ICN - SDEWAN

- Goals
  - Basic Technology
    - OpenWRT
    - ovn4nfv-k8s-plugin
  - Design Proposals
    - Implementation
    - Deployment
    - Runtime Configuration
  - SDWAN CNF Controller Implementations
    - Timeline
    - SDEWAN CNF
    - Sdewan config Agent
- Presentation

Goals

In ICN’s SDWAN usages, SFC (Service Function Chain) is designed to support Corp networks to connect to external internet with security connection. The SFC includes Security VNF (e.g. firewall etc.), WAN Opt CNF and SDWAN VNF/CNF, and SDWAN module is worked as software defined router which can be used to defined the rules when connect to external internet. Below diagram shows where SDWAN module located in the whole system.

Basic Technology

OpenWRT

The OpenWRT Project ([https://openwrt.org/](https://openwrt.org/)) is an open source project based on Linux, and it is primarily used on embedded devices to route network traffic. There are more than 3500 software packages which can be installed on OpenWRT via opkg package management system. OpenWRT provides both docker image and VM image to support virtualization solution ([https://openwrt.org/docs/guide-user/virtualization/start](https://openwrt.org/docs/guide-user/virtualization/start)). In ICN, we run OpenWRT in container.

OpenWRT Mwan3 package (a replacement for multiwan package) provides the capabilities for multiple WAN management: WAN interfaces management, outbound traffic rules, traffic load balancing etc.

ovn4nfv-k8s-plugin

ovn4nfv-k8s-plugin is a CNI plugin based on ovn. It can work together with Multus CNI to add multiple interfaces for the pod. One of the interfaces is the Multus default interface, it could be flannel, calico, etc. The other interfaces are added by ovn4nfv-k8s-plugin according the the pod annotation. With ovn4nfv-k8s-plugin, we can create virtual network in run-time. Also we can connect the pod with the provider network, this is important for CNF.

Design Proposals

Implementation
SDWAN module is implemented as CNF instead of VNF for better performance and proving of CNF can also be effect way to deploy SDWAN

SDWAN will leverage OpenWRT for its functionality:

- Base Image: x86-generic rootfs image (http://downloads.openwrt.org/releases/18.06.4/targets/x86/generic/)
- Packages:
  - mwan3: multiple wans configuration and management
  - luci-app-mwan3: web configuration interface for mwan3
  - uhttp: web server for luci services

Deployment

Helm will be used to deploy SDWAN CNF as pod on nodes with external network connection with below parameters:

- Network interfaces: multiple interfaces from different CNI plugins, e.g. one is Multus default CNI (e.g. flannel or Calico), one is OVN CNI to connect to virtual network1 then connect to corp network, and the last one is OVN CNI which connect to external router
- WAN configuration: an initial script running in the container will use UCI command to do initial WAN configuration based on the input

```
config interface 'wan'
  option enabled '1'
  list track_ip
    '8.8.4.4'
    list track_ip
    '8.8.8.8'
    list track_ip
    '208.67.222.222'
    list track_ip
    '208.67.220.220'
  option reliability
    '2'
  option count '1'
  option timeout '2'
  option interval '5'
  option down '3'
  option up '8'

config member 'wan_mi_w3'
  option interface 'wan'
  option metric '1'
  option weight '3'
```

- Initial Traffic rules (e.g. policy, rule etc.)

```
config policy 'wan_only'
  list use_member
    'wan_mi_w3'

config rule 'sticky_even'
  option src_ip
    '0.0.0.0/0.0.0.1'
  option dest_port
    '443'
  option proto 'tcp'
  option use_policy 'wan_only'
```

Runtime Configuration

SDWAN traffic rules and WAN interfaces are required to be configured at runtime through Restful API interface.

OpenWRT luci invoking mechanism:

1) logon: POST /cgi-bin/luci with luci_username and luci_password to get sysauth Cookie

```
e.g. wget --post-data "luci_username=root&luci_password=" http://192.168.56.2/cgi-bin/luci/
```
response header will include sysauth cookie, like: "Set-Cookie: sysauth=e5b2e5c2ae0099c078bb3cb72052ed95;"

(2) Call luci http service with sysauth Cookie
e.g. `wget --header="Cookie:sysauth=e5b2e5c2ae0099c078bb3cb72052ed95" http://192.168.56.2/cgi-bin/luci`

Mwan3 supported services:

* `interface_status`: GET /cgi-bin/luci/admin/status/mwan/interface_status

<table>
<thead>
<tr>
<th>Sample response:</th>
</tr>
</thead>
</table>
| "interfaces": {"wan": {"running": true, "score": 0, "track_ip": [{"status": "down", "latency": 0, "packetloss": 0, "ip": "208.67.220.220"}, {"status": "down", "latency": 0, "packetloss": 0, "ip": "208.67.222.222"}, {"status": "down", "latency": 0, "packetloss": 0, "ip": "8.8.8.8"}], "lost": 536, "status": "offline", "age": 5, "turn": 134}, "wanb6": {"running": false, "score": 0, "track_ip": [], "lost": 0, "status": "up", "age": 0, "turn": 0}, "wanb": {"running": false, "score": 0, "track_ip": [], "lost": 0, "status": "up", "age": 0, "turn": 0}, "wan6": {"running": false, "score": 0, "track_ip": [], "lost": 0, "status": "up", "age": 0, "turn": 0}}]}

* `detailed_status`: GET /cgi-bin/luci/admin/status/mwan/detailed_status
* `diagnostics_display`: GET /cgi-bin/luci/admin/status/mwan/diagnostics_display
* `troubleshooting_display`: GET /cgi-bin/luci/admin/status/mwan/troubleshooting_display

Note: configuration for WAN's interface, member, policy and rule are supported in the web UI, but with no direct restful API available. It needs implement luci cgi plugin to provide restful API for WAN configuration. In ICN, one interface is implemented as POC to support running commands in CNF through Rest API call.

* `command`:

  POST /cgi-bin/luci/admin/config/command

  Execute commands in OpenWRT CNF

  Normal Response Code: 200
  Error Response Code: 400

  Request:

<table>
<thead>
<tr>
<th>Name</th>
<th>In</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>Body</td>
<td>String</td>
<td>commands to be executed in CNF, each command is separated by semicolon</td>
</tr>
</tbody>
</table>

**SDWAN CNF Controller Implementations**

The Akraino-SDEWAN-OutOfBoxNetworking.pptx describes the whole SDEWAN solution design.

ICN R3 release will focus on implementing the SDEWAN CNF and EWAN config Agent then designing the End-to-End demo with other components described in the SDEWAN solution.

Below diagrams describe CRD definition and the interaction the calling sequence of SDEWAN Conf Mgr, K8s, Sdewan CNF and Sdewan Conf Agent.

Sdewan CRD definition:
Calling Sequence:

```
apiVersion: batch.sdwron
kind: Sdewon
metadata:
  name: sdewon1
spec:
  node: node1
  networks:
    - name: ovn-net1
      isProvider: true
    - name: ovn-net2
      isProvider: false
  mwan3Conf: mwan3conf1
  firewallConf:
    status:
      msg: triggered by...
      mwan3Status:
        appliedTime: "...
        isApplied: true
        name: mwan3conf1
```

Apply conf via calling api on port 80

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: sdewon1
    name: sdewon1
  namespace: default
  ownerReferences:
    - apiVersion: batch.sdwron
      blockOwnerDeletion: true
      controller: true
      kind: Sdewon
      name: sdewon1
  spec:
    containers:
      - command:
          - /bin/sh
          - /tmp/sdewonentrypoint.sh
        image: integratedcloudnative/openwrt:dev
        name: sdewon
        readinessProbe:
          httpGet:
            path: /
            port: 80
            scheme: HTTP
    restartPolicy: OnFailure
```

```
apiVersion: batch.sdwron
kind: Mwan3Conf
metadata:
  name: mwan3conf1
spec:
  policy:
    balance1:
      members:
        - network: ovn-net1
          weight: 2
          metric: 2
        - network: ovn-net2
          weight: 3
          metric: 3
      rule:
        http:
          policy: balance1
          dest_ip: 0.0.0.8/0
          dest_port: 80
```

```
apiVersion: batch.sdwron
kind: FirewallConf
metadata:
  name: firewallconf1
spec:
  policy:
    balance1:
      members:
        - network: ovn-net1
          weight: 2
          metric: 2
        - network: ovn-net2
          weight: 3
          metric: 3
      rule:
        http:
          policy: balance1
          dest_ip: 0.0.0.8/0
          dest_port: 80
```

SDEWON Controller

Sdewon Reconcile Func

- On Sdewon CR creating/updating:
  1. Create openwrt pod if not exist
  2. Wait for pod ready
  3. Apply mwan3 configuration if specified (via openwrt API)
  4. Apply firewall configuration if specified (via openwrt API)

- On Sdewon CR deleting:
  1. Delete openwrt pod

Mwan3Conf Reconcile Func

- On Mwan3Conf CR creating/updating:
  1. List the Sdewon CRs which uses this Mwan3Conf CR
  2. Update the listed Sdewon CRs status (Status.Mwan3Status.IsApplied=false) to trigger Sdewon Reconcile which applies new configurations.

- On Mwan3Conf CR deleting:
  1. List the Sdewon CRs which uses this Mwan3Conf CR
  2. Update the listed Sdewon CRs status (Status.Mwan3Status.IsApplied=false) to trigger Sdewon Reconcile which applies new configurations.
1. System Deployment process

- ICN deploys K8s cluster and installs kud addon: (1) Multus CNI plugin as default CNI plugin (2) ovn4k8s CNI plugin (3) Sdewan Conf Agent (sdewan-operator) as deployment
  - Note: Sdewan-operator includes (1) Sdewan Controller (monitor Sdewan CR) (2) Mwan3conf Controller (monitor Mwan3Conf CR) (3) FirewallConf Controller (monitor FirewallConf CR) (4) IpSec Controller (Monitor IpSec CR)
- Admin (or SDEWAN Config manager?) creates (1) Network CR (to setup OVN virtual network) (2) Provider Network CR (to setup provider network by configuration network interface on each node)

2. Create SDEWAN CNF (Pod) process

- SDEWAN conf mgr creates Mwan3Conf CR (or FirewallConf CR, IpSecConf CR), the CRs (for Mwan3Conf CR, it defined the mwan3 policy/rule) are saved in k8s etcd as K8s resources
- SDEWAN conf mgr creates Sdewan CR with below information:
  - Node: the CNF pod should be created on which node
  - Interfaces: include (1) internal network interface which connect to OVN virtual network (2) provider network interface which connect to provider network
  - Configuration: the name of pre-defined Mwan3Conf/FirewallConf/IpSecConf CR
- Sdewan Controller (running inside Sdewan Conf Agent) gets the notification of new-created Sdewan CR, call K8s API to (1) create Sdewan CNF (Pod and Service) on required Node (through NodeSelector) (2) Create ConfigMap which stores logical network interface information(used to generate /etc/config/network file in Sdewan CNF container).
  - Note: OpenWRT applications (such as mwan3, firewall, ipsec etc.) do not use system network interfaces (e.g. "eth0", "net1" which can be listed by "ip a") directly, instead, it uses the logical interfaces (such as "lan", "wan1" etc.), and the logical interfaces are map to real network interfaces in file /etc/config/network
- K8s creates the Sdewan CNF pod and call ovn4k8s CNI plugin to attach required network interfaces (defined in Sdewan CR) with the Pod
- When the Pod is ready (through k8s readiness check), Sdewan Controller (running inside Sdewan Conf Agent) calls the rest API (through Node’s FQDN) to (1) login (2) Set configuration (defined in Mwan3Conf, FirewallConf or IpSecConf) to setup initial rule inside the CNF (3) restart Mwan3 (or Firewall, IpSec) service to apply the rules in the CNF
  - Note: the configuration rules can be updated/added/deleted at runtime in Update/Delete Rule process

3. Update/Delete Rule process (use Mwan3conf as example)

- SDEWAN conf mgr updates Mwan3Conf CR (or FirewallConf CR, IpSecConf CR), the CR is saved inside k8s etcd as resource
- K8s notifies Mwan3Conf controller (run inside Sdewan Conf Agent) the CR update/delete event
- Mwan3Conf controller (run inside Sdewan Conf Agent) finds all Sdewan CRs which uses this Mwan3Conf (through Sdewan CR’s Mwan3Conf property), then update the status of the found Sdewan CRs to trigger Sdewan controller's Reconcile which updates/deletes new Mwan3Conf and restarts Mwan3 service to apply the change.

4. Delete SDEWAN CNF (Pod) process
- **SDEWAN conf mgr deletes Sdewan CR**
- **K8s notifies Sdewan controller (run inside Sdewan Conf Agent) the CR delete event**
- **Sdewan controller (run inside Sdewan Conf Agent) finds the Sdewan CNF owned by this deleted CR, call k8s API to delete the CNF**

**Timeline**

<table>
<thead>
<tr>
<th>Module</th>
<th>Tasks</th>
<th>Owner</th>
<th>Due</th>
<th>Current Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDEWAN CNF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service API</td>
<td></td>
<td>Huifeng</td>
<td></td>
<td>Done</td>
<td>Start/stop/restart/reload SDWAN service, includes: mwan3, firewall/NAT, IPsec. Referenc: SDEWAN CNF#SDEWANService</td>
</tr>
<tr>
<td>MWAN3 API</td>
<td></td>
<td>Huifeng</td>
<td></td>
<td>Done</td>
<td>Support MWAN3 rule/policy configuration. Referenc: SDEWAN CNF#MWAN3</td>
</tr>
<tr>
<td>Firewall API</td>
<td></td>
<td>Huifeng</td>
<td>Design</td>
<td>WW08: Initial design Done</td>
<td>Support firewall configuration for zone (general rule for a group of interfaces), forwarding (iptables forward), rule, redirect (DNAT/SNAT). Referenc: SDEWAN CNF#Firewall</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WW09: Implementation - 50%</td>
<td>WW10: 80%</td>
<td>WW11: done</td>
</tr>
<tr>
<td>IPSec API</td>
<td></td>
<td>Ruoyu</td>
<td>Design</td>
<td>WW08: Initial design Done</td>
<td>Support IPSec configuration for remote site, proposal. Referenc: <a href="https://wiki.akraino.org/display/">https://wiki.akraino.org/display/</a> AK/IPSec+Design#IPSecDesign-IPSecRestAPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WW09: design done (to be reviewed)</td>
<td>WW10/11/12: 90%</td>
<td></td>
</tr>
<tr>
<td>SDEWAN E2E scenario</td>
<td></td>
<td></td>
<td></td>
<td>E2E demo for SDEWAN solution</td>
<td></td>
</tr>
<tr>
<td>manual steps</td>
<td></td>
<td>All</td>
<td>WW13-14</td>
<td>manual steps (create CNF, openwrt configuration for ipsec/NAT rule, manual connectivity test for ms) to verify E2E test scenari</td>
<td></td>
</tr>
<tr>
<td>auto test scripts to enable demo in ICN</td>
<td></td>
<td>All</td>
<td>WW15-16</td>
<td>leverage kud to setup 3 clusters (Hub, edge1, edge2)</td>
<td>use pre-defined yaml file (with network interface information and rules definition) to create Sdewan CNF use linux shell script to call CNF Rest API (e.g. update rule, restart service etc.) shell script to verify ms connectivity in different edge cluster</td>
</tr>
<tr>
<td>SDEWAN CNF Controller</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POC to verify CR &amp; CNF matching by label</td>
<td></td>
<td>WW17</td>
<td></td>
<td>R3.1</td>
<td>POC to verify the flow for n:m label matching between CR instances and CNF instances (e.g. a CR can apply to multiple CNF and a CNF can have multiple CR)</td>
</tr>
</tbody>
</table>
### SDEWAN CNF

#### Sdewan config Agent

#### Presentation

ICN Weekly meeting video recordings- Weekly Akraino ICN Engineering Meeting