Transport … and much more.
Network: control plane

SDN Architecture

Controller

Network Node

Network Node

Network Node

Network Node

Control Plane

Data Plane

http://aryaka.com/
Network: control plane

- Global network view
- Optimal flow management decisions
- Functional flexibility
Network services
A game of strategies
Network services: the past ...

- Specific manufacturer interfaces
- Limited competition
- Slow & error prone
- Little automation
Network services: the present ...

We need faster and more flexible deployment.
Network services: ... and the future

Forwarding Pipeline

NETWORK VIRTUALIZATION
Single Logical L2-L7 Chassis

TRAFFIC STEERING
VLAN X Switch
VLAN Y Switch
Appliance CONTEXT 4
Router VRF 8

Receive Traffic

Forward to Destination

COMPLEX

SIMPLE
Network Function Virtualization (NFV)

- Implementing network function in software
- Dynamic chaining (composition)
NFV and SDN: Controller applications

- Independent module per functionality
  - Bandwidth provisioning
  - Load balancing
  - Monitoring
  - etc.
NFV and SDN: Controller applications

Example: Firewall as a service
NFV and SDN: Controller applications

- Independent module per functionality
  - Bandwidth provisioning
  - Load balancing
  - Monitoring
  - etc.

- Shared infrastructure & resource

- Isolation
Shared everything
Service description

- Service characteristics
  - qualitative (service type)
  - quantitative (performance metrics)

- Resource constraints
  - Hardware and software
  - Accurate estimations

- Format & interpretation
  - For automatic deployment

OpenStack Heat Template (for deployment of VMs)

```yaml
heat_template_version: 2015-04-30

description: Simple template to deploy a single compute instance

parameters:
  key_name:
    type: string
    label: Key Name
    description: Name of key-pair to be used for compute instance
  image_id:
    type: string
    label: Image ID
    description: Image to be used for compute instance
  instance_type:
    type: string
    label: Instance Type
    description: Type of instance (flavor) to be used

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      image: { get_param: image_id }
      flavor: { get_param: instance_type }
```
Service decomposition

Modularity and compositions are the key to scalability & performance

What is the proper level to decompose functionality?
Service decomposition

- Monolithic function
  - inflexible and cumbersome
  - hard to program, debug, tests, enhance

- Fine-grained partitioning
  - functionality overlap and overhead
  - easier to program, debug, test; reuse
Service composition

M1  M2  M3
SDN Controller

Application 1: Content
Application 2: Payment

Sequential

SDN Controller

M1

M2

M3

client

Application 1: Content
Application 2: Payment

M1

M3
Service composition

**Sequential**

Client → M1 → M2 → Application 1: Content

Client → M1 → M3 → Application 2: Payment

**Parallel**

Client → M2 → Application 1: Content

Client → M1 → Application 3: Delivery

Client → M3 → Application 3: Delivery
Service composition: Languages

- Network policy:
  - instructive set of rules
  - instructs composition of services
  - translates to flows processing rules

```
Network policy
src >> LB >> dst
src >> FW.BW >> dst
```

```
OpenFlow rules
dstip = 5.6.7.8 → path 1
dstip = 5.6.7.8 → path 2
```
Service composition: Languages

- Network programming languages: the set of constructs instructing how network policies can be written

- Languages
  - Frenetic >> Pyretic
  - Merlin, Nettle, Maple

Merlin syntax

\[ \text{loc} \in \text{Locations} \]
\[ \text{t} \in \text{Transformation Functions} \]
\[ \text{pol} ::= (s_1; \ldots; s_n) \quad \text{Policy} \]
\[ s ::= q \ p \ \Rightarrow \ e \ \text{at} \ r \quad \text{Statement} \]
\[ q ::= \forall \ | \ \exists \quad \text{Quantifier} \]
\[ p ::= p_1 \ \text{and} \ p_2 \ | \ p_1 \ \text{or} \ p_2 \ | \ !p_1 \quad \text{Predicate} \]
\[ e ::= . \ | \ c \ e \ | \ e | e \ | \ ! e \quad \text{Path Expression} \]
\[ r ::= \max(n)\? \ | \ \min(n)\? \quad \text{Rate} \]
The challenges

Control Applications

SDN Controller

Control channel

Feedback loop

Local CPU/NPU

Switching fabric

How to describe a service?

How to decompose services?

How to compose services?
Placement & aggregation

- Single VM per service, best placement points
- Multiple services in VM
- Physical infrastructure
  - Software only or hybrid platforms
Placement & aggregation

- Placement decisions are driven by:
  - resource availability
  - housing availability and cost
  - energy and cooling
  - maintenance needs
  - enterprise policy
  - security and privacy
  - other regulatory issues
  - ...

ACN Network Functions Virtualization
The challenges

Control Applications
- SDN Controller
- Control channel
- Local CPU/NPU
- Switching fabric

Feedback loop

How to describe a service?
Where to deploy?
Should we aggregate?
How to decompose services?
How to compose services?
Monitoring & Troubleshooting

- Large data sets due to personalised service chains
- Privacy & security constraints
- Automation needs proper interfaces
Scalability

- Dynamic chaining
  - Per subscriber, per application, per flow

- Signalling overhead

- Multi-tenancy & resource sharing
The challenges

Control Applications

SDN Controller

Control channel

Local CPU/NPU

Switching fabric

Feedback loop

How to describe a service?

How to decompose services?

Where to deploy?

Should we aggregate?

How to compose services?

What is an error?

Error handling

What to monitor

Where to monitor

What is the performance?

Does it scale?
Shared everything

no
The importance of being isolated.
Multi-tenancy: recall Network Virtualization

- Logical slices on top of physical network

- Shared network resources
  - Topology
  - Bandwidth
  - Device CPU
  - Forwarding table entries

Source: ONF
Multi-tenancy: recall Network Virtualization

- Slice ownership
  - Different administrative units
  - Different customers (cloud)
  - Different applications
Multi-tenancy: recall Network Virtualization

- Slice ownership
  - Different administrative units
  - Different customers (cloud)
  - Different applications

- Slice control
  - Flowspace
  - Isolation

Source: ONF
Multi-tenant: FlowVisor
Multi-tenancy: FlowVisor

- ‘Master controller’ concept
- Policy-driven switch instructions
- Central coordination of OpenFlow messages
- Isolation
Will this all work?

- Hardware accelerated services
- Software implementations generally slower
- Selective service migration
Reading

- Wolfgang John et al., “Research Directions in Network Service Chaining”
- Nate Foster et al., “Languages for Software-Defined Networks”
- Rob Sherwood et al., “FlowVisor: A Network Virtualization Layer”
Contact information and credits

ETH Zurich
Systems Group
Universitätstrasse 6
8092 Zürich

www.ethz.ch/en.html

© ETH Zurich, February 2016