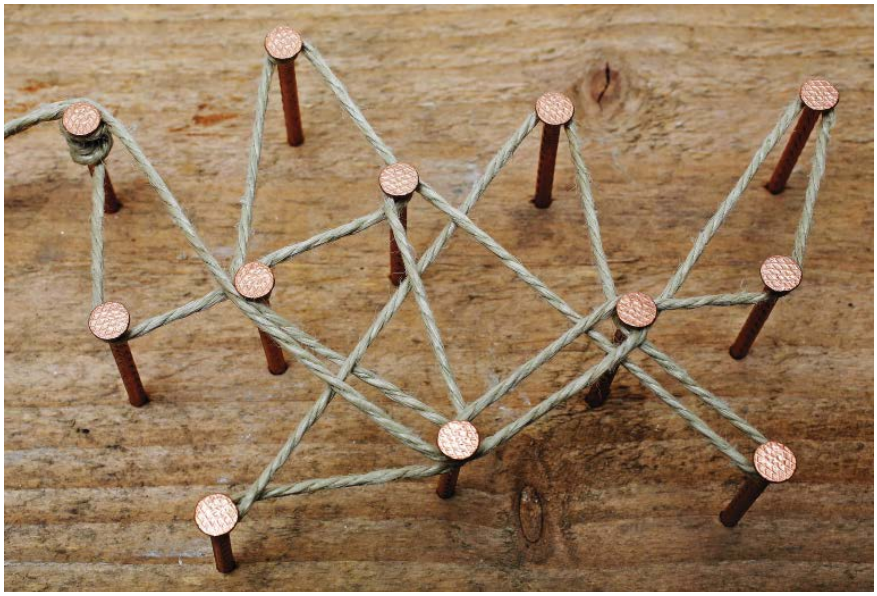


OpenStack-Neutron-Alternative Open Virtual Network

In Kontakt

Sebastian Biedler

In bestimmten Anwendungsszenarien stößt die SDN-Implementierung Neutron schnell an ihre Grenzen. OVN will hier eine leistungsfähige Alternative bieten.

Besonders in sehr großen OpenStack-Clouds oder einer dynamischen Umgebung mit extrem vielen Änderungen im Netzwerk gerät OpenStacks Neutron-Architektur schnell an ihre Grenzen. Cloud-Architekten und -Administratoren sind daher immer öfter im Netzwerkbereich auf der Suche nach einer Alternative zur Neutron-Architektur, da der klassische OpenStack-Neutron-Aufbau nicht den Anforderungen ihres Use Case genügt.

In solchen Fällen erweist sich die designbedingte Nutzung des Messaging Service meist als Quell ständigen Ärgers: Neutron ist mit all seinen Agenten der Hauptnutzer des Messaging Service (beispielsweise RabbitMQ) und auf dessen Funktionieren angewiesen. So gerät dieser Service leicht zum Single Point of Failure (SPoF) – auch hinsichtlich der Performance – beim Verarbeiten von Aktionen in OpenStack. Vor allem in größeren Cloud-Landschaften ist es unerlässlich, diesen Dienst genau auf den Einsatzfall zu optimieren. Die große Anzahl von Talks auf Konferenzen dazu unterstreicht die Komplexität des Themas.

Einem in diesem Umfeld aktiven Cloud-Architekten bieten sich zwar reichlich Alternativen, aber viele davon sind entweder sehr komplex, nicht produktionsreif oder fügen sich nur schwer in ein bestehendes Rechenzentrumsnetzwerk ein. So gehören die meisten Vertreter zu Controller-basierten SDN-Ansätzen (Software-defined Networking). Diese spielen ihre vollen Vorteile aber oft erst aus, wenn sie die vollständige Kontrolle über das Underlay- und das Overlay-Netzwerk haben.

In einer reinen Overlay-Netzwerkarchitektur, wie sie etwa OpenStacks Neutron-Architektur bereitstellt, gab es bisher kaum produktionsreife Alternativen, die auch konsequent weiterentwickelt werden. In diese Lücke stößt das OVN-Projekt (Open Virtual Network), das aus Open vSwitch hervorgegangen ist und sich als Ergänzung zu diesem versteht. Wie Open vSwitch steht auch OVN unter der Apache License 2.0 und lässt sich daher frei verwenden. Das Projekt möchte seine Software als Alternative zu Neutron im Overlay-Netzwerkbereich etablieren.

Aus der Vogelperspektive

Für ein grundlegendes Verständnis der Funktionsweise von OVN hier zunächst ein Blick auf die Architektur und die beteiligten Komponenten. Als Schnittstelle zwischen dem Cloud-Management-System (CMS) und der Northbound DB dient ein Treiber, der Letztere ansprechen kann (siehe Grafik). CMS meint in diesem Kontext OpenStack oder Kubernetes. Die Northbound DB speichert eine sehr abstrahierte Sicht des virtuellen Netzes, die der bei einem klassischen Neutron-Aufbau in der OpenStack-Datenbank gespeicherten ähnelt. Die abstrahierte Darstellung des Overlay-Netzes ist von Menschen les- und interpretierbar.

Ein eigener Kommandozeilen-Client realisiert die Interaktion mit der Northbound DB. Über den Befehl `ovn-nbctl` und seine Unterkommandos kann man sich Informationen aus der Northbound DB anzeigen lassen, Listing 1 zeigt einen Auszug als Beispiel. Außerdem lassen sich darüber Einträge in der Datenbank ändern beziehungsweise löschen. Zwar ist es im Normalfall nicht nötig, in der Datenbank manuell Änderungen vorzunehmen, aber so besteht zumindest die Option, sollte es im Fehlerfall nötig sein.

Die Southbound DB ist mit ihren gespeicherten Daten näher an der eigentlichen OVN-Sichtweise auf das Netz. Sie speichert Informationen zum physischen und logischen Netz und wie beide miteinander verknüpft sind. Diese Verknüpfung wird in sogenannten Logical Flows dargestellt, deren Konzept man eigens für OVN entwickelte. Die Notation des Logical Flows ist aus dem OpenFlow-Protokoll entlehnt [1].

Ein Logical Flow sieht einem per OpenFlow-Protokoll erstellten Flow ähnlich. Die in einem Logical Flow enthaltenen Informationen sind jedoch andere und beschreiben die Beziehung zwischen dem eigentlichen und dem virtuellen Netzwerk. Listing 2 zeigt den passenden Logical Flow des VM-Ports aus Listing 1.

Auch die Southbound DB bietet mit `ovn-sbctl` analog zur Northbound DB einen eigenen CLI-Client für die direkte Interaktion mit der Datenbank. Der OVN-Daemon `northd` hält die beiden Datenbanken zusammen. Er übersetzt auch die in der Northbound DB gespeicherten Informationen in ein Format, das sich in der Southbound DB speichern lässt.

Listing 1: CLI-Zugriff auf Northbound DB

```
ovn-nbctl show neutron-4b653d01-377b-4a72-ac43-d95c1c0ad492
switch 212d3b32-b002-4e51-b008-7439be7a51c0 (neutron-4 b653d01-377b-4a72-ac43-d95c1c0ad492) (aka testnetz)
  port d1f064ae-846c-403e-81a9-488272cb08b8
    addresses: ["fa:16:3e:a3:0a:18 192.168.10.16"]
  port 19fc4e2b-40dd-4de9-bcb0-44ec6ce695ee
    type: router
    router-port: lrp-19fc4e2b-40dd-4de9-bcb0-44ec6ce695ee
  port 467fb420-4c90-4bd4-a0f9-349adfe2aae5
    type: localport
    addresses: ["fa:16:3e:38:12:dd 192.168.10.2"]
```

Die vierte Komponente einer OVN-Umgebung ist der OVN-Controller. Die Bezeichnung Controller mag etwas irreführend erscheinen, da dieser Service nicht die gesamte Landschaft kontrolliert, sondern immer nur den jeweiligen Node (auch Chassis genannt), auf dem er läuft. Dieser Dienst läuft auf allen Compute Nodes und auch auf anderen Nodes, die über einen Open vSwitch verfügen. Er ist über eine TCP-Verbindung direkt mit der Southbound DB verbunden und empfängt von dort alle für seinen jeweiligen Node relevanten Informationen und Updates.

Die Informationen erhält er im erwähnten Logical-Flow-Format. Die übersetzt er dann in OpenFlow-Regeln, die er per Unix Domain Socket in die lokale OVSDB schreibt. Durch seinen Ursprung im Open-vSwitch-Projekt und die daraus resultierende enge Bindung an dieses Projekt lässt sich OVN nur in Verbindung mit dem OVS und nicht mit Switch-Alternativen wie der Linux Bridge einsetzen.

Neutron versus OVN – alles muss sich ändern

Da sich das OVN-Projekt zum Ziel gesetzt hat, eine echte Alternative zu sein, ist die Integration in OpenStack relativ einfach. OVN bietet in einer OpenStack-Landschaft klassische L2-/L3-Funktionen. Was einem OpenStack-Administrator wahrscheinlich zuerst auffallen wird, ist der Umstand, dass sämtliche Neutron-Agenten entfallen. OVN nutzt nur die Neutron-API und somit ändert sich für einen Benutzer der Cloud augenscheinlich nichts. Auch wird auffallen, dass der Netzwerk-Node nicht mehr benötigt wird, da im Prinzip alles auf den Compute Nodes stattfindet. Auch die bekannten Namespaces entfallen, da für OVN nur Interfaces und Flow-Regeln existieren.

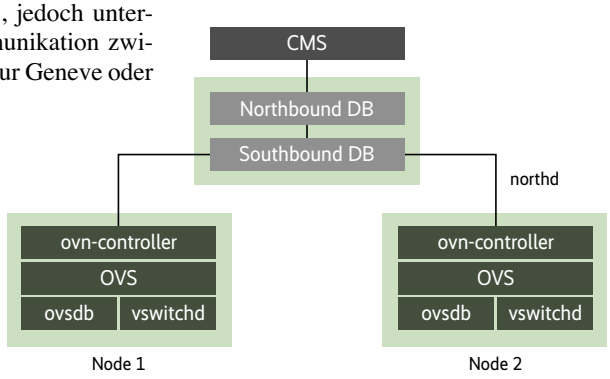
Im Unterschied zu Neutron-Landschaften setzt OVN auf Geneve als Tunnelprotokoll. Es ist zwar möglich, andere Protokolle wie VXLAN zu nutzen, um mit sogenannten ToR-Switches (Top of the Rack) zu kommunizieren, jedoch unterstützt OVN in der Kommunikation zwischen den Hypervisoren nur Geneve oder das STT-Protokoll (Stateless Transport Tunneling).

Das liegt unter anderem daran, dass nur diese beiden Protokolle Metadaten, die größer sind als 32 Bit pro Paket, und zufällige UDP- oder TCP-Source-Ports implementieren. Der Mehrbedarf an Informationen im Header erklärt sich damit, dass eine OVN-Tunnelverbindung immer versucht, Informationen über den Ingress-Egress-Port sowie über den Datapath eines Pakets mitzusenden. Die zufälligen Source-Ports erlauben eine bessere Verteilung der Tunnelverbindungen, sollte im Underlay-Netzwerk ECMP (Equal-Cost Multi-Path Routing) zum Einsatz kommen.

Eine andere Änderung betrifft den Aufbau der Firewall. Das von Neutron bekannte FWaaS (Firewall as a Service) entfällt in OVN. Das hat zum einem damit zu tun, dass dieser Service vom Neutron-l3-Agent gestartet und verwaltet wird und eine OVN-Architektur ohne Neutron Agents auskommt. Zum anderen erfolgt die Abbildung von Firewall und Securitygruppen nach einem völlig anderen Ansatz: Sie werden innerhalb von Open vSwitch als Flow dargestellt.

Auch wenn sich durch die Nutzung der Neutron-API für den Benutzer nichts ändert, werden im Hintergrund sogenannte ACLs (Access Control List) für jeden einzelnen Port angelegt. Sie enthalten alle Informationen, wie ein Paket, das diesen Port erreicht, zu behandeln ist. Diese ACLs zu einem Port findet man in der Northbound DB sowie auch im Logical-Flow-Format in der Southbound DB. Diese schreibt der passende OVN-Controller dann in die lokale Datenbank OVSDB.

Seit dem Erscheinen von Octavia lassen sich in OpenStack mit OVN auch LBaaS (Load Balancer as a Service) anbieten. Allerdings muss man wissen, dass im Einsatz mit OVN hierzu zwei Möglichkeiten für die Realisierung existieren. Der erste Weg wäre, LBaaS wie in einem reinen Neutron-Umfeld über die von Octavia bereitgestellte VM zu nutzen. Alternativ kann man den eigentlichen OVN-Treiber verwenden. Diese Option bietet im Vergleich zum ersten Weg verschiedene Vor- und Nachteile. Die Vorteile bestehen darin, dass zum einen keine VM benötigt und zum anderen der Load Balancer schneller aufgebaut wird. Ein weiterer Vorteil ergibt sich daraus, dass das OVN-Netzwerk für VM und Container zur Verfügung stehen kann. Somit kann man den Octavia-OVN-Treiber in Verbindung mit Kuryr Kubernetes nutzen. Allerdings existieren auch ein paar Einschränkungen gegenüber dem VM-basierten Aufbau. OVN beherrscht nur Load Balancing bis L4, daher funktioniert L7 Load Balancing nicht. Auch ist in OVN kein Health-Monitoring implementiert. Der dritte Nachteil ist, dass OVN nur Listeners und Pools des gleichen Protokolls miteinander verbinden kann. Das heißt, ein TCP Listener benötigt einen TCP Pool und für UDP gilt dieselbe Einschränkung.



Schematische Darstellung der OVN-Architektur mit all ihren Komponenten

Zentral oder dezentral kommunizieren

Eine wichtige Frage, die man beim Design eines OpenStack-Clusters beantworten muss, ist die nach der Kommunikation mit der Außenwelt. OpenStack mit OVN eröffnet hier mehrere Möglichkeiten. Es bietet zwei Architekturen für die Kommunikation mit der Außenwelt: die Centralized- und die Distributed-Floating-IP-Architektur.

Erstere gleicht dem Ansatz von Neutron-Legacy. Das heißt, der Router und auch die genutzten Floating-IPs zu einem OpenStack-Projekt befinden sich auf einem Node. Der gesamte externe Verkehr, sei es per Floating-IP oder SNAT (Source Network Address Translation), fließt über diesen zentralen Router. Diese Methode ist auch die OVN-Standardarchitektur.

Bei der Distributed-Floating-IP-Architektur ändert sich das. Die Floating-IPs befinden sich dabei nicht zentral an einer

Listing 2: CLI-Zugriff auf Southbound DB

```
ovn-sbctl dump-flows | grep d1f064ae-846c-403e-81a9-488272cb08b8
  table=0 (ls_in_port_sec_l2 ), priority=50, match=(inport="d1f064ae-846c-403e-81a9-488272cb08b8" &&
    eth.src={fa:16:3e:a3:0a:18}), action=(next;)
  table=1 (ls_in_port_sec_ip ), priority=90, match=(inport="d1f064ae-846c-403e-81a9-488272cb08b8" &&
    eth.src=fa:16:3e:a3:0a:18 && ip4.src=0.0.0.0 && ip4.dst=255.255.255.255 && udp.src=68 &&
    udp.dst=67), action=(next;)
  table=1 (ls_in_port_sec_ip ), priority=90, match=(inport="d1f064ae-846c-403e-81a9-488272cb08b8" &&
    eth.src=fa:16:3e:a3:0a:18 && ip4.src={192.168.10.16}), action=(next;)
  table=1 (ls_in_port_sec_ip ), priority=80, match=(inport="d1f064ae-846c-403e-81a9-488272cb08b8" &&
    eth.src=fa:16:3e:a3:0a:18 && ip), action=(drop;)
  table=2 (ls_in_port_sec_nd ), priority=90, match=(inport="d1f064ae-846c-403e-81a9-488272cb08b8" &&
    eth.src=fa:16:3e:a3:0a:18 && arp.sha=fa:16:3e:a3:0a:18 &&
    arp.spa={192.168.10.16}), action=(next;)
  table=2 (ls_in_port_sec_nd ), priority=80, match=(inport="d1f064ae-846c-403e-81a9-488272cb08b8" &&
    (arp||nd)), action=(drop;)
  table=11 (ls_in_arp_rsp ), priority=100, match=(arp.tpa=192.168.10.16 && arp.op=1 &&
    inport="d1f064ae-846c-403e-81a9-488272cb08b8"), action=(next;)
```

Stelle, sondern verteilt auf den jeweiligen Compute Nodes, wo sich auch die zugehörige VM befindet. Somit ähnelt diese Architektur dem von Neutron bekannten DVR (Distributed Virtual Router). Die Nutzung von Distributed-Floating-IPs lässt sich global in der Neutron-Server-Konfigurationsdatei einschalten, indem man den Schalter `ovn / enable_distributed_floating_ip` auf `true` setzt. OVN legt daraufhin Flow-Regeln auf dem Compute Node an, auf dem die VM liegt, und bindet die Floating-IP an diese Regeln. Somit fließt der externe Verkehr dieser VM direkt in das externe Netzwerk und umgeht so den SNAT-Router.

Für welche Architektur man sich entscheidet, hängt vom jeweiligen Nutzungsszenario der Landschaft ab. Aber auch der geplante oder vorhandene Aufbau des Underlay-Netzwerkes und die damit verbundene Frage, ob das externe Netz auch an allen Nodes verfügbar ist, haben großen Einfluss auf die Einscheidung.

Bei beiden Architekturen sind Nodes für die Gateway-Router zu definieren. Mit dem Kommando

```
ovs-vsctl set open . ⤸
                external-ids:ovn-cms-options= ⤸
                          enable-chassis-as-gw
```

erklärt man den Node, auf dem das Kommando ausgeführt wird, zu einem sogenannten Gateway Node. Auf diesem kann OVN die Gateway- beziehungsweise SNAT-Router platzieren. Dabei spielt es keine Rolle, ob es sich um einen Compute Node oder separaten Node handelt.

Egal, welche der beiden Architekturen in einer Landschaft zum Einsatz kommt, es stellt sich immer die Frage nach der Ausfallsicherheit des SNAT-Routers. OVN aktiviert den L3HA-Modus automatisch, sobald mehr als ein Gateway Node in der Landschaft existiert. Der Router-Failover wird ausgelöst, sobald der Gateway Node vom Netzwerk getrennt wird, wenn der OVS ausfallen sollte und bei einem Ausfall des OVN-Controllers auf dem Node. Das L3HA nutzt einen sogenannten Controller-unabhängigen Aktiv-passiv-Ansatz. Ein Router wird als Gateway-Router genutzt, und auf den anderen Gateway Nodes befinden sich inaktive Kopien dieses Routers. Die Gateway Nodes werden mithilfe des BFD-Protokolls (Bidirectional Forwarding Detection) von allen OVN Nodes überwacht. Es handelt sich dabei um ein verbindungsorientiertes Protokoll, das sehr schnell Zustandsänderungen in IP-Netzen erkennen kann.

Um im Fehlerfall zu ermitteln, welcher inaktive Router übernehmen soll, erhält jeder Router beim Erstellen der Kopie eine Priorität zugewiesen und gibt diese auf seinem Node bekannt. Es gilt das Prinzip, dass der Router mit der höchsten Priorität auf einem funktionieren Gateway Node der aktive Router ist. Das führt dazu, dass, nachdem ein ausgefallener Node wieder zurückkehrt, der aktive Router wieder auf diesen zurückschwenkt. Dieses Verhalten soll Engpässe verhindern. Das ganze Handling zeigt, dass OVN den verschiedensten Anforderungen hinsichtlich der Verteilung des Netzwerkverkehrs und der Ausfallsicherheit gerecht werden kann.

Vorteil OVN?

Nach der Betrachtung der OVN-Architektur im Allgemeinen und der Besonderheiten im Zusammenspiel mit OpenStack stellt sich die Frage nach den genauen Vor- und Nachteilen gegenüber dem klassischen Neutron-Aufbau. Ein großer Vorteil ist, dass der Messaging-Service außen vor bleibt. Wie eingangs erwähnt, hängt der klassische Neutron-Aufbau völlig von diesem Dienst ab, und er kann sich leicht als Flaschenhals und SPoF erweisen. Zwar kann sich der `northd`-Dienst mit der Northbound DB und Southbound DB ebenfalls als ein SPoF herausstellen, allerdings lässt sich dieser Service sehr leicht hochverfügbar aufbauen, beispielsweise mit keepalive. Eine Möglichkeit, den `northd` active-active zu betreiben, wollen die Entwickler in den kommenden OVN-Versionen implementieren.

Ein weiterer Vorteil besteht darin, dass die Migration von Neutron mit Agenten auf OVN sehr leicht gelingt. Im Rahmen des OVN-Projektes entstand ein Migrationsskript, das diese Aufgabe erledigt. Größter Pluspunkt für OVN ist aber, dass man mit OVN eine Overlay-Netzwerkarchitektur aufbauen kann, die sowohl OpenStack- als auch Containerlandschaften (wie Kubernetes) miteinander verbindet. Somit kann ein Cloud-Administrator eine Netzwerkarchitektur aufbauen, die verschiedenste Plattformen bedient, und so den Verwaltungsaufwand verringern.

Wie bereits im Zusammenhang mit LBaaS angedeutet, fehlen OVN noch einige Features, die von einem Produktivsystem erwartet werden. Auch sollte man sich bewusst sein, dass OVN erst vor wenigen Releases einen Stand erreicht hat, bei dem eine Firma wie Red Hat dem Projekt das Vertrauten schenkt. Dennoch sollte jeder Cloud-Administrator, der vor der Aufgabe steht, eine neue Landschaft zu planen, OVN zumindest in seine Überlegungen einbeziehen. (avr@ix.de)

Quellen

[1] Zdravko Bozakov; Netzwerktechnik; Alles fließt; OpenFlow – eine Softwareschnittstelle zur Netzprogrammierung; *iX* 8/2012, S. 112

[2] Sriam Subramanian, Sreenivas Voruganti; Software-Defined Networking (SDN) with OpenStack; Packt Publishing Ltd., 1st Edition 2016

[3] Omar Khedher, Chandan Dutta Chowdhury; Mastering OpenStack; Packt Publishing Ltd., 2nd Edition 2017

[4] Weiterführende Dokumentation zu OVN mit OpenStack ix.de/ix1906064

Sebastian Biedler

ist seit 2015 bei der B1 Systems GmbH als Linux Consultant und Trainer beschäftigt. Er befasst sich seit fünf Jahren mit Themen rund um OpenStack sowie auch aus privatem Interesse mit den Schwerpunkten Software-defined Networking und Network Function Virtualization.