Utility Computing and Cloud Networking

Delivering “Networking as a Service”
Overview

- Utility Computing
- OpenStack Virtual Networking
- Network Functions Virtualization
Utility Computing
Utility Computing: Everything as a Service

- Goal of Utility Computing:
  - Deliver computation, storage, and networking/communication “as a service”
  - Like electricity, gas, water

- What does “as a service” mean?
  - You pay for only as much as you use
    - MIPS/Gigabytes/Bytes per second
  - You deploy your application on a virtualized compute/storage/networking platform
  - Someone else, the cloud provider, takes care of maintaining the platform

Advantages of “as a Service”

- No need for a large, up front capital investment in hardware and system software infrastructure
  - Your cost scales as your business grows

- Your operating costs are reduced by the cloud provider’s economy of scale
  - Reduction in system administration costs for hardware and system software
  - Elimination of additional costs for power, cooling, etc.

- Equipment utilization improves through multiple customers using the same virtualized platform
  - With proper management, reduction in energy use

Source: http://dyamar.com/images/software-as-service.jpg
First Major Success: Three Tier Apps

- Client (top) Tier is Web browsers on Internet
  - Ajax programming for UI
  - HTTP/HTTPS to connect with cloud provided Application Tier

- Application (middle) Tier is Web server with business logic
  - Stateless or “soft” state
    - Can be easily regenerated
    - Example: customer shopping cart
  - Autoscaling starts new instances of Application Tier VMs as load ramps up and removes them as load ramps down
  - Load balancing distributes load between Application Tier VM replicas

- Database (bottom) Tier contains all the state
  - Can be SQL database
  - Also NoSQL (graph)
  - JDBC or some other protocol to connect Middle Tier

Networking requirements for 3 Tier applications are straightforward
Networking for 3 Tier Apps

- Web Clients use TCP to access Web application
- Load Balancer evens load on application instances
  - DNS or other technique used for load balancing
  - Keeps track of load
  - Tells autoscaler when load exceeds threshold
- Autoscaler brings up a new instance when required
- Database is replicated, highly available
  - Doesn’t matter which instance the application accesses
  - UDP used for JDBC
- In principle, virtual networks not required
  - Just TCP, DNS
Next Challenge: Enterprise LAN as a Service

- Wanted: Open source API allowing tenant operations on virtual network abstractions
  - Create
  - Read
  - Update
  - Delete

Source: wikimedia.org
OpenStack Virtual Networking
OpenStack History

2010
- NASA and Rackspace partner to form OpenStack
- Inaugural Design Summit in San Antonio attracts around 200 people

2011
- Diablo first release usable by applications

2012
- Rackspace offers commercial OpenStack cloud service
- Openstack Foundation formed under umbrella of Linux Foundation
- OpenStack Networking introduced in Folsom release
  - Known as “Quantum” at that time but now “Neutron”
  - Led by Nicira, now VMWare

2013
- First International Design Summit in Hong Kong attracts over 5000 attendees
  - Fastest growing open source project in history!
- First additional service, Load Balancing as a Service (LBaaS), introduced in Grizzly release
- Neutron ML2 driver architecture in Havana release

2014
- Distributed Virtual Router and IPv6 support in Juno release
OpenStack Cloud Controller
High Level Architecture

Source: openstack.org
OpenStack Cloud Controller Components

- **API Server**
  - Enables tenant command and control of virtualized compute, storage and network resources
  - Represents cloud resources toward tenants as abstractions

- **Message Queue**
  - Brokers interaction between compute, storage, and networking resource managers

- **Resource Managers**
  - Manages the partitioning of compute, storage, and networking resources between the cloud’s tenants
  - Each resource has at least one resource manager
    - Storage has several
API Server Detail

- API endpoints are HTTP servers
  - Each individual service (Compute, Networking, etc.) has a separate API server

- Operations handled by API endpoints:
  - Authentication, authorization
  - Dispatching tenant requests to appropriate resource manager
    - Some API servers also handle resource management

- Programming model:
  - REpresentational State Transfer (REST) RPC
  - Stateless: no pointers or other state, uuids instead

- Resource representation:
  - Typed abstractions for major resource classes (VMs, networks, etc.)
    - Object properties represented as Attributes
    - Some resource classes can be subclassed

- Operational model:
  - Create, Read, Update, Delete (CRUD)
  - Some operations complete asynchronously
    - Example: allocating a tenant network may take time because lots of devices may need to be touched
Make me a new tenant virtual network!

1) REST Create Call
2) Is she OK?
3) Yup!
4) Make a new tenant network for Ampelfrau!
5) Networking service, make a network for Ampelfrau!
6) That’s me!
7) (goes off and does it)
8) Ampelfrau network is now active
9) That’s done!
10) Done!

Sources:
- Ampelfrau: wikimedia.org
- Keystone logo: github.org
- Bus: metrotransit.org
OpenStack System Component Interaction

Keystone
- Provides auth

Cellometer
- Monitor

Horizon
- Provides UI

---

Glance
- Fetchs images via
- Registers guest images in
- Boots database instances via
- Provides images

Governor
- Orchestration

Neutron
- Provides PXE network for

Ironic
- Orchestration

We will focus here!

Swift
- Stores images in
- Saves data or job binary in
- Orchestrates clusters via

Cinder
- Backups volumes in
- Provides volumes to

Trove
- Provision, operation and management

Sahara
- Orchestrates clusters via

Source: http://docs.openstack.org/admin-guide-cloud/content/conceptual-architecture.html
Basic Neutron Abstractions

- **Network**
  - Virtual isolated Layer 2 broadcast domain with connections to specific VMs and other networks
  - Networks are subclassed (later...)

- **Subnet**
  - A Network supporting a specific, routable IP prefix
  - A Subnet can also optionally have following attributes:
    - A gateway,
    - A list of DNS name servers
    - Host routes
  - Subnet attributes pushed to VMs instantiated on the subnet

- **Port**
  - Virtual port on a logical network switch
  - MAC addresses and IP addresses associated with a Port
    - If IP addresses, then Port is associated with a subnet
  - Ports are used to attach VMs and other Networks to their owning Network

Note: All calls are REST (HTTP RPC) with attributes/responses provided as JSON or XML Request/Response bodies.
Network API Calls

**POST** /v2.0/networks
Create network
Creates a network.

**POST** /v2.0/networks
Bulk create networks
Creates multiple networks in a single request.

**GET** /v2.0/networks/{network_id}
Show network
Shows information for a specified network.

**PUT** /v2.0/networks/{network_id}
Update network
Updates a specified network.

**DELETE** /v2.0/networks/{network_id}
Delete network
Deletes a specified network and its associated resources.

Source: developer.openstack.org
Subnet API Calls

GET /v2.0/subnets  List subnets
Lists subnets to which the specified tenant has access.

POST /v2.0/subnets  Create subnet
Creates a subnet on a specified network.

POST /v2.0/subnets  Bulk create subnet
Creates multiple subnets in a single request. Specify a list of subnets in the request body.

GET /v2.0/subnets/{subnet_id}  Show subnet
Shows information for a specified subnet.

PUT /v2.0/subnets/{subnet_id}  Update subnet
Updates a specified subnet.

DELETE /v2.0/subnets/{subnet_id}  Delete subnet
Deletes a specified subnet.

Source: developer.openstack.org
<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
</table>
| GET    | `/v2.0/ports` | List ports  
Lists ports to which the tenant has access. |
| POST   | `/v2.0/ports` | Create port  
Creates a port on a specified network. |
| POST   | `/v2.0/ports` | Bulk create ports  
Creates multiple ports in a single request. Specify a list of ports in the request body. |
| GET    | `/v2.0/ports/{port_id}` | Show port  
Shows information for a specified port. |
| PUT    | `/v2.0/ports/{port_id}` | Update port  
Updates a specified port. |
| DELETE | `/v2.0/ports/{port_id}` | Delete port  
Deletes a specified port. |

Source: developer.openstack.org
Neutron Architecture and Interconnection

Plug-ins support different vendor equipment and virtualization types

Heat Orchestration Engine calls into Neutron to connect VM vNIC to a virtual network

Neutron calls into Keystone Authentication Service to ensure tenant is authorized to operate on a virtual network.

Horizon GUI calls into Neutron to CRUD Neutron tenant networks.

Other subsystems call into Neutron for various services.

Source: openstack.org
Neutron Plugin Architecture

- Core + Extension REST APIs
- Message queue for communicating with Neutron agents
- Core and service plugins
- Hardware and software devices
- Different vendor core plugins
- Service plugin backend drivers

Source: http://de.slideshare.net/lewtucker/open-stack-atlanta-2014tucker
Implementing the Network Abstraction

- Each Network abstraction has below it a virtual network and a physical network
- A *segment* is a section of virtual network that is implemented by a particular overlay protocol
  - Examples:
    - VLAN
    - VxLAN
    - GRE
    - ...
- Physical network support provided by particular hardware and software devices:
  - Software Examples:
    - OpenVSwitch
    - Cisco Nexus 1000V
    - OpenDaylight SDN Controller
    - ...
  - Hardware Examples:
    - Cisco Nexus 3000 ToR Switch
    - Pica8 P-3922 OpenFlow ToR Switch
    - ...
Example: GRE Segment over OVS and Pica8
Function of Plugins

Plugins provide the bridge between the Neutron API and the network protocols and hardware that implement the logical and physical network.

Plugins convert the API calls into:
- Updates to the Openstack databases indicating a tenant virtual network has been created/updated/deleted
- Specific hardware or software device configuration calls
- Protocol operations for setting up a virtual network

Plugins are like drivers in an operating system.
Three Types of Code in Plugins

- Boilerplate code that deals with OpenStack related issues
  - Database updates
  - Authentication
  - ...

- Protocol specific code for implementing a particular *type* of virtual network segment
  - Encapsulating and decapsulating packets in GRE headers
  - Tagging packets with a specific VLAN tag
  - ...

- Device specific code that implements the virtual network for a particular hardware or software *mechanism*
  - OpenFlow and OVSDB protocol for implementing GRE tunnels on OVS
  - Cisco 3000 CLI commands for configuring VLAN trunking
  - ...

Every hardware or software device supplier must write and support duplicate code!
ML2 Architecture

ML2 driver – only deals with the OpenStack overhead
- Database updates, etc.

Type driver – only deals with the details of implementing a particular segment type
- Example: GRE tunnels

Mechanism driver – only deals with the details of a particular supplier’s hardware or software device
- Example: OpenVSwitch

Source: http://de.slideshare.net/lewtucker/open-stack-atlanta-2014tucker
SDN and Neutron

- From Neutron’s viewpoint, an SDN controller looks like a device
  - Details of actual physical network control hidden under controller northbound API
- From the SDN controller’s viewpoint, Neutron looks like another northbound application
  - Specific northbound API crafted for supporting Neutron
- ML2 mechanism driver code simply becomes a passthrough to the SDN controller API
Neutron with OpenDaylight ML2 Provider

Data Center Network

OpenFlow Enabled Devices

Additional Virtual & Physical Devices

Open vSwitches

OpenStack Neutron

OpenStack Service

OVSDB Neutron

OpenFlow

OVSDB
How to Get Out to the Internet?

- Neutron supports a basic router abstraction
- Router abstraction supports a NAT function
  - NAT function located on uplink gateway ports
  - NAT creates one-one mapping between a public IP address and a private IP address on the Neutron Network
- Connecting a Router to a Network provides connectivity to outside Subnets
  - Including the Internet
- How is the Internet represented in Neutron?
Subclasses of Neutron Networks

- Tenant network subclass
  - Created by any tenant
  - Tenant can only specify details through the Neutron API
  - Implementation details hidden

- Provider network subclass
  - Created only by a user with administrative privileges
  - Details of how implemented can be specified
  - Usually match some existing network in the data center

Provider Network allows representation of Internet as a Neutron object
OpenStack Datacenter Network Design

Provider Network
VID=01 – Gehfrau
VID=02 – Gehmann
VID=03 – Stopfrau
...

Tenant Network for Gehfrau

Tenant Network for Gehmann

Provider Edge (PE) Router

Customer Edge (CE) Router

Neutron Server

Source: wikimedia.org
Network Functions
Virtualization
Operator Network Infrastructure as a Service?

- Can public operator network services be put into the cloud?
  - Example public operators: Deutsche Telekom, ATT, Vodafone/Verizon

- Example services offered by public operators
  - IP Multimedia Subsystem for voice/media calls
  - Authentication, Authorization, Accounting, Billing
  - Enterprise Virtual Private Networks (VPN)
  - Service assurance, operations support, network management
  - For wireless networks, the whole Evolved Packet Core control/data plane

ETSI Network Functions Virtualization (NFV) addresses virtualization of operator networks
Goals of NFV

- Rapid service innovation through software-based deployment and operationalization of end-to-end services
  - Customers can compose services through service portal
- Improved operational efficiencies resulting from common automation and operating procedures
  - “Zero touch” deployment
- Reduced power usage achieved by migrating workloads and powering down unused hardware
  - Managing data centers for power efficiency
- Standardized and open interfaces between network functions and their management entities so that such decoupled network elements can be provided by different players
  - Open source, startup contributions to market
- Greater flexibility in assigning Virtualized Network Functions (VNFs) to hardware
- Improved capital efficiencies compared with dedicated hardware implementations
  - Deploy on virtualized white box server platform

**SDN and NFV**

- NFV is about deploying operator network services in a virtualized environment.
- SDN is about separating the control and data planes and controlling routing and forwarding from a centralized controller.  
  - Note: Some researchers include SDN and NFV under SDNv2.
- NFV can be deployed on an OpenFlow or non-OpenFlow networks.

**Key requirement for NFV is virtualization of compute and networking.**
NFV Definitions

Network Function
- Functional building block with well defined interfaces
- Example: router

Virtualized Network Function
- Software implementation of network function that can be deployed on a virtualized platform
- Example: software router

Service chain
- A directed graph of middleboxes through which customer packets must flow before being delivered to the end host
NFV Architecture

Source: ETSI NFV Working Group
What are Service Chains?

- Middleboxes perform specific functions on packets
- Operator configures a service chain for particular customers
- Today, services are deployed as specific hardware boxes
  - Inflexible
  - Difficult to scale
Can Cloud Networking Support VNFs?

Problem: most VNFs have state, cloud is designed for stateless applications

- Example: IP Multimedia subsystem keeps state on outstanding voice and media sessions
- State must be highly accessible and cannot be regenerated quickly
- When a VM moves, need to move state along
- Solution: Need to re-engineer application

Problem: software implementations of networking VNFs don’t achieve line speed forwarding

- Example: software routers abandoned in the early 1990s for ASICs
- Solution: Specialized system software for forwarding applications
Software Routers: Intel DPDK Architecture

Source: embedded.comunities.intel.com
Software Routers: DPDK Performance

Broadband Network Gateway (BNG) can be implemented on a virtualized platform.
Summary

- Utility computing provides compute, networking, and storage “as a service”
- Three tier applications don’t need virtual networks but more advanced applications do
  - Enterprise LAN as a service
- OpenStack Neutron provides virtual networking to tenants
  - Abstractions are Network, Subnet, Port
- Neutron supports a plugin architecture
  - ML2 plugin separates boilerplate, type and mechanism code
- Neutron Provider networks support access to outside
- Operator network services virtualized through ETSI NFV
Acknowledgements

- Lew Tucker, Cisco