Router Architects

Ucentral-Client and VYOS Setup as Gateway for APNOS

B17O739-79

Router Architects

TABLE OF CONTENTS

TABLE OF CONTENTS 2
POC OverView 4
Objective 4
Prerequisites 4
Demo Network Topology 5
Components Overview 5
Network Structure 6
OLG Architecture 6
Communication Workflow Between uCentral and VyOS 6
Step-by-Step Process 7
POC Setup 9
Step 1: Setup Host, VyOS and Ucentral-Client Containers 9
Step 2: Wait for the VyOS Container to become Functional 12
Step 3: Run the Ucentral Container 12
Container Image Generation for VYOS 13
Steps to Build VyOS Docker image 13
Container Image Generation for Ucentral Client 13
Build Image RootFs and Prepare paths and a clean build context 13
Copy the Files Required to prepare image for Ucentral Client for OLG 14
Add BusyBox and applet symlinks (so sh, ps, ls, … work) 14
Copy Ucentral's ucode scripts 15
Copy All Libraries 15
Make Executables Runnable 15
Copy necessary files required to make ucentral functional with cloud controller and VyOS 15
Build Container Image for Ucentral 15

Date	Revision	Description	Modified by	Reviewed by
28/10/2025	.9	OLG POC POC Setup OLG Architecture Ucentral Image	Navneet Barwal	
13/11/2025	0.9.1	Updates for Topology	Navneet Barwal	

POC OverView

It is a PoC to show the interoperability of OpenWiFi AP with OLG (Ucentral Client + VyOS) which should be configured by the Cloud Controller.

VyOS and Ucentral Client run in separate Docker Containers, Ucentral container handles configure commands from the cloud controller and convert these to VyOS style configuration and calls VyOS HTTP Server API's to push those configurations to VyOS container . The VyOS Container then provides connectivity to APNOS devices by configuring a DHCP server on its LAN. The host machine on which this is tested runs UBUNTU 24.04.1. The image generation for VyOS and Ucentral Client will be covered at last. The setup comes with a prerequisite to have both container images. The OpenWifi Version of Ucentral Client for OLG, and Cloud Controller is 3.1.

Objective

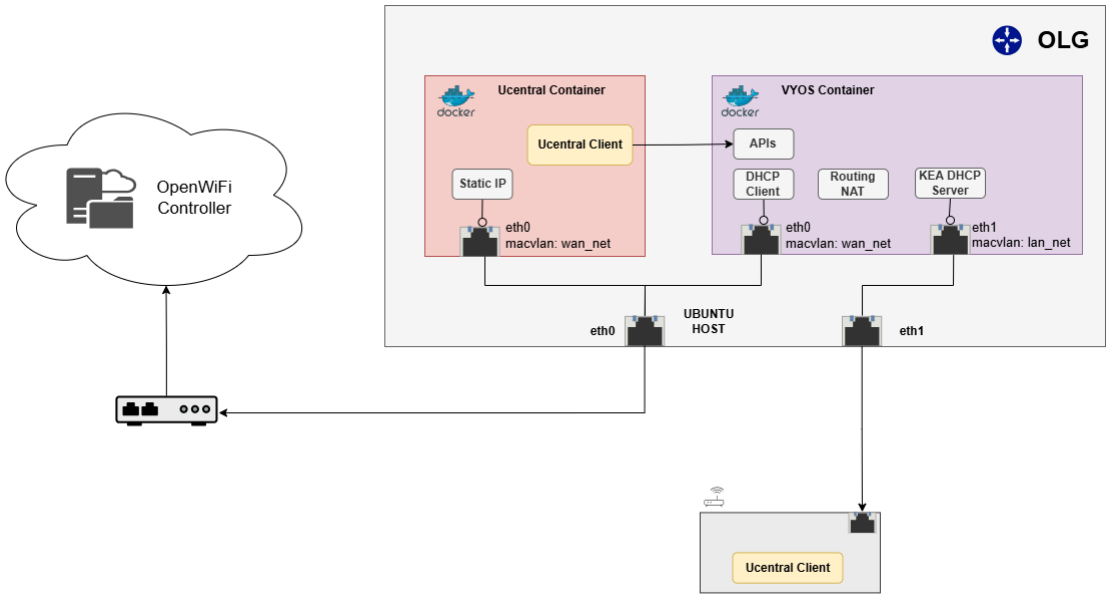
- Demonstrate how the OpenWiFi AP-NOS (via uCentral Client) can be managed from the cloud (uCentral Gateway) with a gateway device (VyOS) acting as the "edge" router.
- Validate that the gateway (VyOS) is properly configured (WAN/LAN separation) and that the AP-NOS devices can connect through to the cloud.
- Demonstrate zero-touch or near zero-touch provisioning of the AP via uCentral Client when pointed to the gateway and cloud.
- Confirm traffic flows: from AP → LAN net → VyOS LAN interface → VyOS WAN interface → Internet .

Prerequisites

- A Linux/Ubuntu host with **2 physical ethernet interfaces**:
    - `eth0` to WAN, configured via DHCP from upstream or manually.
    - `eth1` to LAN side, manual static IP (e.g. `192.168.50.2`).
- Docker installed; ability to use macvlan networks (requires appropriate kernel & privileges).
- VyOS container (image: `vyos-2025.09.10-0018-rolling-generic:olgV1`) — acts as the gateway/router.
- uCentral container (image: `ucentral-client:olgV1`) — acts as the configuration endpoint for the VyOS.
- The contents for host vyos_config volume which contains a postconfig script and config.boot to load default VyOS configurations.

Demo Network Topology

The diagram below illustrates the **OpenWiFi Local Gateway (OLG) Proof-of-Concept topology** for integrating **uCentral-Client** with **VyOS** as a gateway. The setup demonstrates how an APNOS connects through a local VyOS router toward the uCentral cloud, using Docker-based virtual networks on a single host system.



Components Overview

- **Host Machine :** The physical or virtual Linux system running Docker. It has two physical interfaces — `eth0` for the WAN side and `eth1` for the LAN side. These interfaces act as parents for the macvlan networks used by the containers.
- **uCentral Container :** It runs the uCentral Client process responsible for establishing a WebSocket connection to the uCentral cloud server. It uses a static IP in the `wan_net` subnet and connects to the cloud directly. It also translates json type uCentral configurations into VyOS text style configurations and also calls VyOS HTTPS Server API's to configure these configurations received from the cloud.
- **VyOS Container** : Acts as the **gateway router**. It provides Layer-3 routing and NAT between wan_net and lan_net. It has:
    - **eth0** in wan_net (DHCP client)
    - **eth1** in lan_net A **KEA DHCP Server** that assigns IP addresses to clients on the LAN subnet.
- **uCentral Client Device :** Represents the access point or OpenWiFi device running the uCentral client. It connects to the VyOS LAN (lan_net) and obtains an IP address

via DHCP from VyOS. From there, it reaches the cloud through NAT on the VyOS container.

- **Cloud Controller :** The remote endpoint (e.g., openwifi1.routerarchitects.com:15002) that manages device configuration, telemetry, and provisioning.

Network Structure

The system uses two isolated Docker macvlan networks to emulate real physical subnets:

- **wan_net :** Connects VyOS and uCentral containers to the WAN side (Internet/cloud).
- **lan_net :** Connects VyOS (LAN side) and APNOS.

Each container has its own macvlan child interface on the host's physical NIC, effectively isolating their traffic like separate physical devices.

OLG Architecture

The **Open Local Gateway (OLG)** functions as an **Edge Gateway** bridging the **uCentral Client** (northbound interface) with the **VyOS Router** (southbound interface).
It acts as an intermediary layer between the **Cloud Controller** (OpenWiFi backend) and **APNOS/OLS devices** deployed at the edge.

The OLG runs on a host system (for example, Ubuntu) and consists of two key containers:

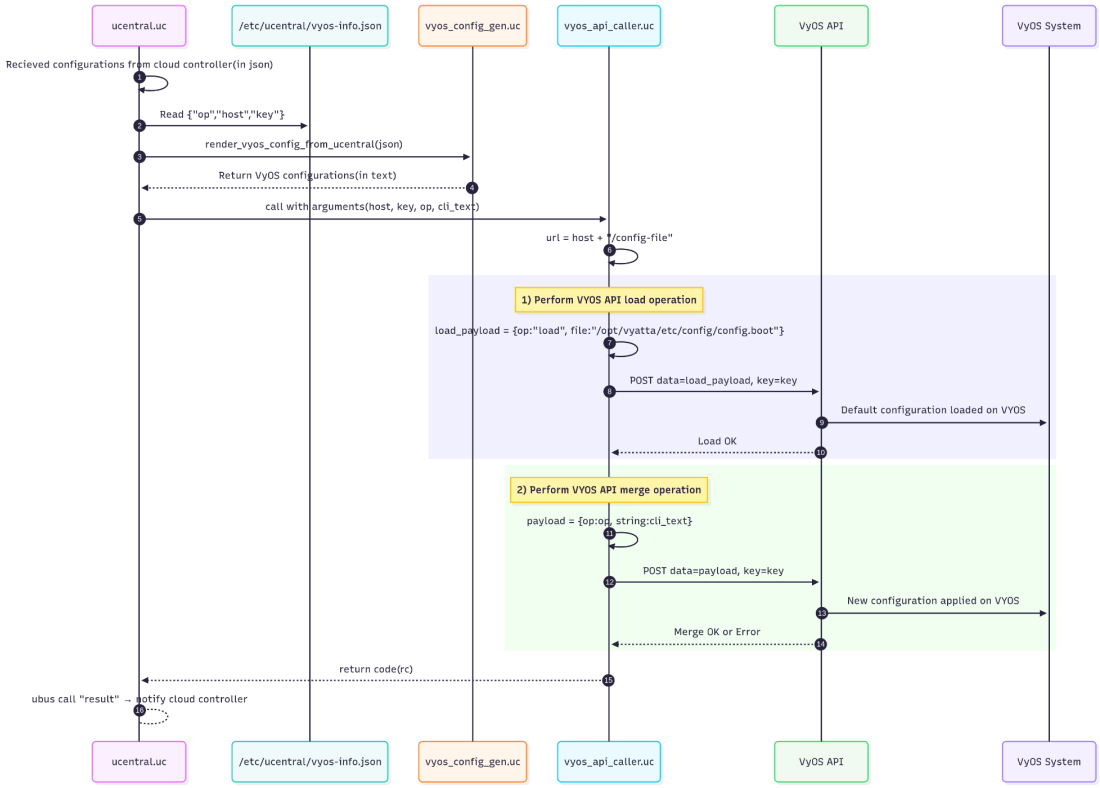
- **uCentral Client Container** – Handles communication with the OpenWiFi Cloud Controller, receives configuration updates, and reports status through the uCentral protocol.

- **VyOS Container** – Acts as a virtual gateway device responsible for routing, NAT, and DHCP functionality. It applies configurations dynamically as received from the cloud via uCentral.

This section focuses on the **interaction between uCentral Client and VyOS** containers within the OLG environment.

Communication Workflow Between uCentral and VyOS

The diagram below illustrates the step-by-step interaction flow that occurs when the **uCentral Client** receives configuration data from the **Cloud Controller**, processes it, and applies it to **VyOS** through its REST API.



Step-by-Step Process

1. **Configuration Reception (uCentral.uc)**

    - The uCentral Client receives configuration data (in JSON format) according to ucentral-schema from the Cloud Controller.

2. **Reading Local VyOS Configuration Info**

    - The uCentral client reads `/etc/ucentral/vyos-info.json` to retrieve connection details such as the VyOS host address, operation type and authentication parameters.

3. **Generate VyOS Configuration (vyos_config_gen.uc)**

    - The uCentral Client invokes the `render_vyos_config_from_ucentral()` function.
      This function translates uCentral configuration JSON into a VyOS CLI configuration format (`cli_text`), which is compatible with VyOS APIs.
    - The generated VyOS configuration (in text form) is returned to the uCentral logic layer, ready to be pushed to the VyOS system.

4. **Call VyOS API Caller (vyos_api_caller.uc)**

    - The uCentral client prepares a request and calls `vyos_api_caller.uc` with arguments:

        - `host` – VyOS API endpoint
        - `key` – VyOS HTTP server authentication key
        - `op` – operation type (e.g., "load" or "merge")
        - `cli_text` – generated configuration text

5. **Perform VyOS API "Load" Operation**

    - The VyOS API Caller constructs a payload:
      `{ "op": "load", "file": "/opt/vyatta/etc/config/config.boot" }`
    - It sends a POST request to the VyOS API endpoint `/config-file` with this payload.
    - This instructs VyOS to load its default configuration from the specified file path.

6. **Perform VyOS API "Merge" Operation**

    - After the base configuration is loaded, the client constructs a second payload:
      `{ "op": "merge", "string": "<cli_text>" }`
    - The configuration text generated earlier is merged into the active VyOS configuration.

    - The API returns "Merge OK" if successful, or an error message if the merge fails.

    - The result is a dynamically updated VyOS configuration reflecting the settings from the cloud.

7. **Result Handling and Notification**

    - The uCentral Client collects the operation's return code and response status.
    - It then makes a **ubus call** named `"result"` to notify the Cloud Controller of success or failure.
    - This closes the feedback loop between the cloud, OLG, and VyOS gateway.

POC Setup

Step 1: Setup Host, VyOS and Ucentral-Client Containers

This is a script to setup Host and Containers for OLG (olg_setup.sh)

```
#!/usr/bin/env bash


# === CONFIG VARIABLES ===

# Network & IP settings

WAN_NET_SUBNET="192.168.76.0/24"

WAN_NET_GATEWAY="192.168.76.1"

LAN_NET_SUBNET="192.168.50.0/24"

LAN_NET_GATEWAY="192.168.50.254"


UCENTRAL_IP="192.168.76.31"


# Volume / paths

VYOS_CONFIG_VOLUME="./vyos/vyos_config/"

MODULES_VOLUME="/lib/modules"


# Image names

VYOS_IMAGE="routerarchitect123/vyos-2025.09.10-0018-rolling-generic:olgV1"

UCENTRAL_IMAGE="routerarchitect123/ucentral-client:olgV1"


# === END CONFIG VARIABLES ===
```

```
# --- Clean up previous run ---
docker stop vyos-olg      2>/dev/null || true
docker rm   vyos-olg      2>/dev/null || true
docker stop ucentral-olg 2>/dev/null || true
docker rm   ucentral-olg 2>/dev/null
docker network rm wan_net 2>/dev/null || true
docker network rm lan_net 2>/dev/null || true

# --- Create networks ---
docker network create -d macvlan \
 --subnet=${WAN_NET_SUBNET} \
 --gateway=${WAN_NET_GATEWAY} \
 --subnet=fd00:aaaa:bbbb::/64 \
 --ipv6 \
 --gateway=fd00:aaaa:bbbb::1 \
 -o parent=eth0 \
 wan_net

docker network create -d macvlan \
 --subnet=${LAN_NET_SUBNET} \
 --gateway=${LAN_NET_GATEWAY} \
 --subnet=fd00:cccc:dddd::/64 \
 --gateway=fd00:cccc:dddd::1 \
 --ipv6 \
```

```
  -o parent=eth1 \
  --aux-address="vyos=192.168.50.1" \
  lan_net

# --- Run VyOS container ---
docker run -d --name vyos-olg --privileged \
  --network wan_net \
  -v ${MODULES_VOLUME}:/lib/modules \
  -v ${VYOS_CONFIG_VOLUME}:/opt/vyatta/etc/config \
  ${VYOS_IMAGE} /sbin/init

docker network connect lan_net vyos-olg

# --- Run UCentral container ---
docker run -dit --name ucentral-olg --privileged \
  --network wan_net --ip ${UCENTRAL_IP} \
  ${UCENTRAL_IMAGE}

echo "Setup completed successfully."
```

What this script does

- Removes any previous containers/networks to start clean.
- Creates two macvlan docker networks: `wan_net` (for WAN side) and `lan_net` (for LAN side).
- Runs a VyOS container attached first to `wan_net`, then connects to `lan_net` (so the container has two interfaces: one in WAN, one in LAN).
- Runs a uCentral container on the WAN network with a static IP.

- Leaves you with a VyOS container having both WAN & LAN connectivity; uCentral container on WAN side.
- Note: The variables are provided to be configured as per your requirement.

Step 2: Wait for the VyOS Container to become Functional

- Get Access to the VyOS container , it may take over a minute.

  docker exec -it vyos-olg su - vyos

- Inside the VyOS container: wait until eth0 (WAN) gets an IP from DHCP Server and have the default route set on eth0 . Note the IP received by eth0 , we need to provide this IP to Ucentral Configuration for VyOS.

Step 3: Run the Ucentral Container

- Get Access to the Ucentral Container.

  docker exec -it ucentral-olg /bin/ash

- Modify /etc/ucentral/vyos-info.json file with IP Address of the VyOS container noted earlier.
- Start Ubus , execute the following commands

  /sbin/ubusd &

- This is a preconfigured image with certificates and a serial number for demo. Hence the serial number and cloud controller URL remains static . Now start the Ucentral process in the foreground.

  /usr/sbin/ucentral -S 74d4ddb965dc -s openwifi1.routerarchitects.com -P 15002 -d

Step 4: Verify Default Configurations

Inside the VyOS Container check eth1 interface is configured with IP provided for LAN from Ucentral and DHCP Server and this DHCP server provides connectivity to the APNOS Client connected to the eth1 interface.

VyOS Command to check DHCP lease

show dhcp server leases state all

You can ssh onto this device and also check its internet connectivity and can see in the list of added devices in the cloud.

Step 5: Modify Configurations from Cloud

Now the cloud controller modifies the configuration of Gateway Ucentral(74d4ddb965dc) for changing the network of LAN and it should reflect successfully on the device.

You can also monitor logs on VyOS through command

monitor log

Container Image Generation for VYOS

- Download VYOS Rolling Image from https://vyos.net/get/nightly-builds/

Steps to Build VyOS Docker image

```
mkdir vyos && cd vyos

mkdir rootfs

sudo mount -o loop
vyos-2025.09.10-0018-rolling-generic-amd64.iso rootfs

sudo apt-get install -y squashfs-tools

mkdir unsquashfs

sudo unsquashfs -f -d unsquashfs/
rootfs/live/filesystem.squashfs

sudo tar -C unsquashfs -c . | docker import -
vyos-2025.09.10-0018-rolling-generic

sudo umount rootfs

cd ..

sudo rm -rf vyos
```

Container Image Generation for Ucentral Client

Build Image RootFs and Prepare paths and a clean build context

mkdir WORKSPACE

cd WORKSPACE

mkdir OPENWIFI_WLANAP

cd OPENWIFI_WLANAP

```
git clone https://github.com/routerarchitects/ra-openwifi-wlan-ap.git
git checkout  release/v3.1.0
./build.sh x64_vm
```

After complete build check

```
ls openwrt/build_dir/target-x86_64_musl/root-x86/
```

Then Set Paths to copy contents for ucentral image

```
ROOT=~/WORKSPACE/OPENWIFI_WLANAP/ra-openwifi-wlan-ap/openwrt/build_
dir/target-x86_64_musl/root-x86
DEST=~/ucentral-init
rm -rf "$DEST"
mkdir -p
"$DEST"/{bin,sbin,usr/bin,usr/sbin,usr/share,lib,usr/lib,etc/ucentral,lib/config,lib/
functions}
```

Copy the Files Required to prepare image for Ucentral Client for OLG

```
cp -a "$ROOT"/usr/sbin/ucentral     "$DEST"/usr/sbin/
cp -a "$ROOT"/etc/group    "$DEST"/etc/
cp -a "$ROOT"/etc/passwd  "$DEST"/etc/
cp -a "$ROOT"/sbin/ubusd   "$DEST"/sbin/
cp -a "$ROOT"/bin/ubus   "$DEST"/bin/
cp -a "$ROOT"/usr/bin/curl  "$DEST"/usr/bin/
```

Add BusyBox and applet symlinks (so sh, ps, ls, … work)

```
cp -a "$ROOT"/bin/busybox "$DEST"/bin/
( cd "$DEST/bin" && ln -sf busybox sh && ln -sf busybox ash )
for a in ls ps ifconfig ping ip cat grep cut mkdir rm cp mv ln touch \
      mount umount basename readlink vi date uname echo sleep dmesg logger
flock; do
 ln -sf busybox "$DEST/bin/$a"
done
```

```
# if BusyBox lacked an applet, copy real ones if present:
[ -f "$ROOT/usr/bin/logger" ] && cp -a "$ROOT/usr/bin/logger" "$DEST/usr/bin/"
[ -f "$ROOT/usr/bin/flock"  ] && cp -a "$ROOT/usr/bin/flock"  "$DEST/usr/bin/"
```

Copy Ucentral's ucode scripts

```
cp -a "$ROOT"/usr/share/ucentral   "$DEST"/usr/share/       2>/dev/null || true
cp -a "$ROOT"/usr/bin/ucode        "$DEST"/usr/bin/         2>/dev/null || true
cp -a "$ROOT"/usr/lib/ucode        "$DEST"/usr/lib/         2>/dev/null || true
```

Copy All Libraries

```
cp -a "$ROOT"/lib/*   "$DEST"/lib/
cp -a "$ROOT"/usr/lib "$DEST"/usr/
```

Make Executables Runnable

```
chmod +x "$DEST"/bin/busybox \
      "$DEST"/usr/sbin/ucentral \
      "$DEST"/usr/bin/ucode 2>/dev/null || true
```

Copy necessary files required to make ucentral functional with cloud controller and VyOS

```
cp cas.pem "$DEST"/etc/ucentral/
cp cert.pem "$DEST"/etc/ucentral/
cp key.pem "$DEST"/etc/ucentral/
cp capabilities.json "$DEST"/etc/ucentral/
cp vyos-info.json "$DEST"/etc/ucentral/
cp ucentral.cfg.0000000001 "$DEST"/etc/ucentral/
cp vyos_config_gen.uc "$DEST"/usr/share/ucentral/
cp  vyos_api_caller.uc "$DEST"/usr/share/ucentral/
cp  ucentral.uc "$DEST"/usr/share/ucentral/
```

Build Container Image for Ucentral

```
cat > "$DEST/Dockerfile" <<'EOF'
FROM scratch
COPY bin/  /bin/
COPY sbin/ /sbin/
COPY usr/  /usr/
COPY lib/  /lib/
```

```
COPY etc/  /etc/
RUN mkdir -p /var/run/
RUN mkdir -p /var/lock/
RUN mkdir -p /tmp
ENV PATH=/bin:/sbin:/usr/bin:/usr/sbin
ENV LD_LIBRARY_PATH=/lib:/usr/lib
WORKDIR /
CMD ["/bin/busybox","sleep","infinity"]
EOF

cd "$DEST"
docker build -t ucentral:latest .
```