge) to Site (Hub, using edge's subnet as rightsubnet)
- Hub to Hub: Host to Host

Environment Setup (Pre-condition)

Central Cloud:

- K8s cluster is setup (by Kud)
- Web UI (Optional), API Server, Rsync backend, DB service are deployed (manually or through EMCO)

Traffic Hub:

- K8s cluster is setup (by Kud)
- Hub SDEWAN CRD Controller and CNF are deployed (through EMCO) with initial configuration (e.g. NAT: enable DNAT for k8s API service and Istio Ingress service).

Edge Location (With Public IP):

- K8s cluster is setup (by Kud)
- Edge SDEWAN CRD Controller and CNF are deployed (through EMCO) with initial configuration (e.g. NAT: enable DNAT for k8s API service and Istio Ingress service).

Edge Location (With Private IP):

- K8s cluster is setup (by Kud)
- Edge SDEWAN CRD Controller and CNF are deployed (through EMCO) with initial configuration (e.g. NAT: enable DNAT for k8s API service and Istio Ingress service; IPSec: as Initiator for Control plane - left: %any, leftsourceip:%config, right: Owned Hub's HIP, rightsubnet:0.0.0.0/0).

Restful API definition and Back-End flow

Resource	Description	URL	Fields	Back-End flow
Overlay	Define a group of edge location clusters (devices) and hubs, a overlay is usually owned by one customer and full mesh connections are setup automacally between hub-hub and device-device (with public IPs)	/scc/v1/overlays	<ul style="list-style-type: none">• name• description• cid	Registration: <ul style="list-style-type: none">• SCC requests a CA from cert-manager, the CA is used as root CA for this overlay• SCC save the caid in DB
Proposal	Define proposals which can be used for IPSec tunnel in this overlay	/scc/v1/overlays/{overlay-name}/proposals	<ul style="list-style-type: none">• name• description• encryption• hash• dhgroup	Registration: <ul style="list-style-type: none">• SCC saves the proposals information in DB
Hub	Define a traffic Hub in an overlay	/scc/v1/overlays/{overlay-name}/hubs	<ul style="list-style-type: none">• name• description• publicips• certificate• kubeconfig	Registration: <ul style="list-style-type: none">• SCC checks hub's k8s API server access with kubeConfig for each ip in publicips• For each registered hub in this overlay<ul style="list-style-type: none">◦ SCC requests cert-manager to generate a public/private key pair based on overlay CA◦ SCC generates the IPSec CR for new hub and registered hub then call rsync to deploy CR to setup route based host-IPsec tunnel (with BGP/OSPF enabled):<ul style="list-style-type: none">▪ All proposals in this Overlay will be used as candidate proposals for IPSec configuration▪ Use the public/private key pair generated in previous step as IPSec cert• SCC saves hub information in DB

IPRange	Define the overlay IPRange which will be used as OIP of devices	/scc/v1/overlays/{overlay-name}/ipranges	<ul style="list-style-type: none"> name description subnet minip maxip 	Registration: <ul style="list-style-type: none"> SCC save ip range information in DB
Device	Define a edge location device information which may be a CNF, VNF or PNF	/scc/v1/overlays/{overlay-name}/devices	<ul style="list-style-type: none"> name description publicips forceHubConnection proxyHub proxyHubPort useHub4Internet dedicatedSFC certificateid kubeConfig 	Registration: <ul style="list-style-type: none"> If has publicips and forceHubConnection==false: <ul style="list-style-type: none"> SCC checks device's k8s API server access with kubeConfig for each ip in publicips For each registered device of this overlay: <ul style="list-style-type: none"> SCC requests cert-manager to generate a public/private key pair based on overlay CA SCC generates the IPsec CR for new device and registered device then call rsync to deploy CR to setup host-host IPsec tunnel: <ul style="list-style-type: none"> All proposals in this Overlay will be used as candidate proposals for IPsec configuration Use the public/private key pair generated in previous step as IPsec cert else <ul style="list-style-type: none"> (Assumption) Kud configures device as Initiator to proxyHub SCC find 1 available OIP from overlay's IPRange, configure then deploy (through rsync) IPsec CR for proxyHub as Responder with OIP as the only 1 candidate ip for Initiator <ul style="list-style-type: none"> Expectation: the IPsec tunnel between proxyHub and device should setup up and the device will get the assigned OIP SCC creates DNAT CR (dst: HIP, dst_port: proxyHubPort change to dst: OIP, dst_port: 6443) then deploy to proxyHub (SCC will auto generate a proxyHubPort if not provided) SCC checks device's k8s API server access with kubeConfig for proxyHub:proxyHubPort For each registered device with public ip and forceHubConnection==false: <ul style="list-style-type: none"> SCC requests cert-manager to generate a public/private key pair based on overlay CA SCC generates the IPsec CR for new device and registered device (with public IP) then call rsync to deploy CR to setup host-host IPsec tunnel: <ul style="list-style-type: none"> All proposals in this Overlay will be used as candidate proposals for IPsec configuration Use the public/private key pair generated in previous step as IPsec cert SCC saves device information in DB
Hub-device connection	Define a connection between hub and device	/scc/v1/overlays/{overlay-name}/hubs/{hub-name}/devices/{device-name}	<ul style="list-style-type: none"> N/A 	Create: <ul style="list-style-type: none"> SCC find 1 available OIP from overlay's IPRange, configure then deploy (through rsync) IPsec CR for hub as Responder with OIP as the only 1 candidate ip for Initiator SCC configure the deploy IPsec CR as Initiator to Hub for device <ul style="list-style-type: none"> Expectation: the IPsec tunnel between hub and device should setup up and the device will get the assigned OIP <p>Todo: Confirm "ip route" rule for OIP in this hub and all other hub are setup automatically or need new CR to execute linux shell in host</p>

Error handling

DB Schema

Module Design

Task Breakdowns

Tasks	Due	Owner	Status	Description
Scheduler Manager				
-- Overlay: Setup tunnels for hubs and edges				Generates relevant K8s CRs of SD-EWAN CNFs of various hubs and edges to establish the tunnels
-- IP Address manager				Assigns/frees IP addresses from "overlay IP ranges" and dedicates them to that cluster
-- Application connectivity scheduler				Creates K8s resources required to be pushed into the edges and corresponding traffic hubs to facilitate the connectivity
-- Resource Synchronizer				
-- CNF				
API Server				
-- Rest API Backend				Rest API server framework

-- DB Backend				Proxy to DB
-- Application Cluster management				
-- Hub management				
-- Overlay management				
-- Status monitoring management				
-- logging				
Web UI				
-- Web UI framework				
-- Application Cluster Registration				
-- Hub Registration				
-- Overlay				
-- Application/Service Registration				
-- Status tracking				
EMCO plugin for SDEWAN				
E2E Integration				Integration test of overall system