TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN MẠNG MÁY TÍNH VÀ VIỄN THÔNG

TRẦN THÀNH CÔNG - NGUYỄN ĐỨC DUY

TÌM HIỂU NFV VÀ TRIỂN KHAI HỆ THỐNG THỬ NGHIỆM

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

TP. HCM, 2017

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN MẠNG MÁY TÍNH VÀ VIỄN THÔNG

TRẦN THÀNH CÔNG – 1312057
NGUYỄN ĐỨC DUY – 1312084

TÌM HIỂU NFV VÀ TRIỂN KHAI HỆ THỐNG THỬ NGHIỆM

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

GIÁO VIÊN HƯỚNG DẪN
ThS. LÊ NGỌC SƠN

KHÓA 2013 – 2017

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Khóa luận đáp ứng yêu cầu của Khóa luận cử nhân CNTT.

Tp.HCM, ngày ..... tháng ..... năm 2017

Giáo viên hướng dẫn

NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Khóa luận đáp ứng yêu cầu của Khóa luận cử nhân CNTT.

Tp.HCM, ngày ..... tháng ..... năm 2017

Giáo viên phản biện

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn Bộ môn Mạng máy tính và Viễn thông, Khoa Công Nghệ Thông Tin, trường Đại Học Khoa Học Tự Nhiên TP.HCM đã tạo mọi điều kiện thuận lợi cho chúng em thực hiện tốt luận văn này.

Chúng em xin chân thành cảm ơn thầy Lê Ngọc Sơn và anh Nguyễn Duy Vũ đã tận tình hướng dẫn, quan tâm và dìu dắt chúng em trong thời gian thực hiện luận văn.

Chúng em cũng xin gửi lời cảm ơn chân thành đến các Thầy Cô trong Khoa đặc biệt là thầy Trần Trung Dũng, thầy Lê Giang Thanh, thầy Nguyễn Thanh Quân, những người đã hỗ trợ nhiệt tình trong lúc chúng em thực hiện đề tài.

Dẫu đã cố gắng hết sức trong suốt quá trình thực hiện đề tài, song nhóm vẫn không tránh khỏi những sai sót nhỏ. Chính vì vậy, nhóm rất mong nhận được sự góp ý, phê bình thẳng thắn từ quý thầy cô và các bạn.

Nhóm thực hiện

Trần Thành Công và Nguyễn Đức Duy

Khoa Công Nghệ Thông Tin

Bộ môn Mạng Máy Tính và Viễn Thông

ĐỀ CƯƠNG CHI TIẾT

Tên Đề Tài: Tìm hiểu NFV và triển khai hệ thống thử nghiệm.
Giáo viên hướng dẫn: Th.S: Lê Ngọc Sơn Cử nhân: Nguyễn Duy Vũ
Thời gian thực hiện: Từ 1/2017 đến 6/2017
Sinh viên thực hiện: Trần Thành Công – 1312057 Nguyễn Đức Duy - 1312084
Loại đề tài: Tìm hiểu công nghệ - Nghiên cứu lý thuyết và triển khai hệ thống minh họa.
Nội Dung Đề Tài: Tìm hiểu công nghệ NFV (Network Function Virtualization) • Tìm hiểu tổng quan về NFV. • Tìm hiểu kiến trúc của NFV. • Tìm hiểu lợi ích và các ứng dụng thực tiễn của NFV. Xây dựng hệ thống NFV thử nghiệm • Tìm hiểu, so sánh các giải pháp hiện có trên thị trường.

- Xây dựng mô hình thử nghiệm.

Phát triển thành phần bổ trợ

- Phát triển một thành phần bổ trợ cho hệ thống đã triển khai.

Kế Hoạch Thực Hiện:

Tuần 1 – 12: Tìm hiểu lý thuyết

- Tuần 1 – 4:
    - Xác định phương hướng và cách thức nghiên cứu
    - Tìm hiểu tổng quan về NFV
    - Tìm hiểu lợi ích và các ứng dụng thực tiễn của NFV
- Tuần 5 - 8:
    - Tìm hiểu kiến trúc NFV
        - NFV MANO (NFV Manage and Orchestrate)
        - NFVI (Network Functions Virtualisation Infrastructure)
        - VNF (Virtualised Network Function)
- Tuần 9 - 10:
    - Tìm hiểu sự kết hợp giữa NFV và SDN (Software-defined Network)

Tuần 11 – 15: Triển khai cài đặt

- Tìm hiểu so sánh các giải pháp để triển khai mô hình.
- Thử nghiệm các giải pháp và xây dựng mô hình NFV thử nghiệm.

Tuần 16 – 20:

- Tìm hiểu và phát triển một thành phần bổ trợ cho hệ thống đã triển khai.
- Viết báo cáo.

Tuần 21 – 24:

- Hoàn chỉnh hệ thống, quay demo.
- Hoàn thành báo cáo.

Xác nhận của GVHD **ThS. Lê Ngọc Sơn**	**Ngày……tháng……năm……** **Nhóm sinh viên thực hiện** **Trần Thành Công - Nguyễn Đức Duy**

MỤC LỤC

Chương 1 Giới thiệu 1

1.1. Giới thiệu chung 1

1.2. Giới thiệu đề tài 4

1.3. Nội dung luận văn 4

Chương 2 Công nghệ NFV 6

2.1. Định nghĩa 6

2.2. Lợi ích của NFV 8

2.3. Quá trình phát triển 10

2.4. Kiến trúc 11

2.4.1. Tổng quan kiến trúc 11

2.4.2. Khối hàm chức năng mạng ảo hóa - VNF 14

2.4.2.1. Tổng quan 14

2.4.2.2. Kiến trúc 17

2.4.3. Khối hạ tầng ảo hóa - NFVI 18

2.4.3.1. Compute Domain 19

2.4.3.2. Hypervisor Domain 19

2.4.3.3. Infrastructure Network Domain 20

2.4.3.4. Một số công nghệ tối ưu cho hạ tầng NFVI 21

2.4.4. Khối điều phối và quản lý – NFV MANO 26

2.4.4.1. Tổng quan 26

2.4.4.2. Kiến trúc 27

2.4.5. Các kho chứa dữ liệu của khối MANO 29

2.4.5.1. NFVI Resources Repository 29
2.4.5.2. NFV Instances repository 29
2.4.5.3. VNF Catalog 29
2.4.5.4. Network Service Catalog 31
2.5. Mối liên hệ giữa NFV và SDN 33
2.5.1. Sơ lược về SDN 33
2.5.2. Mối liên hệ giữa NFV và SDN 34
2.5.3. Kết hợp SDN vào NFV 36
2.6. Các ứng dụng thực tiễn tiêu biểu của NFV. 41
2.6.1. Theo ETSI 41
2.6.1.1. Dịch vụ hạ tầng ảo hóa chức năng mạng 41
2.6.1.2. Ảo hóa hạ tầng viễn thông và di động 43
2.6.1.3. Kết nối các dịch vụ mạng (Service Chaining) 45
2.6.2. Tiềm năng ứng dụng NFV tại Việt Nam 47
2.6.2.1. Các nhà cung cấp dịch vụ Internet và viễn thông 47
2.6.2.2. Các công ty, tổ chức, đơn vị sự nghiệp 47
Chương 3 Các giải pháp triển khai 48
3.1. Khối hạ tầng ảo hóa – NFVI 48
3.1.1. OpenStack 48
3.1.2. OPNFV 49
3.2. Khối điều phối và quản lý – NFV MANO 51
3.2.1. OpenBaton 51
3.2.1.1. Tổng quan 51
3.2.1.2. Kiến trúc 52

3.2.1.3. Nhận xét 53
3.2.2. OpenSource MANO 53
3.2.2.1. Tổng quan 53
3.2.2.2. Kiến trúc 54
3.2.2.3. Nhận xét 55
3.2.3. Tacker 56
3.2.3.1. Giới thiệu 56
3.2.3.2. Kiến trúc 57
3.2.3.3. Nhận xét 59
3.2.4. Tổng kết các giải pháp MANO 60
Chương 4 Triển khai thử nghiệm 62
4.1. Triển khai thử nghiệm hệ thống NFV 62
4.1.1. Thử nghiệm giải pháp OpenBaton 62
4.1.1.1. Mô hình 62
4.1.1.2. Kết quả 63
4.1.1.3. Nhận xét OpenBaton 64
4.1.2. Thử nghiệm giải pháp Tacker 64
4.1.2.1. Mô hình 64
4.1.2.2. Kết quả 66
4.1.2.3. Nhận xét 67
4.1.3. So sánh và đánh giá hai giải pháp thử nghiệm 67
4.2. Triển khai thử nghiệm VNF 67
4.2.1 Nhu cầu 67
4.2.2 Đề xuất 69

4.2.3 Kết quả thử nghiệm.....73

Chương 5 Phát triển công cụ hỗ trợ giám sát hệ thống NFV 78

5.1. Thực trạng.....78

5.2. Kiến trúc 80

5.3. Kết quả.....82

Chương 6 Tổng kết và hướng phát triển 84

6.1. Kết quả đạt được.....84

6.2. Hướng phát triển của đề tài 84

Danh mục tài liệu tham khảo 85

PHỤ LỤC 86

Alarm monitoring framework của Tacker 86

Viết một monitor driver mới cho Tacker 87

DANH MỤC VIẾT TẮT

API	Application Programming Interface
CDN	Content Delivey Network
COTS	Commercial Off The Shelve
CPE	Customer Premises Equiment
DNS	Domain Name System
DPDK	Data Plane Development Kit
EMS	Element Management System – Hệ thống quản lý thực thể
ETSI	European Telecommunications Standards Institute
IMS	IP Multimedia Sybsystem
IoT	Internet of Things
KVM	Kernel-based Virtual Machine
MANO	Management and Orchestration
NAT	Network Address Translation
NIC	Network Interface Card
NF	Network Function – Hàm chức mạng
NFFG	Network Function Fowarding Graph
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NIC	Network Interface Card
NS	Network Service

NUMA	Non-Uniform Memory Access
OPNFV	Open Platform for NFV
OSS/BSS	Operation/Business Support System
OVS	Open vSwitch
PCI	Peripheral Component Interconnect
PF	Physical Function
PNF	Physical Network Function
PoP	Point of Presence
QEMU	Quick Emulator
SDN	Software Defined Networking
SFC	Service Function Chainning - Liên kết các dịch vụ mạng
VF	Virtual Function
VIM	Virtualised Infrastructure Manager – Trình quản lý hạ tầng ảo hóa
VNF	Virtual Network Function - Hàm chức năng mạng ảo
VNFD	VNF Descriptor
VNFFG	VNF Forwarding Graph
VNFM	VNF Manager
VR	Virtual Reality
VXLAN	Virtual Extensible Local Area Network

DANH MỤC HÌNH ẢNH

Hình 1-1 Các vấn đề của hạ tầng mạng hiện tại.....2
Hình 1-2 Nhu cầu của các chủ thể. 3
Hình 2-1 So sánh giữa thiết bị mạng truyền thống và NFV (Theo ETSI).....6
Hình 2-2 So sánh chi phí tiết giảm khi ứng dụng NFV tại doanh nghiệp (Nguồn ADVA).....10
Hình 2-3 Kiến trúc tham chiếu của NFV (Theo ETSI).....12
Hình 2-4 Mối liên hệ giữa NS, VNF và VM.....15
Hình 2-5 Kiến trúc của một VNF.....17
Hình 2-6 Vị trí các miền con của NFVI.....18
Hình 2-7 Các thành phần của Hypervisor Domain (Nguồn: Verizon) 19
Hình 2-8 Tốc độ xử lý gói tin ở L3 tăng lên nhiều lần khi ứng dụng DPDK.....22
Hình 2-9 So sánh giữa PCI passthrough và ảo hóa card PCI.....23
Hình 2-10 Phân tách card mạng vật lý thành các card mạng ảo (Nguồn: Juniper)...25
Hình 2-11 Sơ đồ khối MANO và các interface – Nguồn: ETSI 27
Hình 2-12 Tập tin VNFD sau được viết theo cấu trúc TOSCA.....30
Hình 2-13 Mối liên hệ giữa NSD và VNFD 31
Hình 2-14 Ví dụ một Network Service Descriptor theo chuẩn TOSCA.....31
Hình 2-15 Cấu trúc một tập tin NSD có đuôi zip theo chuẩn CSAR.....32
Hình 2-16 Kiến trúc tổng quát SDN 34
Hình 2-17 Mô hình kiến trúc tổng quan 1 hạ tầng NFV + SDN do Verizon đề ra...36
Hình 2-18: Workflow hoàn chỉnh của giải pháp NFV kết hợp SDN.....38
Hình 2-21 Danh sách các ứng dụng phổ biến nhất của NFV năm 2016 (Nguồn: ADVA).....41
Hình 2-19 Một mô hình NFVIaaS tiêu chuẩn.....42
Hình 2-20 Ảo hóa mạng tại căn hộ của người sử dụng.....44
Hình 2-21 Distributed vCPE và Centralized vCPE 44
Hình 2-22 Một Network Service gồm 3 Network Function 46

Hình 2-23 Một VNF Forwarding Graph phức tạp gồm nhiều luồng khác nhau.....46
Hình 3-1 Những mục tiêu chính của dự án OPNFV 50
Hình 3-2 Kiến trúc của OpenBaton.....52
Hình 3-3 Logo của OpenSource MANO 54
Hình 3-4 Kiến trúc của OSM 54
Hình 3-5 Kiến trúc của Tacker.....57
Hình 3-6 Quy trình làm việc của Tacker.....58
Hình 3-7 Quy trình khởi tạo VNFFG bằng Tacker và SDN Controller 59
Hình 4-1 Mô hình thử nghiệm OpenBaton 63
Hình 4-2 Network service demo trong OpenBaton 63
Hình 4-3 Kịch bản thử nghiệm Tacker 65
Hình 4-4 Mô hình thử nghiệm Tacker 66
Hình 4-5 Một VNF demo trong Tacker 66
Hình 4-6 Kiến trúc của IMS ClearWater 69
Hình 4-7 Đồ hình mạng Demo vIMS.....70
Hình 4-8 Luồng dữ liệu của demo VNFFG 72
Hình 4-9 Tạo VNF vIMS ClearWater từ dashboard.....73
Hình 4-10 Người dùng đăng kí tài khoản tại tổng đài vIMS 73
Hình 4-11 Người dùng 1 thực hiện cuộc gọi tới người dùng 2.....74
Hình 4-12 Hai người dùng thực hiện được cuộc gọi video với nhau 74
Hình 4-13 Thông tin chi tiết VNF1.....75
Hình 4-14 Thông tin chi tiết VNF2.....75
Hình 4-15 VNF OpenWRT đã được cho phép forward gói tin. 76
Hình 4-16 Traceroute từ máy Source tới máy Dest trước khi tạo VNFFG 76
Hình 4-17 Tạo VNFFG tên là VNFFG1 76
Hình 4-18 Chi tiết tập tin VNFFGD của VNFFG1.....77
Hình 4-19 Traceroute từ máy Source tới máy Dest sau khi tạo VNFFG.....77
Hình 5-1 Giao diện quản lý VIM của Tacker 78
Hình 5-2 Giao diện quản lý chi tiết một VIM Instance của Tacker.....78

Hình 5-3 Giao diện quản lý các VNF của Tacker.....79
Hình 5-4 Giao diện quản lý chi tiết một VNF.....79
Hình 5-5 Quy trình theo dõi các VNF.....80
Hình 5-6 Quy trình theo dõi VIM tại các site khác nhau.....81
Hình 5-7 Công cụ theo dõi các VNF.....82
Hình 5-8 Công cụ theo dõi các VIM.....83

DANH MỤC BẢNG

Bảng 2-1 Bảng so sánh giữa NFV và hạ tầng mạng truyền thống.....8
Bảng 2-2 So sánh giữa SDN và NFV.....35
Bảng 2-3 Bảng chú thích cho hình 2–18.....39
Bảng 2-4 Bảng phụ lục cho hình 2-18 40
Bảng 3-1 Cấu hình tối thiểu để triển khai một hệ thống OPNFV.....51
Bảng 3-2 Cấu hình tối thiểu để triển khai một node OSM 56
Bảng 3-3 Bảng so sánh các giải pháp MANO 61

Chương 1 Giới thiệu

Trong chương này chúng tôi sẽ trình bày về thực trạng hạ tầng mạng của nhà cung cấp dịch vụ viễn thông hiện nay nhằm nói lên tính cấp thiết của đề tài. Phần cuối sẽ trình bày về mục tiêu và ý nghĩa của khóa luận.

1.1. Giới thiệu chung

Trong thời đại hiện nay, hàng loạt các công nghệ mới đang lần lượt ra đời và phát triển như mạng di động 5G, Internet của vạn vật (Internet of Things), điện toán đám mây (Cloud Computing), thực tại ảo (Virtual Reality)… Bên cạnh đó, việc mở rộng hoạt động kinh doanh của các doanh nghiệp Internet và viễn thông dẫn đến nhu cầu về băng thông, chất lượng đường truyền và quản lý luồng dữ liệu tăng lên theo cấp số nhân. Sau đây là một số dự đoán về hiện trạng mạng toàn cầu trong giai đoạn 2015 - 2020:

- *Mức độ sử dụng Internet sẽ tăng từ mức 10GB/người/tháng vào năm 2015 lên đến 25GB/người/tháng vào năm 2020. Kéo theo đó, lưu lượng Internet toàn cầu có thể sẽ bước qua mốc zettabyte (1 zettabyte = 1 tỷ terabyte) và có thể sẽ đạt đến mức 2.3 zetabyte vào năm 2020.*
- *Lưu lượng dữ liệu của các thiết bị không dây và điện thoại di động sẽ chiếm đến gần ⅔ tổng lưu lượng Internet toàn cầu vào năm 2020.*
- *Tốc độ Internet băng thông rộng sẽ tăng gần như gấp đôi vào năm 2020 (từ mức 24.7 Mbps vào năm 2015 và sẽ đạt mức 47.7Mbps năm 2020).*

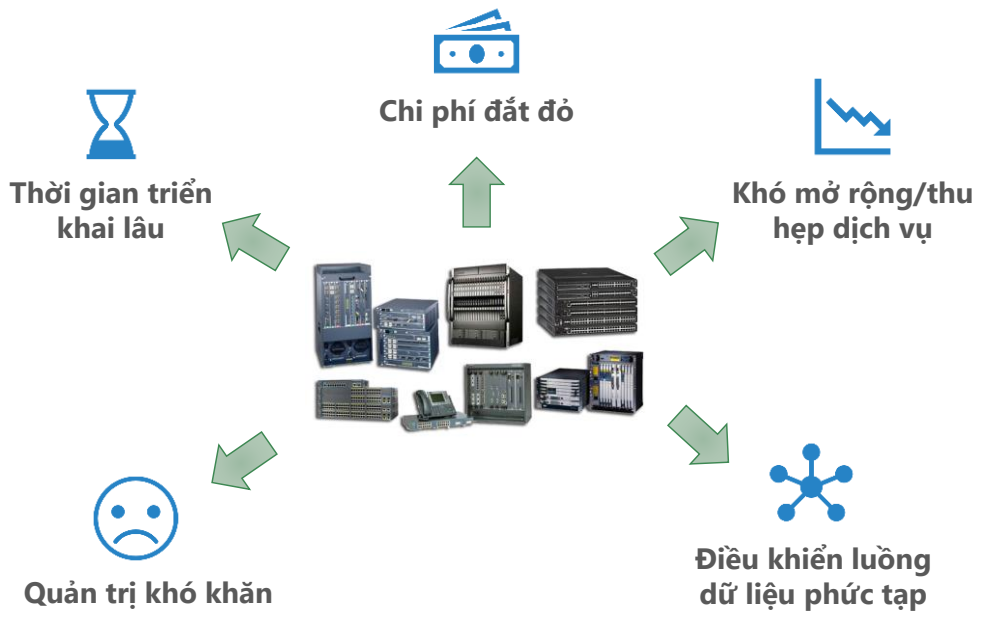
(Nguồn: Cisco Visual Networking Index 2015-2020)

Tại Việt Nam, vào nửa đầu năm 2017, Viettel cũng đã bắt đầu triển khai rộng rãi hệ thống mạng 4G tại Việt Nam. Việc này đặt ra một bài toán phức tạp về việc xây dựng lại hạ tầng phần cứng mạng bên dưới để đáp ứng các nhu cầu mới.

Những số liệu trên cho thấy xu hướng phát triển vũ bão của Internet nhằm đón đầu thời đại Cách Mạng Công Nghiệp 4.0 đã và đang diễn ra. Đồng thời, nhu cầu phải

luôn không ngừng cải thiện hạ tầng mạng là một nhu cầu thiết yếu không chỉ ở thế giới mà còn ở Việt Nam.

Điều này đặt ra cho các nhà cung cấp dịch vụ mạng áp lực phải luôn không ngừng mở rộng qui mô cũng như nâng cao chất lượng dịch vụ truyền dẫn. Thế nhưng, đây lại không phải là một vấn đề đơn giản. Cách làm phổ biến hiện tại của các nhà cung cấp dịch vụ viễn thông hiện tại đa phần là mua sắm thêm các thiết bị phần cứng chuyên dụng cho mỗi một dịch vụ mạng mới. Cách tiếp cận này hiện đang bộc lộ nhiều bất cập.



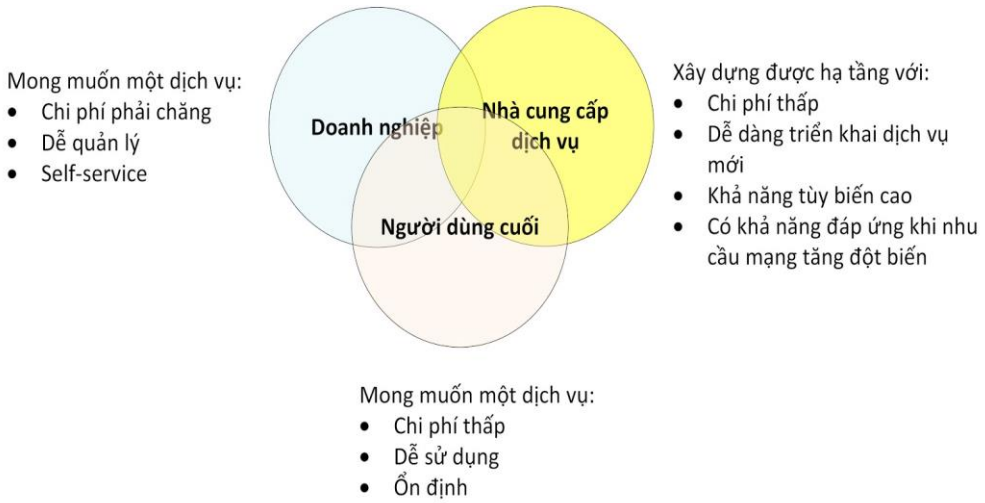
Hình 1-1 Các vấn đề của hạ tầng mạng hiện tại.

Đa phần những hệ thống mạng hiện tại đều sử dụng thiết bị chuyên dụng của các hãng như Cisco hay Juniper,... Tuy nhiên, những hệ thống này lại có các khuyết điểm như: giá thành thiết bị đắt đỏ, khó quản lý tập trung, kém tương thích với các hệ thống của hãng khác, tốc độ cập nhật phần mềm chậm, giấy phép sử dụng phần mềm thường ngắn...

Thông thường, quy trình khởi tạo một dịch vụ mạng hiện nay gồm những bước sau: xác định nhu cầu, thiết kế, lắp đặt thiết bị mạng, đấu nối dây, cấu hình dịch vụ, kiểm

thử và cuối cùng mới là đưa vào vận hành. Với mỗi một quy trình như vậy có thể phải cần tới vài ngày, vài tuần hoặc lâu hơn để đưa hệ thống mới vào hoạt động. Trong khi đó, mỗi một dự án lại có các yêu cầu riêng, đòi hỏi những loại thiết bị chuyên dụng khác nhau. Đặc biệt là có những dự án có thời gian sử dụng ngắn từ vài tháng đến chỉ vài ngày hoặc thậm chí là vài giờ thì việc triển khai dịch vụ theo mô hình truyền thống là vô cùng lãng phí thời gian.

Đây là những khuyết điểm không thể chấp nhận trong môi trường công nghệ thông tin hiện nay bởi nhu cầu của từng khách hàng hiện tại là rất đa dạng và đặc thù. Mỗi một giây chậm trễ đều lãng phí tiền bạc và nguồn lực của công ty mà quan trọng hơn là đánh mất sự tín nhiệm của người dùng. Với những vấn đề tồn đọng trên thì hạ tầng mạng hiện có được dự báo sẽ không thể đáp ứng kịp nhu cầu của thị trường cũng như đảm bảo lợi ích của các chủ thể bao gồm doanh nghiệp, nhà cung cấp dịch vụ và người dùng cuối.



Hình 1-2 Nhu cầu của các chủ thể.

Giải pháp ở đây chính là ứng dụng công nghệ ảo hóa vào hạ tầng mạng tại các trung tâm dữ liệu các điểm chuyển mạch lớn trên đường truyền hoặc tại nhà của người dùng bằng công nghệ Ảo hóa Chức năng Mạng (Network Function Virtualization - NFV).

Công nghệ NFV cho phép ta tách biệt các hàm chức năng mạng (Network Function - NF) như: NAT, Firewall, Intrusion Detection, DNS, Caching,... khỏi các thiết bị vật lý chuyên biệt và triển khai các NF này dưới hình thức phần mềm có thể chạy trong

môi trường ảo hóa trên các thiết bị phần cứng phổ thông. Các thiết bị vật lý lúc này không còn là các phần cứng độc quyền của các hãng nữa, mà có thể là các máy chủ, thiết bị chuyển mạch và thiết bị lưu trữ dữ liệu *phổ thông*.

Việc này sẽ giúp ta giảm chi phí đầu tư và sự phụ thuộc vào các thiết bị phần cứng chuyên biệt của từng hãng như trước đây. Đồng thời, các nhà mạng có thể khởi tạo, điều phối và di chuyển các hàm chức năng mạng, các dịch vụ mạng một cách linh hoạt, từ đó tận dụng tốt hơn hạ tầng phần cứng đã đầu tư. Không chỉ chi phí đầu tư mà cả chi phí vận hành, bảo dưỡng và nâng cấp thiết bị sau này cũng sẽ được cắt giảm đáng kể.

1.2. Giới thiệu đề tài

Nhận thấy được những ưu điểm không thể chối cãi của NFV và ứng dụng của công nghệ này vào hạ tầng mạng viễn thông, nhóm đã nghiên cứu và thử nghiệm thành công luận văn với đề tài: ***"Tìm hiểu NFV và triển khai hệ thống thử nghiệm".***

1.3. Nội dung luận văn

Luận văn này sẽ tập trung tìm hiểu công nghệ NFV cũng như khả năng kết hợp của NFV với công nghệ SDN. Sau đó, nhóm thực hiện sẽ tìm hiểu các mô hình, giải pháp triển khai hệ thống NFV hiện có tính đến thời điểm của luận văn này nhằm triển khai một mô hình thử nghiệm và phát triển một số thành phần cho hệ thống.

Các thuật ngữ, định nghĩa và kiến trúc tổng quát của NFV sẽ chủ yếu dựa theo tiêu chuẩn của Viện Tiêu chuẩn Viễn thông Châu Âu (European Telecommunications Standards Institute – ETSI). Luận văn được chia thành sáu chương theo bố cục sau:

Chương 1: Giới thiệu đề tài

Đây là chương giới thiệu tổng quan về đề tài và nội dung của đề tài. Chương này bao gồm thực trạng về hạ tầng mạng hiện nay của thế giới và Việt Nam và xu hướng ứng dụng công nghệ ảo hóa vào hệ thống viễn thông. Chương một cũng trình bày mục tiêu và ý nghĩa đề tài.

Chương 2: Công nghệ NFV

Đây là chương đi vào tìm hiểu công nghệ NFV cũng như quá trình phát triển, kiến trúc, mô hình hoạt động, các lợi ích và các ứng dụng thực tiễn của công nghệ này. Bên cạnh đó, nhóm cũng sẽ tìm hiểu sơ lược về mối liên hệ giữa NFV và công nghệ SDN.

Chương 3: Các giải pháp NFV

Trong chương ba, nhóm sẽ tập trung tìm hiểu, so sánh về các giải pháp NFV hiện có trên thị trường cũng như khả năng triển khai thử nghiệm các giải pháp này trên môi trường thực tế.

Chương 4: Triển khai và cài đặt hệ thống

Tiếp theo chương ba, chương bốn sẽ tập trung vào việc triển khai thực tế các giải pháp từ đó chọn lựa ra giải pháp phù hợp nhất nhằm đáp ứng cho các ứng dụng thực tế mà nhóm đề xuất ra.

Chương 5: Phát triển công cụ hỗ trợ giám sát hệ thống NFV

Nhóm sẽ trình bày việc phát triển và mở rộng thêm một số tính năng về theo dõi, quản lý tài nguyên trong hệ thống NFV đã triển khai dựa trên nền tảng mã nguồn mở của các dự án.

Chương 6: Tổng kết và đánh giá

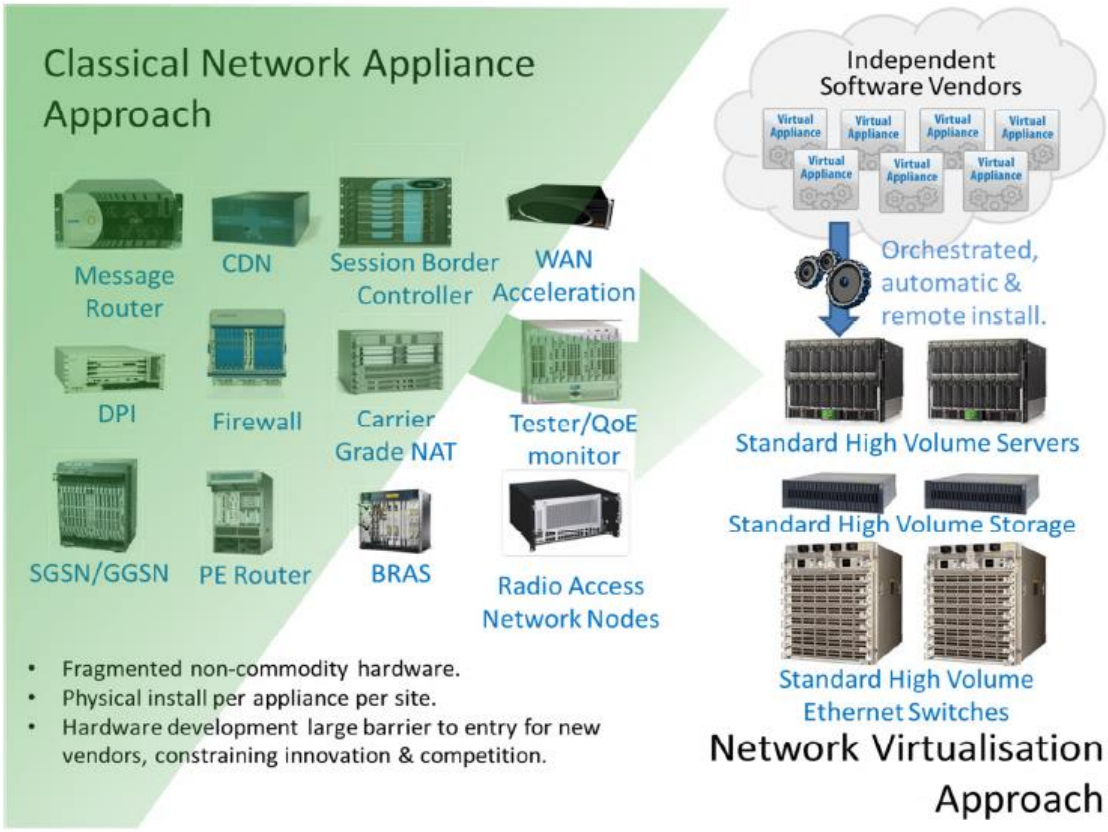
Đây là chương cuối cùng nhằm tổng kết và đánh giá lại những thành quả nghiên cứu đã đạt dược của nhóm cũng như đề xuất thêm một số hướng nghiên cứu mở rộng của công nghệ NFV.

Chương 2 Công nghệ NFV

Trong chương hai, nhóm sẽ đi sâu giới thiệu về công nghệ NFV, kiến trúc, lịch sử phát triển cũng như tìm hiểu các lợi ích và ứng dụng thực tiễn của công nghệ này. Bên cạnh đó, nhóm cũng sẽ trình bày mối liên hệ giữa công nghệ NFV và công nghệ SDN.

2.1. Định nghĩa

Công nghệ Ảo hóa Chức năng mạng chính là công nghệ ứng dụng dụng công nghệ ảo hóa và điện toán đám mây vào các máy chủ và thiết bị mạng phổ thông nhằm tạo ra một môi trường để triển khai các hàm chức năng mạng ảo hóa (Virtualised Network Function - VNF) như: switching, firewall, routing, load balancing,... có chức năng tương tự như trên các thiết bị mạng vật lý truyền thống.



Hình 2-1 So sánh giữa thiết bị mạng truyền thống và NFV (Theo ETSI)

Theo như cách tiếp cận truyền thống, mỗi chức năng mạng sẽ phải do một hoặc một vài thiết bị vật lý chuyên dụng đảm nhận. Do các thiết bị này chỉ đảm trách những nhiệm vụ riêng nên hiệu năng sẽ rất cao nhưng bù lại việc triển khai, vận hành, bảo dưỡng hay mở rộng hệ thống sẽ trở nên phức tạp.

Hướng tiếp cận mới sử dụng NFV sẽ giúp nhà cung cấp dịch vụ mạng linh hoạt hơn trong hoạt động kinh doanh. Tuy vậy, công nghệ này cũng có những khuyết điểm nhất định cần được khắc phục.

Tiêu chí so sánh	Hạ tầng mạng truyền thống	Ứng dụng NFV
Chi phí phần cứng	Chi phí cao hơn: do phải mua cả gói giải pháp của từng hãng sản xuất	Chi phí thấp: do chỉ sử dụng phần cứng phổ thông, đồng thời chủ động được về phần mềm.
Khả năng tùy biến, nâng cấp phần cứng	Khó khăn: do phụ thuộc hoàn toàn vào hãng phần cứng. Khi cần thay thế thì đa phần phải thay thế toàn bộ.	Dễ dàng: do chỉ sử dụng các thiết bị phần cứng phổ thông.
Khả năng tùy biến, quản trị, thay thế, nâng cấp phần mềm	Thấp: do phần mềm trên các thiết bị phần cứng phụ thuộc vào tài nguyên thiết bị và hãng sản xuất.	Cao: do cơ chế nguồn mở và có nhiều hãng cung cấp phần mềm điều khiển.
Khả năng điều khiển luồng traffic	Khó: vì phụ thuộc vào hãng sản xuất và sẽ rất phức tạp nếu sử dụng giải pháp từ nhiều hãng phần cứng khác nhau.	Dễ dàng, linh động: đặc biệt là nếu được kết hợp với công nghệ SDN.
Hệ sinh thái	Nhỏ: bó buộc theo giải pháp của từng hãng.	Rộng: dễ dàng tương tác với các thành phần của các hãng khác thông qua các chuẩn giao tiếp chung được ETSI đặt ra.

Hiệu năng, độ ổn định của dịch vụ.	Cao: do sử dụng các thiết bị được thiết kế chuyên biệt cho từng chức năng mạng đặc thù.	Thấp: hơn do sử dụng các thiết bị phần cứng phổ thông. Tuy nhiên, về lâu dài, hiệu năng sẽ dần được cải thiện.
Khả năng nhận được trợ giúp, hỗ trợ.	Cao: do giấy phép sử dụng của giải pháp đều đi kèm với gói hỗ trợ chính hãng.	Khá thấp: nếu sử dụng các giải pháp nguồn mở. Nếu sử dụng các giải pháp thương mại thì vẫn có nguy cơ nhất định do thị trường NFV còn khá non trẻ.
Đào tạo nhân sự	Nhân sự cần phải tham gia các khóa học của riêng từng hãng phần cứng và lệ thuộc vào hãng đó.	Nhân sự dễ dàng tiếp cận tài liệu, mã nguồn và khóa học về các thành phần của hệ thống do cơ chế mở.

Bảng 2-1 Bảng so sánh giữa NFV và hạ tầng mạng truyền thống

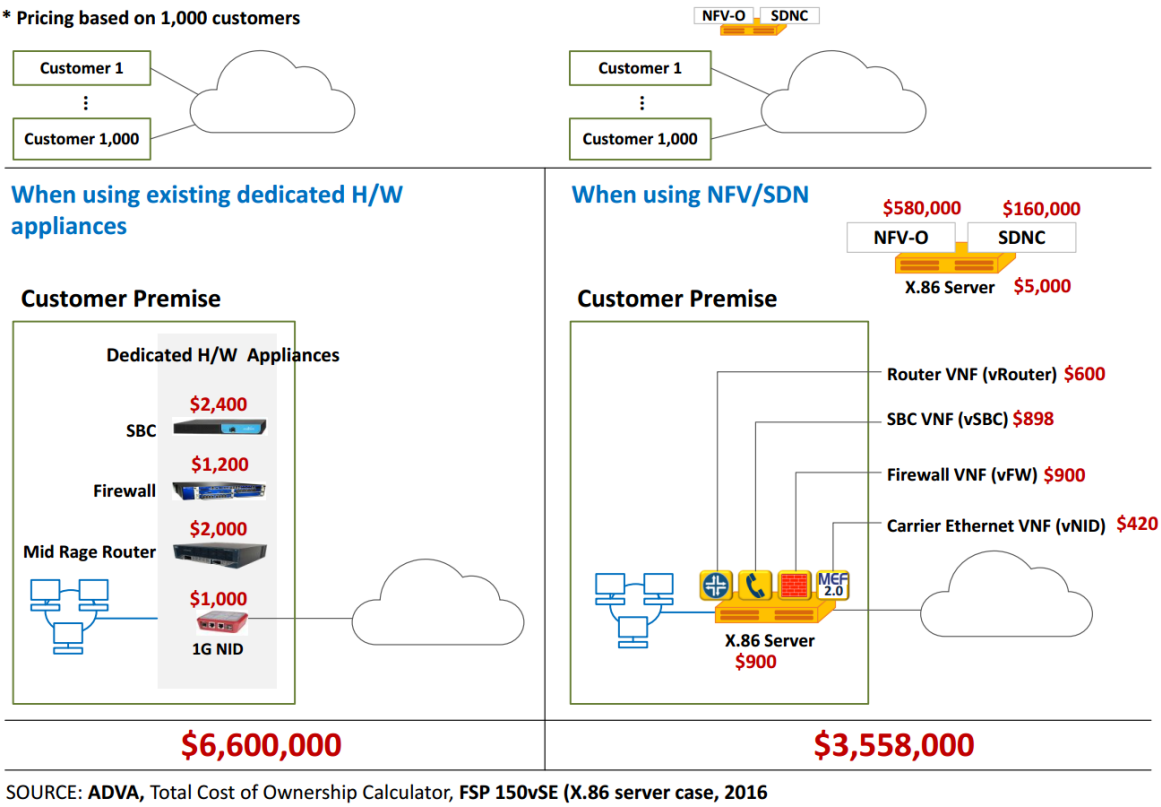
Tuy nhiên, một trong những điểm yếu lớn của công nghệ NFV chính là về hiệu năng của thiết bị chưa thể sánh bằng phần cứng chuyên dụng. Ở mục kiến trúc, nhóm sẽ trình bày một số kĩ thuật để phần nào đó giải quyết vấn đề này.

2.2. Lợi ích của NFV

Sau khi có cái nhìn tổng quan, ta có thể đúc kết được 5 lợi ích chính mà NFV mang lại như sau:

- **Tiết kiệm chi phí đầu tư ban đầu và chi phí vận hành:** Sử dụng hạ tầng ảo hóa trên các thiết bị phần cứng phổ thông sẽ tiết kiệm rất nhiều chi phí. Đồng thời, việc bảo dưỡng, mở rộng, thu hồi hay kết thúc dịch vụ cũng dễ dàng và kinh tế hơn. Bên cạnh đó, nhà cung cấp dịch vụ hoàn toàn có thể chủ động được mã nguồn các phần mềm chức năng mạng của mình, giảm bớt sự phụ thuộc vào chi phí bản quyền trên các thiết bị phần cứng mạng chuyên dụng trước kia.

- **Triển khai các dịch vụ mạng một cách nhanh chóng:** Đây là một trong những ưu điểm quan trọng nhất của NFV. Nhờ tận dụng công nghệ ảo hóa và điện toán đám mây, các kỹ sư có thể cài đặt, thử nghiệm và triển khai các dịch vụ mạng với thời gian được rút ngắn đi rất nhiều so với trước. Điều này đồng nghĩa với việc ta có thể giảm thời gian đưa sản phẩm/dịch vụ mới ra thị trường, tăng tính cạnh tranh cho doanh nghiệp.
- **Tận dụng tối đa hạ tầng phần cứng:** Nhờ tận dụng công nghệ ảo hóa, một hạ tầng mạng vật lý có thể chứa nhiều hạ tầng mạng ảo trên nó. Từ đó, ta có thể phục vụ được nhiều khách hàng khác nhau cũng như tiết kiệm được đáng kể chi phí đầu tư và duy trì hoạt động. Ngoài ra, đội ngũ quản trị còn có thể tạo nên những môi trường thử nghiệm để kiểm thử chức năng phần mềm mới hoặc tập huấn nhân viên song song với các môi trường dịch vụ thật đang hoạt động.
- **Đơn giản hóa việc quản trị:** Thông qua các hàm API, các tập tin cấu hình mẫu được dựng sẵn, các đoạn script tự động,... quản trị viên hệ thống có thể triển khai dịch vụ, kiểm soát luồng dữ liệu, tài nguyên hệ thống,... một cách dễ dàng, linh động, tập trung và tránh được nhiều sai sót từ con người.
- **Linh hoạt trong việc đáp ứng nhu cầu khách hàng:** Hệ thống có thể nhanh chóng khởi tạo thêm hay thay đổi các dịch vụ để đáp ứng nhu cầu đột biến từ khách hàng như khi đột ngột ứng phó với luồng dữ liệu lớn hoặc có sự cố xảy ra. Ngược lại, hệ thống cũng có thể dễ dàng thu hẹp các dịch vụ mạng lại khi lượng tải thấp, giúp tiết kiệm tài nguyên và điện năng tiêu thụ.



SOURCE: **ADVA,** Total Cost of Ownership Calculator, **FSP 150vSE (X.86 server case, 2016**

Hình 2-2 So sánh chi phí tiết giảm khi ứng dụng NFV tại doanh nghiệp (Nguồn ADVA)

2.3. Quá trình phát triển

Định nghĩa về NFV bắt nguồn từ các nhà cung cấp dịch vụ mạng. Họ là những người luôn tìm kiếm giải pháp để thúc đẩy việc triển khai các dịch vụ mạng mới được nhanh hơn, tốt hơn từ đó thu về lợi nhuận cao hơn. Những hạn chế của các thiết bị phần cứng đòi hỏi họ phải áp dụng công nghệ ảo hóa vào hệ thống mạng của họ. Vì chung mục đích như vậy, nhiều nhà cung cấp dịch vụ đã hợp tác với nhau và thành lập nên nhóm ETSI ISG NFV (ETSI Industry Specification Group for Network Functions Virtualization) thuộc ETSI. Đây là nhóm có nhiệm vụ phát triển các yêu cầu và kiến trúc để áp dụng ảo hóa cho các chức năng trong hệ thống mạng viễn thông.

Ban đầu, ETSI ISG NFV gồm có sự tham gia của bảy nhà mạng viễn thông hàng đầu là: AT&T, BT, Deutsche Telekom, Orange, Telecom Italia, Telefonica và Verizon. Ngoài ra còn có sự tham gia của 52 nhà mạng, các nhà cung cấp thiết bị viễn thông,

hãng sản xuất phần cứng khác. Sau đó, cộng đồng ETSI ISG NFV dần dần mở rộng ra với qui mô 230 công ty thành viên, bao gồm nhiều nhà cung cấp dịch vụ trên toàn cầu.

Đến tháng 10-2012, tại hội nghị SDN and OpenFlow World Congress diễn ra tại Darmstadt (Đức), ETSI ISG NFV đã công bố sách trắng đầu tiên về NFV. Từ đó đến nay, nhóm nghiên cứu của ETSI đã tiếp tục cho ra đời những tài liệu đặc tả chuyên sâu hơn về NFV, bao gồm các định nghĩa về thuật ngữ tiêu chuẩn, các ứng dụng, mô hình kiến trúc tổng quát,... Chúng đã trở thành những tài liệu tham khảo chuẩn về một mô hình NFV cho cả các nhà phát triển và nhà cung cấp dịch vụ ứng dụng NFV.

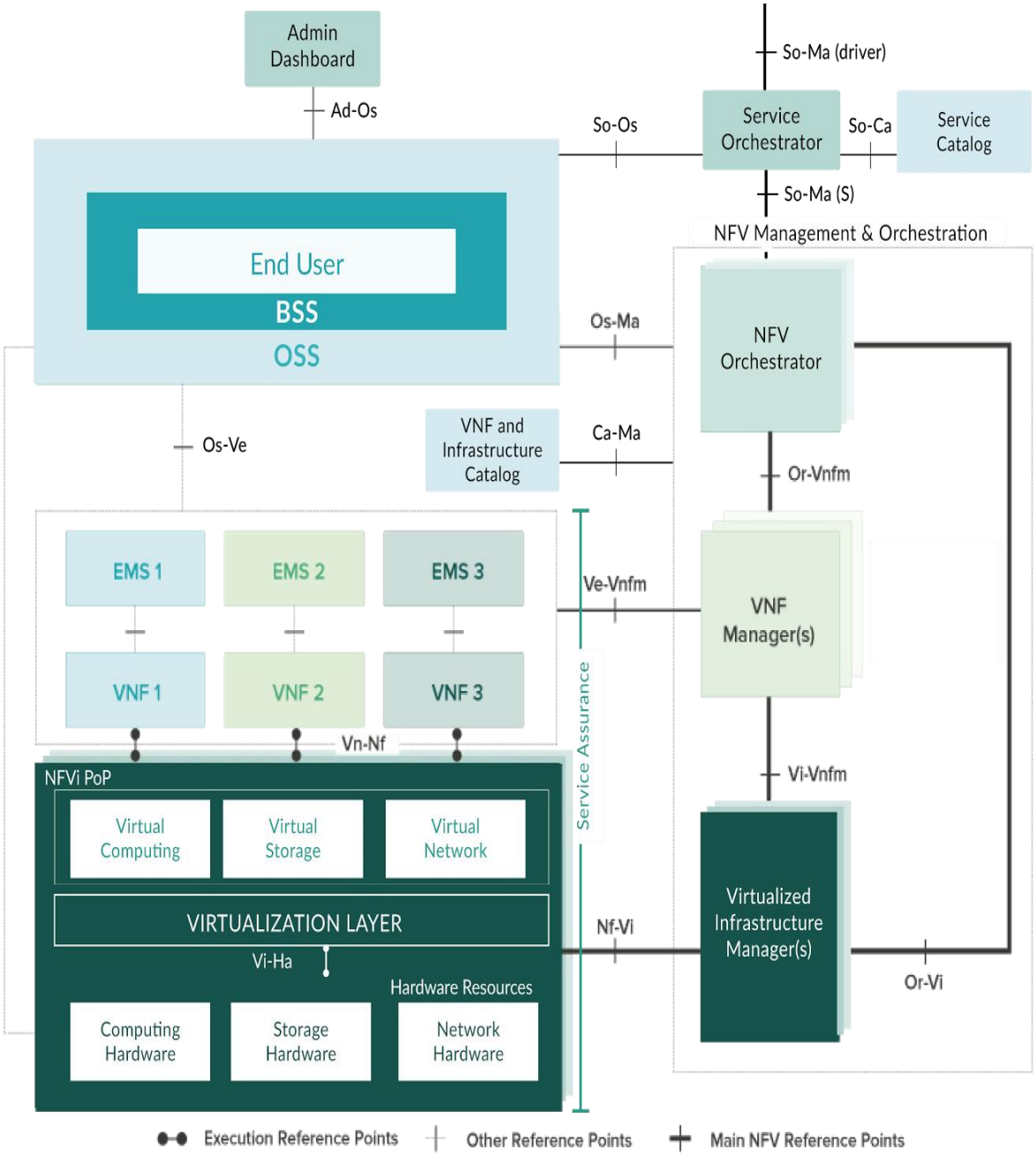
Hiện nay, trên thế giới đã và đang có rất nhiều hãng công nghệ, viện nghiên cứu, trường đại học lớn đầu tư vào nghiên cứu công nghệ NFV:

- Về hướng nghiên cứu các tiêu chuẩn, đặc tả: bên cạnh Viện Tiêu chuẩn Viễn thông Châu Âu, còn có 3GPP, OASIS... đã đưa ra các sách trắng và tài liệu đặc tả nhằm tạo nên bộ khung để các công ty, cộng đồng nguồn mở xây dựng các dự án của riêng mình và có thể liên kết chúng với nhau.
- Về hướng phát triển các sản phẩm, giải pháp NFV
    - Các dự án nguồn mở như: OPNFV, OpenSourceMano, OpenBaton,...
    - Các sản phẩm thương mại như: Red Hat, Mirantis, VMWare NSX,...
- Về hướng ứng dụng trong hoạt động kinh doanh thực tế
    - Dự án triển khai hệ thống 5G của SK Telecom (Hàn Quốc)
    - Nhà mạng AT&T (Mỹ): AT&T đã tuyên bố rằng hãng sẽ ảo hóa 75% hạ tầng mạng của mình vào năm 2020 bằng cách ứng dụng công nghệ NFV và công nghệ SDN (Software-defined Networking).

2.4. Kiến trúc

2.4.1. Tổng quan kiến trúc

ETSI đã đề ra một mô hình kiến trúc tổng quan cho một hệ thống NFV. Mô hình này được số đông cộng đồng phát triển chấp nhận và tuân theo.



Hình 2-3 Kiến trúc tham chiếu của NFV (Theo ETSI)

Theo ETSI, một nền tảng NFV sẽ gồm có ba khối chính là:

- **Các hàm chức năng mạng ảo (Virtualised Network Function - VNF):** là các phần mềm đảm nhiệm các chức năng mạng như switching, routing, load balancing,... đã được ảo hóa. Về cơ bản, VNF chính là các phần mềm chạy trên các máy ảo và không yêu cầu phần cứng chuyên dụng như các thiết bị

mạng truyền thống. VNF chạy trên hạ tầng mạng được ảo hóa (NFVI), được quản lý bởi khối điều phối và quản lý (MANO) cũng như hệ thống quản lý các thực thể (Element Management System - EMS) bên trong các máy ảo chạy VNF đó.

- **Khối hạ tầng ảo hóa chức năng mạng (Network Functions Virtualisation Infrastructure - NFVI):** là tổng thể các thành phần (cả phần cứng lẫn phần mềm) cung cấp tài nguyên cần thiết cho các VNF hoạt động. Tầng này bao gồm các thành phần phần cứng phổ thông và một lớp phần mềm ảo hóa phân tách giữa VNF và tài nguyên phần cứng. NFVI sẽ thông qua lớp ảo hóa để cung cấp tài nguyên cho các VNF bên trên. NFVI được quản lý bởi khối MANO và có thể nằm trên nhiều vị trí địa lý khác nhau tùy theo kịch bản riêng của từng dịch vụ. NFVI bao gồm hai thành phần là:
    - Tài nguyên phần cứng vật lý (Hardware Resource): gồm tài nguyên tính toán, lưu trữ và mạng vật lý.
    - Lớp ảo hóa (Virtualisation Layer): lớp ảo hóa tạo ra các tài nguyên tính toán, lưu trữ và kết nối mạng ảo.
- **Khối điều phối và quản lý (NFV Manage and Orchestrate - NFV MANO):** đảm nhiệm việc điều phối và quản lý vòng đời của các tài nguyên vật lý, quản lý các phần mềm hỗ trợ ảo hóa, quản lý vòng đời của các VNF. NFV MANO có thể tương tác với nhiều hệ thống NFVI khác nhau do các interface giao tiếp đã được ETSI thống nhất. Điều này giúp tăng tính linh hoạt cho giải pháp NFV. Các nhà phát triển hệ thống NFV giờ đây không cần phải tập trung xây dựng một giải pháp NFV đầy đủ bao gồm cả khối NFVI, MANO và các VNF mà chỉ cần tập trung vào một thành phần. Khối MANO gồm có các khối con sau:
    - **NFV Orchestrator (NFVO):** Quản lý các dịch vụ mạng (Network Services), quản lý chức năng của VNF và các cụm VNF, quản lý vòng đời của dịch vụ mạng, tài nguyên toàn hệ thống, chứng thực, cấp quyền sử dụng tài nguyên cho NFVI.

- **VNF Manager (VNFM):** Quản lý vòng đời của các VNF, điều phối, tùy chỉnh cấu hình, cung cấp thông tin liên lạc giữa NFVI và EMS.
- **Virtualized Infrastructure Manager (VIM):** Quản lý và điều phối các tài nguyên về tính toán, lưu trữ và kết nối mạng của NFVI.

Ngoài ra, còn có các thành phần khác như:

- **OSS/BSS - Operation/Bussiness Support System:** là hệ thống quản lý việc vận hành hệ thống, tương tác với người vận hành, khách hàng.
- **Service, VNF & Infrastructure Catalog:** chính là các tập tin đặc tả để khởi tạo các dịch vụ mạng, các VNF hay kết nối với các hạ tầng ảo hóa một cách nhanh chóng.

Sau đây, nhóm sẽ lần lượt trình bày chi tiết ba khối chính đã đề cập ở trên.

2.4.2. Khối hàm chức năng mạng ảo hóa - VNF

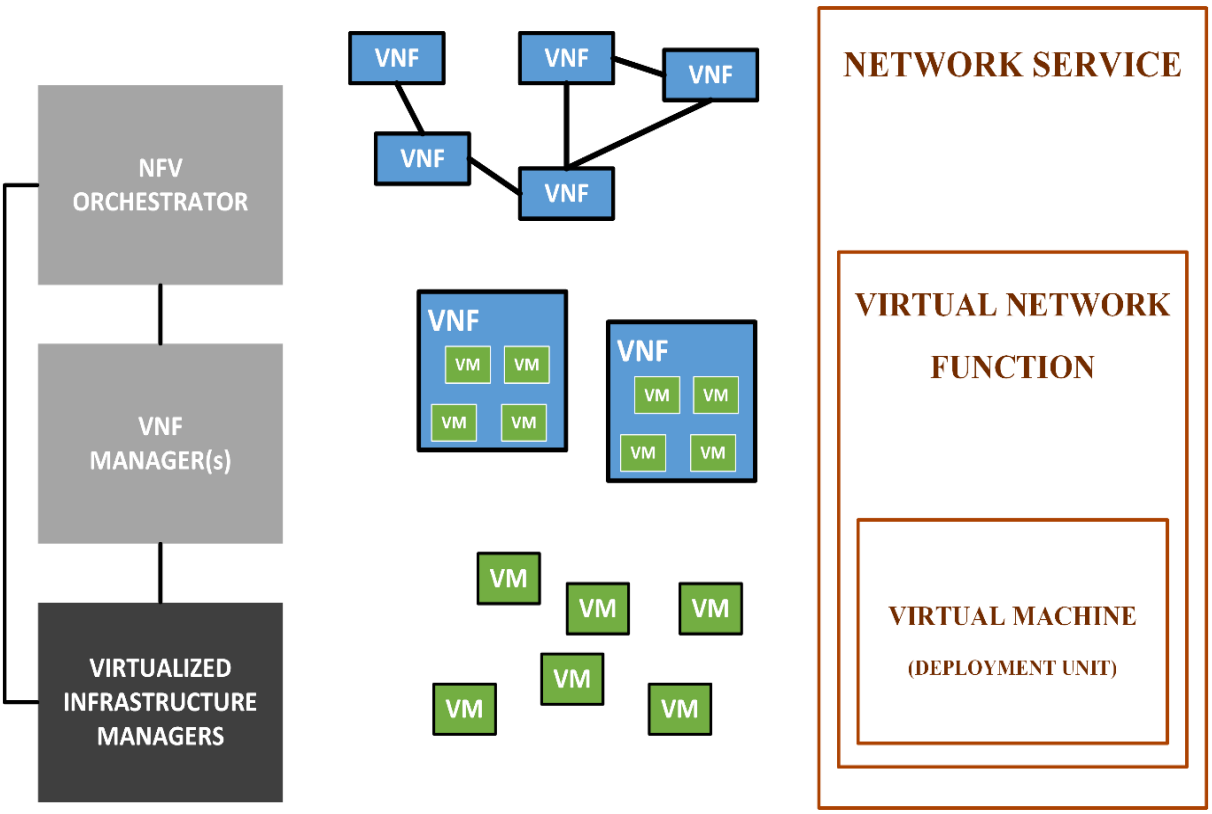
2.4.2.1. Tổng quan

Hàm chức năng mạng ảo hóa (Virtualised Network Function - VNF) là một trong ba thành phần cơ bản trong kiến trúc NFV. Khác với các thiết bị mạng truyền thống (PNF) vốn đòi hỏi phần cứng riêng biệt, một VNF là một hàm đảm trách chức năng mạng được triển khai trên môi trường ảo hóa. Nói một cách đơn giản thì VNF là một máy ảo được cài đặt các chức năng định tuyến, chuyển mạch, tường lửa… Điều này giúp việc triển khai, quản lý và điều phối các VNF trở nên linh hoạt và dễ dàng hơn.

Giống như các PNF, các VNF sẽ đảm trách một chức năng mạng cụ thể nào đó như: routing, switching, firewall... Nhưng dù là ảo hóa, các VNF vẫn phải tuân thủ các chuẩn thiết kết chung của các tổ chức như 3GPP hay IETF,… nhằm tạo thuận lợi cho các nhà phát triển tích hợp giữa các giải pháp với nhau.

Trong kiến trúc của hệ thống NFV, các VNF chạy trong các VM được tạo ra trên hạ tầng NFVI và được điều khiển bởi khối quản lí và điều phối MANO. Bên trong mỗi VNF là các hệ thống quản lý thực thể (Element Management System - EMS). EMS này chính là phần mềm điều khiển của VNF, có nhiệm vụ thu thập các thông tin của

thiết bị ảo và truyền về cho khối MANO xử lý. Ngoài ra, EMS cũng như nhận lệnh từ MANO để thực hiện các tác vụ quản lý trên VNF.



Hình 2-4 Mối liên hệ giữa NS, VNF và VM

Một ví dụ cơ bản về VNF:

- **Customer Premies Equipement (CPE):** là một thiết bị cho phép người dùng kết nối tới mạng Internet của nhà cung cấp dịch vụ. Phổ biến trên thị trường là các loại router/modem của nhà mạng như FPT, VNPT, Viettel. Theo như cách tiếp cận truyền thống, đây sẽ là một thiết bị phần cứng được cài đặt sẵn các chức năng như routing, switching, PPoE, QoS… và thường thì số lượng tính năng này sẽ bị giới hạn bởi thiết bị vật lý.
- Tuy nhiên, với cách tiếp cận của công nghệ NFV thì thiết bị này sẽ được ảo hóa thành 1 thiết bị ảo (vCPE). Một vCPE có thể chứa nhiều VNF trong đó, mỗi VNF sẽ đảm nhiệm một nhiệm vụ riêng như DHCP, DNS… Các VNF nhỏ này sẽ xâu chuỗi lại thành 1 dịch vụ hoàn chỉnh và nằm trong 1 VNF lớn

là vCPE. Nhờ cách tiếp cận theo dạng mô-đun này, việc nâng cấp phần mềm, thêm/bớt dịch vụ, quản lý các CPE trở nên đơn giản hơn rất nhiều.

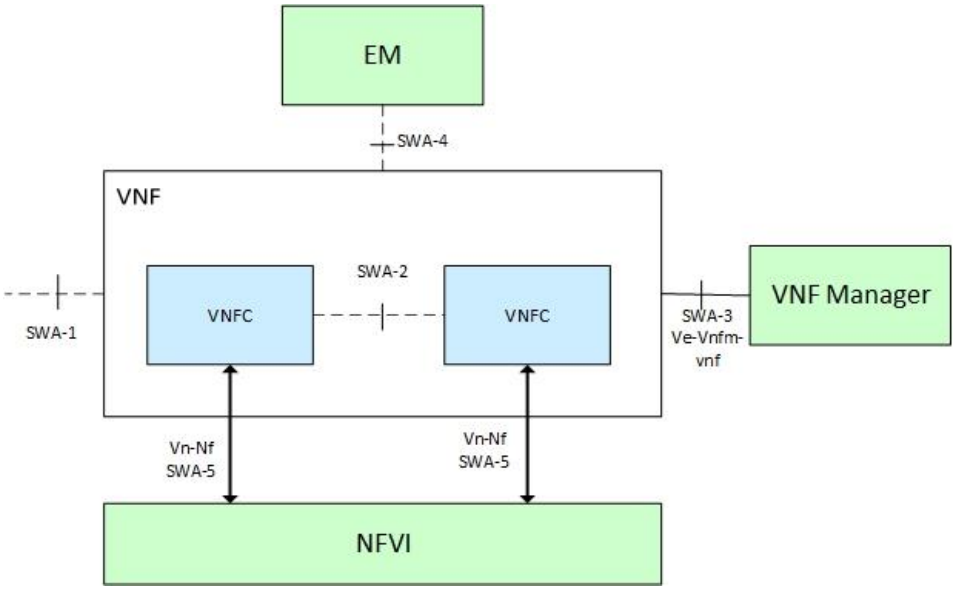
Mỗi một VNF sẽ có những thông tin cấu hình cũng như cách thức hoạt động, chức năng cụ thể. Các thông tin này của từng VNF sẽ được mô tả trong các tập tin gọi là Virtualized Network Function Descriptor (VNFD). VNFD bao gồm các mô tả về cấu hình của một VNF như: số vCPU, memory, số port, thông tin về các kết nối giữa các thành phần trong nội bộ VNF với nhau,...

Khi khởi tạo các VNF, khối MANO sẽ dựa trên những tập tin này để yêu cầu NFVI cung cấp tài nguyên cho hợp lí. Tuy nhiên, việc lựa chọn tài nguyên này đôi khi còn phụ thuộc vào nhiều quy định, yêu cầu khác chứ không nhất thiết phải hoàn toàn theo VNFD (Ví dụ như khả năng đáp ứng của hạ tầng lúc đó, các chính sách bảo mật ở mức người dùng,...).

Nhìn một cách tổng quan, quá trình phát triển một VNF sẽ bao gồm các bước sau:

1. Định nghĩa tập tin VNFD theo yêu cầu người dùng/khách hàng.
2. Định nghĩa các đoạn script quản lý vòng đời của VNF. Các đoạn script này sẽ tạo ra các sự kiện trong vòng đời của một VNF như: khởi tạo máy ảo, cấu hình các thông tin ban đầu, khởi chạy hay hủy dịch vụ,…
3. Phát triển các tập tin image dùng để cài đặt hệ điều hành cho các máy ảo sẽ chạy các VNF.
4. Tải VNFD lên VNF Catalog Repository. Khi được tải lên, hệ thống sẽ kiểm tra thử tính tương thích của VNFD đó với các thông tin tài nguyên khả dụng của hệ thống. Nếu khả thi, hệ thống mới chấp nhận lưu lại VNFD đó.
5. Triển khai thử dịch vụ. Hệ thống sẽ cấp phát tài nguyên và cài đặt VNF.
6. Chạy thử dịch vụ. Nếu cài đặt thành công, hệ thống mới chạy thử nghiệm dịch vụ từ đó có thể đánh giá, nghiệm thu VNF.

2.4.2.2. *Kiến trúc*



Hình 2-5 Kiến trúc của một VNF

Mỗi VNF là một phần mềm được cấu thành bởi một hoặc nhiều thành phần gọi là VNF Components (VNFC). Các VNFC sẽ được cài đặt trên các máy ảo (trên hạ tầng ảo hóa do khối NFVI cung cấp. Việc cấu thành một VNF từ một bộ các VNFC như thế nào phụ thuộc vào nhiều yếu tố như: thứ tự ưu tiên về hiệu năng, khả năng mở rộng, độ tin cậy, bảo mật, các mục tiêu phi chức năng khác,...

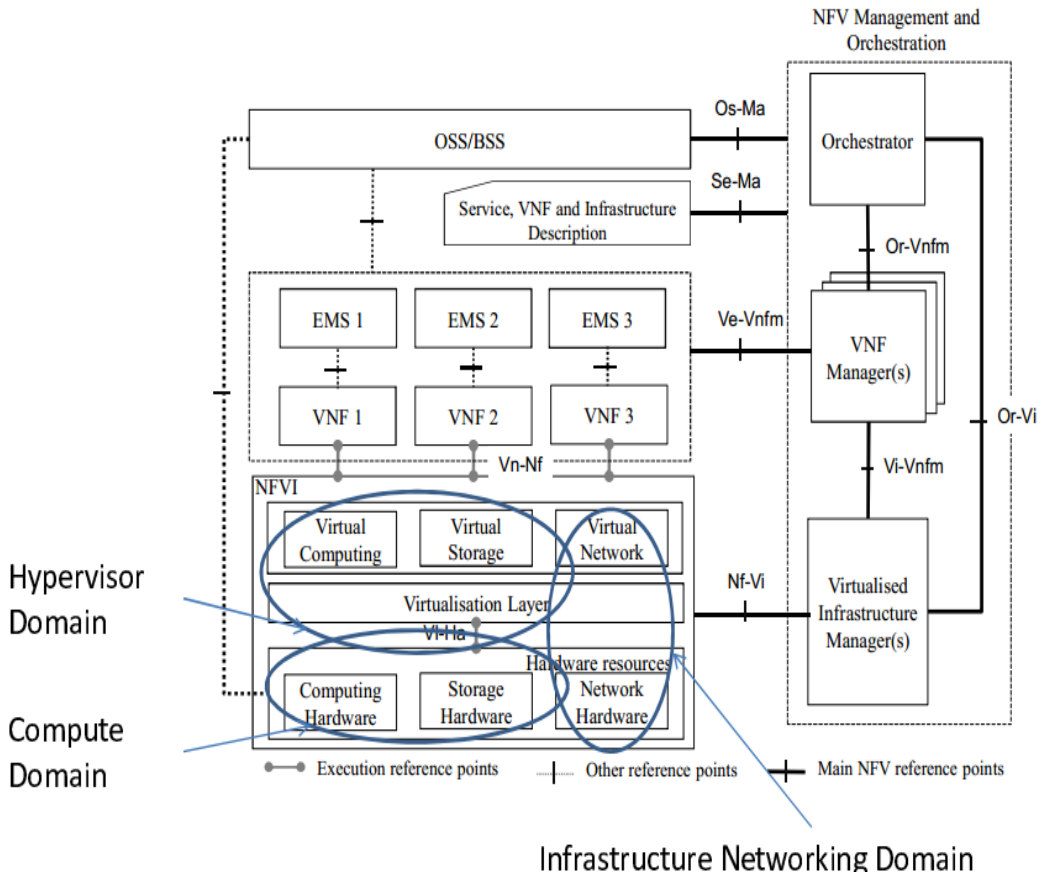
Một cách tổng quan, mỗi VNF có các interface để giao tiếp với các khối khác trong kiến trúc NFV như sau:

1. SWA-1: giao tiếp với các VNF hay PNF khác.
2. SWA-2: giao tiếp giữa các VNFC trong cùng VNF với nhau.
3. SWA-3: giao tiếp với VNF Manager trong khối MANO.
4. SWA-4: giao tiếp với EM (Element Management) riêng của VNF đó.
5. SWA-5: giao tiếp với khối NFVI bên dưới.

Quá trình giao tiếp cụ thể qua các interface này đều được các tổ chức qui chuẩn như ETSI, 3GPP định nghĩa sẵn. Luận văn này xin không trình bày quá sâu vào chi tiết của các tiêu chuẩn này.

2.4.3. *Khối hạ tầng ảo hóa - NFVI*

Khối hạ tầng ảo hóa – Network Function Virtualisation Infrastructure (NFVI) là tập hợp các phần cứng và phần mềm dùng để khởi tạo môi trường cho các VNF hoạt động bên trên. Về phần cứng, NFVI bao gồm các tài nguyên tính toán, lưu trữ, các thiết bị định tuyến, chuyển mạch mạng. Còn phần mềm chính là các phần mềm ảo hóa, các trình điều khiển (driver) tương tác với các thiết bị vật lý, các trình điều khiển thiết bị mạng (OpenFlow, firmware của phần cứng).



Hình 2-6 Vị trí các miền con của NFVI

Do đây là một kiến trúc mở đã được định nghĩa và tiêu chuẩn hóa bởi ETSI nên các doanh nghiệp có thể lựa chọn nhiều công nghệ khác nhau để đảm trách vai trò NFVI mà không phụ thuộc vào bất kỳ một hãng nào cả. Một mô hình triển khai NFV có thể dùng nhiều công nghệ NFVI nhưng vẫn tương tác được với nhau. Một số giải pháp NFVI phổ biến hiện có OpenStack, VMware vSphere, Red Hat, Mirantis…

ETSI tiếp tục chia NFVI thành 3 miền con là:

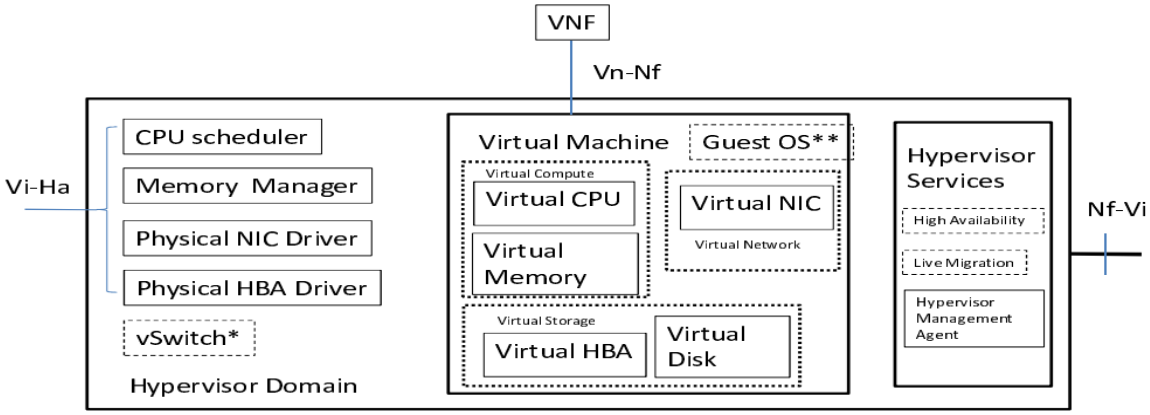
- Miền tính toán (Compute Domain)
- Miền nhân ảo hóa (Hypervisor Domain)
- Miền hạ tầng mạng (Infrastructure Network Domain)

2.4.3.1. Compute Domain

Miền tính toán là các thành phần tài nguyên phần cứng tính toán và lưu trữ vật lý bên dưới. Miền tính toán bao gồm phần cứng tính toán, phần cứng lưu trữ và các interface nhập/xuất (I/O interface) trên các thiết bị, tức là chỉ bao hàm các máy chủ tính toán và hệ thống lưu trữ. Cụ thể là:

- Vi xử lý tính toán & các thành phần tối ưu hiệu năng:
    - Vi xử lý có thể là các dòng chip phổ thông như ARM và Intel x86.
    - Các công nghệ, các thuật toán nén, mã hóa tối ưu hơn để phục vụ bảo mật và tăng cường khả năng xử lý gói tin.
- Lưu trữ:
    - SSD, HDD, tape drive.

2.4.3.2. Hypervisor Domain



Hình 2-7 Các thành phần của Hypervisor Domain (Nguồn: Verizon)

Miền ảo hóa (Hypervisor Domain) có nhiệm vụ cung cấp môi trường thực thi cho các VNF. Miền này sẽ dựa vào miền tính toán để tạo ra môi trường cho các VNF hoạt động. Để thực hiện việc đó, miền ảo hóa sẽ tạo ra một lớp tài nguyên (tính toán, lưu

trữ) ảo hóa, phân tách giữa phần cứng bên dưới và các ứng dụng bên trên. Khi nhận lệnh từ khối điều phối và quản lý, các máy ảo sẽ được tạo ra để chạy các VNF. Đồng thời, thông qua các API/Interface của miền này, người quản trị có thể điều khiển được các máy ảo đã tạo.

Trong một số trường hợp cần tăng tốc độ cũng như đảm bảo băng thông cho VNF thì lớp ảo hóa có thể cho phép các VM kết nối trực tiếp tới phần cứng bên dưới thông qua các kỹ thuật như: SR-IOV, PCI Passthrough,... Các kỹ thuật này sẽ được trình bày kĩ hơn ở mục dưới.

Về mặt quản lý tài nguyên tính toán, đại diện cho miền này trên nền tảng Linux là bộ đôi KVM/Libvirt. Ngoài ra còn có thể kể đến các giải pháp hypervisor (trình ảo hóa) khác như: Xen, VMWare, Hyper-V. Còn về phần quản lý tài nguyên lưu trữ thì hiện có các giải pháp: LVM, Ceph trên Linux...

2.4.3.3. Infrastructure Network Domain

Miền hạ tầng mạng (Infrastructure Network Domain) có nhiệm vụ quản lý các tài nguyên mạng hỗ trợ chuyển mạch và định tuyến của hệ thống như: Top of Rack Switch, router, cáp kết nối giữa các tài nguyên tính toán và lưu trữ khác trong NFVI. Từ đó, miền này cung cấp hạ tầng mạng ảo cho các VNF hoạt động và tương tác với nhau. Cụ thể, miền hạ tầng mạng sẽ:

- Tạo ra các mạng ảo để các VNF liên lạc với nhau.
- Cung cấp không gian địa chỉ và quản lý địa chỉ trong các mạng ảo.
- Cho phép phân tách luồng dữ liệu độc lập giữa các mạng ảo.

Đại diện cho phần này là:

- Các card mạng được nối với vi xử lý qua các cổng PCIe.
- Switch, router phổ thông hỗ trợ các giao thức ảo hóa mạng.
- Các công nghệ tối ưu khả năng xử lý nhập/xuất dữ liệu như SR-IOV, DPDK.
- Các phần mềm ảo hóa mạng như như linux bridge, openvswitch...

2.4.3.4. Một số công nghệ tối ưu cho hạ tầng NFVI

Tuy có nhiều ưu điểm linh hoạt hơn so với hướng tiếp cận sử dụng các thiết bị mạng chuyên dụng, hướng tiếp cận NFV vẫn sẽ có những khuyết điểm nhất định. Một trong số đó là do sử dụng các thiết bị phần cứng phổ thông để tạo môi trường ảo hóa nên sẽ dẫn đến việc giảm hiệu năng hoạt động do chúng vốn không được thiết kế cho những tác vụ mạng đặc thù.

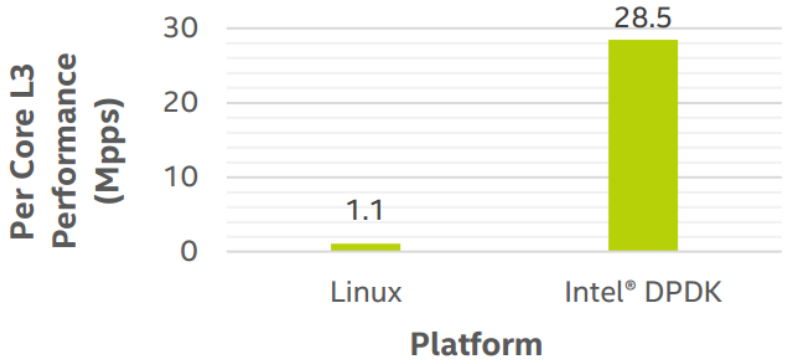
Chính vì thế, bài toán đặt ra là cần có những công nghệ, tập lệnh điều khiển chuyên biệt để giúp các thiết bị phần cứng phổ thông này tăng cường hiệu năng xử lý. Trong phạm vi đề tài này, nhóm xin điểm qua một số công nghệ như vậy.

2.4.3.4.1. DPDK

DPDK (Data Plane Development Kit) là một tập thư viện và trình điều khiển cho các network interface controller. DPDK hỗ trợ việc tăng tốc độ xử lý gói tin trên các dòng CPU đa nhiệm phổ thông như: Intel Atom, Intel Xeon,... Nhờ đó, các dòng CPU phổ thông này có thể đảm trách được việc xử lý các chức năng mạng với hiệu suất tương đương hoặc thậm chí có phần nhanh hơn, linh hoạt hơn so với các dòng NPU, ASIC chuyên dụng truyền thống vốn đã có thể xử lý lưu lượng lên đến hàng trăm gigabit mỗi giây.

Trong các thử nghiệm của mình, Intel đã đưa ra một vài thông số để chứng minh hiệu năng làm việc của DPDK như sau: với chip Xeon E5-2658 v4 thì tốc độ chuyển tiếp các gói tin ở Layer 3 (mỗi gói tin có chiều dài 64 byte) có thể đạt đến 233Gbps.

Cũng theo Intel, về lý thuyết, DPDK có thể tăng tốc độ xử lý gói tin trên CPU nhanh hơn ***25 lần*** so với tốc độ bình thường trên Linux.



Hình 2-8 Tốc độ xử lý gói tin ở L3 tăng lên nhiều lần khi ứng dụng DPDK

Cụ thể, DPDK có thể:

- Gởi và nhận gói tin với ít chu kỳ CPU nhất (thường ít hơn 80 chu kỳ).
- Phát triển các thuật toán bắt gói tin nhanh (tương tự như tcpdump).
- Hỗ trợ chạy các ứng dụng mạng của bên thứ ba.
- Tối ưu việc quản lý bộ đệm.
- Chuyển thao tác nhận gói tin từ push sang poll. Giúp giảm số lần interrupt, context switch và buffer copy nhằm tăng hiệu năng.

Ban đầu, DPDK được Intel phát triển riêng trên các dòng chip Intel x86. Đến thời điểm hiện tại, DPDK đã hỗ trợ nhiều dòng chip khác như: IBM Power 8, EZchip TILE-Gx, ARM... Thế nên, DPDK đã được nhiều đơn vị ứng dụng vào sản phẩm của mình như:

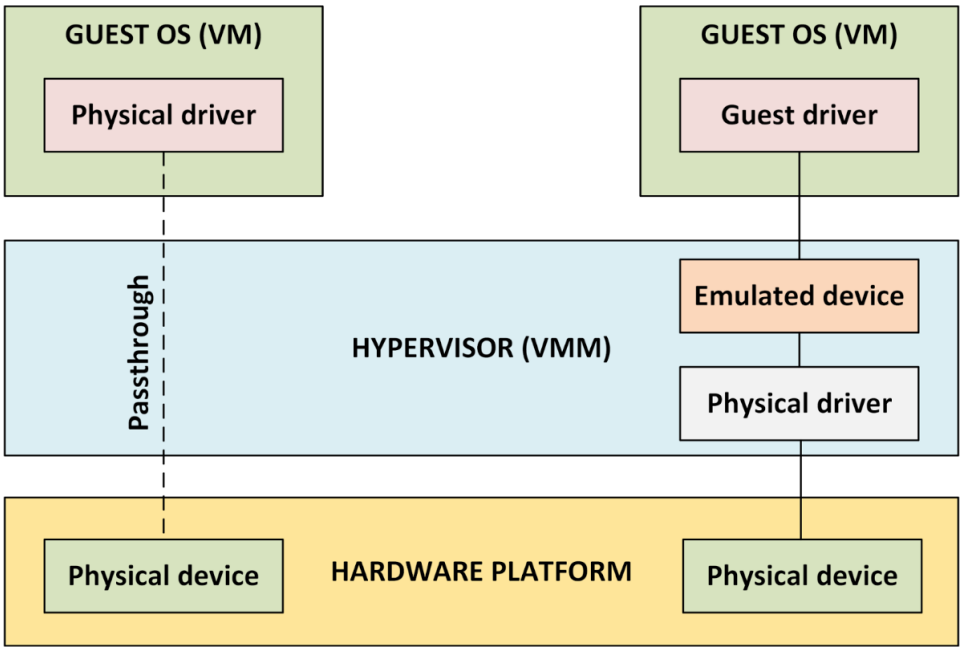
- Các loại card mạng:
    - R2.1: Intel, Cisco, Mellanox, Broadcom, Chelsio,...
    - R2.2: +NetFPGA,...
- Các dòng hệ điều hành:
    - Ubuntu
    - Redhat, Fedora
- Các phần mềm ảo hóa:
    - KVM

- VMware
- XEN

2.4.3.4.2. PCI-Passthrough

PCI-Passthrough là công nghệ cho phép gắn các thiết bị sử dụng cổng PCI (Ethernet, VGA hay hard-drive controllers,..) trên máy host (tài nguyên vật lý thật sự) trực tiếp vào từng máy guest (máy ảo).

Lúc này, máy ảo sẽ xem thiết bị PCI này đã được gắn trực tiếp vào mình và được toàn quyền sử dụng trên thiết bị PCI đó. Khi đã cấu hình tính năng PCI-Passthrough thì ngay cả máy host cũng không còn có thể thấy được thiết bị PCI này cho đến khi thiết bị đó được gỡ khỏi máy guest.



Hình 2-9 So sánh giữa PCI passthrough và ảo hóa card PCI

Ích lợi:

- Tăng đáng kể hiệu năng sử dụng thiết bị PCI cho VM do không còn phải thông qua các bước trung gian nữa.
- Cô lập thiết bị PCI cho riêng VM đó, tránh một số nguy cơ bảo mật, ngẽn cổ chai khi nhiều VM dùng chung một thiết bị vật lý.

- Giúp VM có thể sử dụng một số thiết bị PCI khó hoặc thậm chí là không thể ảo hóa được.

Tuy vậy, PCI-Passthrough vẫn còn có hạn chế:

- Các máy ảo sử dụng chức năng PCI passthrough thì sẽ không còn khả năng lưu, phục hồi hay di cư nữa vì đã bị gán chặt vào một hay một vài thiết bị PCI nhất định. Để thực hiện quá trình di dời các máy ảo, người quản trị cần phải gỡ thiết bị PCI đó ra khỏi hypervisor cũ rồi gắn vào máy ảo trên hypervisor mới.
- Để gán một PCI device vào một VM thì cần phải tách thiết bị đó ra khỏi host vật lý trước.

Một số trình ảo hóa phổ biến hiện nay như KVM, VMware và Xen đều hỗ trợ PCI Passthrough.

2.4.3.4.3. SR-IOV

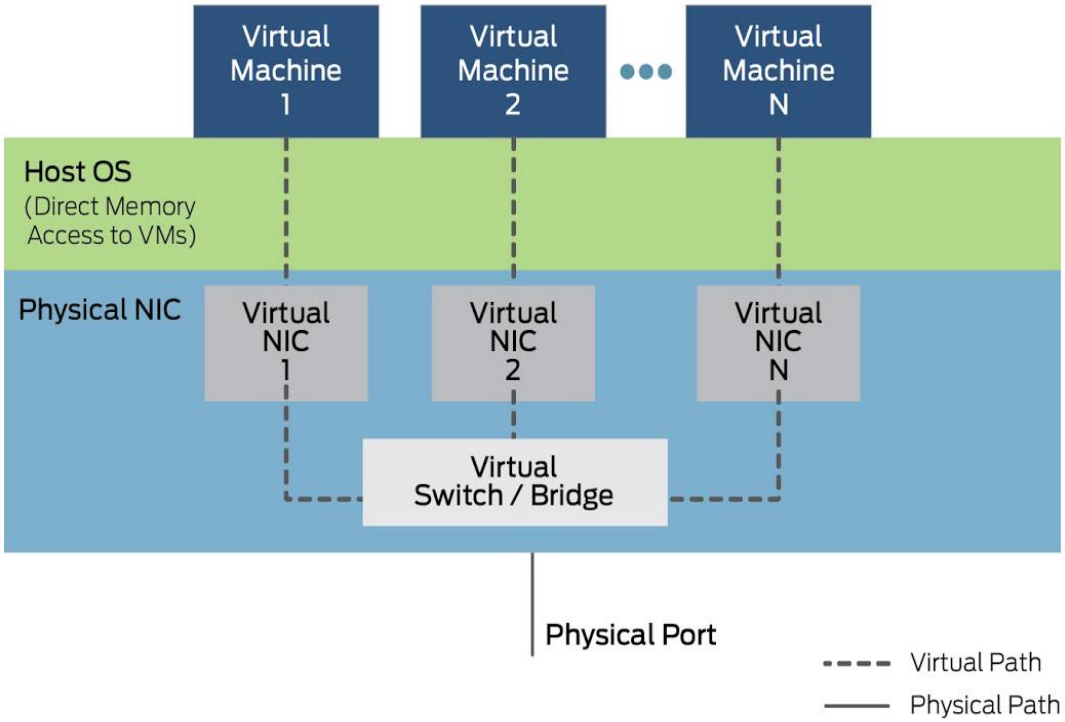
Kỹ thuật SR-IOV (Single Root - I/O Virtualization) do tổ chức PCI-SIG (PCI-Special Interest Group) định nghĩa có thể giúp hỗ trợ việc chia sẻ các thiết bị PCIe cho các máy ảo cùng sử dụng.

SR-IOV cho phép tạo ra nhiều hàm ảo (Virtual Function - VF) trên một thiết bị vật lý với tính năng tương tự như các thiết bị thật. Theo PCI SIG, một thiết bị PCIe vật lý có hỗ trợ SR-IOV có thể tạo ra đến 256 VF khác nhau.

Một số ví dụ:

- Một card mạng 4 port có hỗ trợ SR-IOV, mỗi port coi như 1 thiết bị vật lý. Theo giả thiết, ta có thể có tổng cộng 1 * 4 * 256 VF trên card mạng này, tổng là 1024 VF.
- Tương tự như vậy 1 host bus adapter có 2 port, hỗ trợ SR-IOV thì ta lại có 1 * 2 * 256 = 512 VF.

Dĩ nhiên, hai ví dụ kể trên chỉ là lý thuyết. Mỗi VF khi thực thi tác vụ đều cần sử dụng một phần tài nguyên vật lý thực sự, nên nếu có quá nhiều VF cùng xử lý thì sẽ dẫn đến tình trạng tranh chấp tài nguyên. Vì thế nếu muốn sử dụng hết khả năng của SR-IOV trên các thiết bị PCIe, người quản trị cần có những thử nghiệm và đánh giá thực sự lên chính cấu hình hệ thống đó.



Hình 2-10 Phân tách card mạng vật lý thành các card mạng ảo (Nguồn: Juniper)

Đối với các VM hoặc hypervisor, mỗi card mạng ảo (virtual NIC/VF) đều độc lập với nhau như các thiết bị thật riêng lẻ, thiết bị vật lý bên dưới sẽ không biết đến sự hiện diện của các VF.

SR-IOV cũng có một số giới hạn của mình. Ví dụ như: các VF phải cùng dạng với thiết bị vật lý thật bên dưới, các VF cũng không thể được dùng để cấu hình thiết bị vật lý thật bên dưới.

Ngoài ra, tuy vẫn còn những kỹ thuật giúp tối ưu hiệu năng khác như NUMA, Huge Pages, ASAP.... nhưng do giới hạn của luận văn, nhóm sẽ không đi sâu vào tìm hiểu các kỹ thuật này.

2.4.4. Khối điều phối và quản lý – NFV MANO

2.4.4.1. Tổng quan

Môi trường NFV trong thực tế là một môi trường đặc biệt bao gồm rất nhiều các thành phần phức tạp liên kết với nhau. Từ các hệ thống ảo hóa chạy (VMware vSphere, KVM), các thiết bị mạng vật lý, cho đến các máy ảo chứa các VNF ở bên trên và cả những liên kết giữa chúng. Tất cả các thành phần này sẽ liên kết lại với nhau nhằm tạo ra sản phẩm cuối là dịch vụ mạng cho người dùng.

Bài toán được đặt ra là làm thế nào để quản lý tất cả những thành phần này một cách tập trung và thống nhất. Hệ thống quản lý này cần có khả năng ổn định và tự động hóa cao, giảm bớt sự can thiệp của con người. Kiến trúc NFV MANO chính là lời giải cho bài toán bên trên với khả năng quản lý tập trung, tương thích được với nhiều loại phần cứng lẫn phần mềm và quan trọng nhất là khả năng điều phối chặt chẽ giữa các thành phần trong một hệ thống NFV.

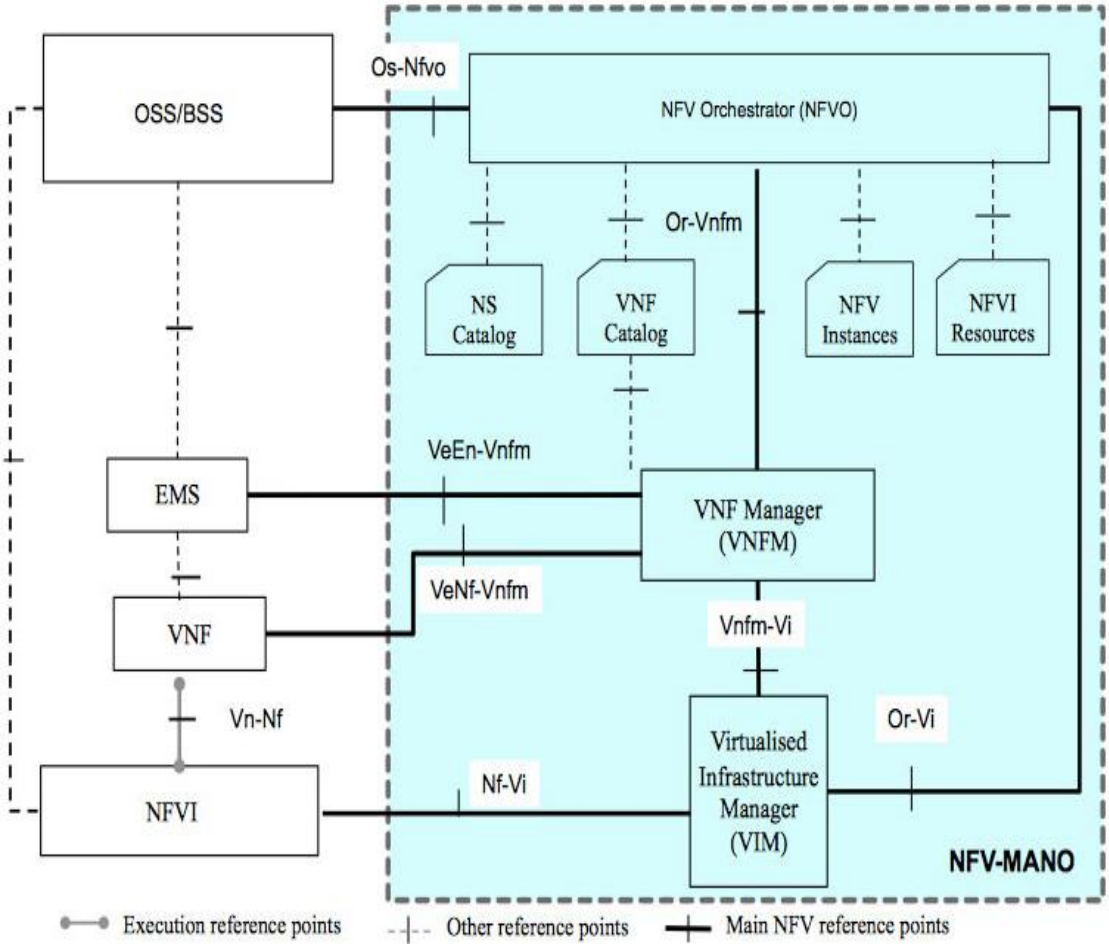
Chức năng chính của NFV MANO là quản lý NFVI và vòng đời của các VNF. Công việc cụ thể của NFV MANO như sau:

- Cấp phát và thu hồi tài nguyên của NFVI (tài nguyên xử lý, bộ nhớ, lưu trữ, kết nối…)
- Quản lý việc kết nối giữa các VM và VNF.
- Khởi tạo, mở rộng, phục hồi, nâng cấp hoặc xóa các VNF
- Theo dõi hiệu năng và các vấn đề khác liên quan đến NFVI

Một số giải pháp MANO tuân theo các quy chuẩn của ETSI là:

- Tacker: dự án thuộc OpenStack
- Open Source MANO: được phát triển bởi chính ETSI
- OpenBaton: dự án thuộc đại học Fraunhofer FOKUS (Đức)
- Cloudidy: công ty GigaSpaces

2.4.4.2. Kiến trúc



Hình 2-11 Sơ đồ khối MANO và các interface – Nguồn: ETSI

Về mặt kiến trúc, MANO gồm 3 khối con chính là NFVO, VNFM và VIM. Bên cạnh đó còn có các khối lưu trữ dữ liệu phục vụ cho các khối chính.

2.4.4.2.1. NFV Orchestrator

Chức năng cụ thể của NFV Orchestrator (NFVO) bao gồm khởi tạo, tinh chỉnh các dịch vụ mạng cũng như kết hợp các dịch vụ mạng lại với nhau. Bên cạnh đó, NFVO cũng quản lý tài nguyên toàn cục, chứng thực, cấp quyền và trực tiếp khởi tạo tài nguyên của NFVI thông qua VIM.

2.4.4.2.2. VNF Manager

Nhiệm vụ chính của VNF Manager (VNFM) là quản lý vòng đời của các VNF. Cụ thể, VNFM sẽ điều phối, tùy chỉnh cấu hình, cung cấp thông tin liên lạc giữa NFVO, VIM và EMS. Các tác vụ của một VNF Manager có thể là:

a) Quản lý vòng đời của VNF (khởi tạo/hủy, bật/tắt, thay đổi thông tin cấu hình, nâng cấp phần mềm, phục hồi khi có sự cố).
b) Mở rộng hay thu hẹp VNF khi cần.
c) Thu thập, giám sát các thông tin về hiệu suất hoạt động, các thông báo lỗi.
d) Làm cầu nối giữa phần mềm quản lý thực thể (EMS) bên trong các VNF và NFVO cũng như VIM.

Việc xây dựng VNF Manager như thế nào hoàn toàn phụ thuộc vào các nhà phát triển. Các nhà phát triển có hai lựa chọn là:

- Xây dựng một VNF Manager phổ thông (Generic VNF Manger) dùng chung cho mọi loại VNF. Đây là chiến lược phổ biến của các nhà phát triển MANO.
- Xây dựng VNF Manager đặc biệt cho một số VNF nhất định, ví dụ như VNF dùng riêng cho VoLTE (Voice over LTE). Các nhà phát triển MANO cung cấp các trình điều khiển, API của NFVO để giúp các nhà phát triển VNF tự xây dựng VNF Manager riêng cho sản phẩm VNF của mình.

2.4.4.2.3. Virtualized Infrastructure Manager

Nhiệm vụ của VIM là quản lý và điều phối các tài nguyên về tính toán, lưu trữ và kết nối mạng của NFVI. Các chức năng chính của VIM bao gồm:

- Quản lý các thông tin liên quan đến phần cứng và phần mềm của NFVI.
- Thực hiện việc phân phối, nâng cấp và thu hồi tài nguyên của NFVI cho các VNF sử dụng.
- Tạo các virtual link, virtual network, subnet, port mạng cũng như security policy nhằm quản lý lượng traffic dễ dàng hơn.

- Thu thập các thông tin về hiệu năng và lỗi của phần cứng, phần mềm và tài nguyên ảo hóa.

Hiện tại thì OpenStack là VIM lý tưởng nhất, do đã có quá trình phát triển lâu dài. Ngoài ra, trên thị trường thương mại, ta không thể không nhắc tới VMware vSphere, hiện đang được rất nhiều dự án về MANO hỗ trợ.

2.4.5. Các kho chứa dữ liệu của khối MANO

2.4.5.1. NFVI Resources Repository

NFVI Resources repository bao gồm các bản ghi mô tả về tài nguyên của các hạ tầng NFVI đã được kết nối tới hệ thống MANO thông qua VIM. Các bản ghi này sẽ liên tục được cập nhật để người quản trị theo dõi được lượng tài nguyên hạ tầng ảo hóa khả dụng của hệ thống, từ đó quyết định xem có thể triển khai, mở rộng dịch vụ được không.

2.4.5.2. NFV Instances repository

NFV Instances repository chứa tất cả các thông tin về các đối tượng VNF hay NS đã được khởi tạo. Mỗi một đối tượng sẽ được thể hiện dưới dạng 1 bảng ghi (record). Các record này sẽ được cập nhật liên tục trong suốt vòng đời của các đối tượng này mỗi khi phát sinh thay đổi. Việc này giúp NFVO và VNFM đảm bảo tính toàn vẹn và dễ dàng theo dõi các NS/VNF và mối quan hệ giữa chúng.

2.4.5.3. VNF Catalog

VNF Catalog là kho chứa tất cả các tập tin mẫu để triển khai các VNF. Mỗi một VNF sẽ có những thông tin cấu hình cũng như những chức năng riêng. Các thông tin này của từng VNF sẽ được mô tả trong các tập tin gọi là Virtualized Network Function Descriptor (VNFD). NFVO và VNFM đều có thể truy xuất VNF Catalog để lấy các tập tin VNFD cho nhiều tác vụ khác nhau như (khởi tạo VNF, kiểm tra tính khả dựng,...).

Việc định nghĩa các trường của các VNFD được ETSI thống nhất bằng những tiêu chuẩn chung. Bên cạnh đó, các tập tin VNFD cũng cần có cấu trúc tập tin chuẩn. Các ngôn ngữ thường thấy là UML, TOSCA-NFV simple profile và YANG:

- UML (The Unified Modeling Language): là ngôn ngữ phổ biến do Object Management Group đề xuất. ETSI sử dụng UML trong các tài liệu mô tả tiêu chuẩn về VNFD của mình.
- TOSCA-NFV (Topology and Orchestration Specification for Cloud Applications-NFV): là ngôn ngữ do OASIS đề xuất. Đây là một ngôn ngữ được thiết kế chuyên biệt cho điện toán đám mây. TOSCA mô tả một đồ thị các thực thể sẽ được khởi tạo và kết nối giữa chúng. Tiếp đó, TOSCA mô tả các sự kiện trong vòng đời của VNF, điều kiện chính sách, điều kiện ràng buộc giữa các VNF,...
- YANG: khác với UML và TOSCA, YANG chủ yếu hướng về mạng. YANG mô tả dữ liệu trạng thái và cấu hình của các thực thể mạng trong hệ thống.

```
tosca_definitions_version:
   This defines the TOSCA definition version on which the template is based.
   The current version being tosca_simple_profile_for_nfv_1_0_0.

tosca_default_namespace:
   This is optional. It mentions default namespace which includes schema,
   types version etc.

description:
   A short description about the template.

metadata:
   template_name: A name to be given to the template.

topology_template:
   Describes the topology of the VNF under node_template field.
   node_template:
       Describes node types of a VNF.
       VDU:
           Describes properties and capabilities of Virtual Deployment
           Unit.
       CP:
           Describes properties and capabilities of Connection Point.
       VL:
           Describes properties and capabilities of Virtual Link.
```

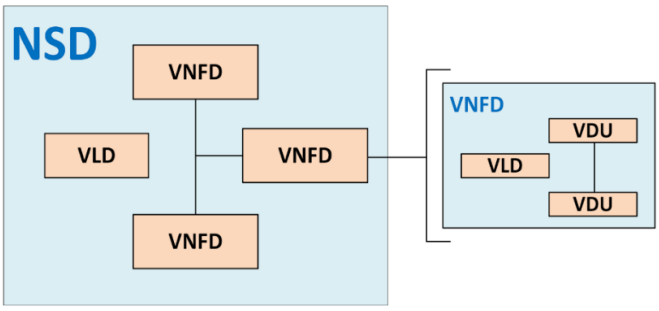
Hình 2-12 Tập tin VNFD sau được viết theo cấu trúc TOSCA

Trong ví dụ, ta thấy được một số thông tin cơ bản của một VNFD bao gồm: phiên bản TOSCA, mô tả về VNFD, metadata mô tả tên của VNFD, các thông tin mô tả về

các máy ảo sẽ triển khai (Virtual Deployment Unit), các interface sẽ tạo (Connection Point), kết nối mạng ảo sẽ sử dụng (Virtual Link),... Khi nạp một VNFD lên kho chứa, khối MANO sẽ kiểm tra tính đúng đắn của các trường trong tập tin VNFD rồi mới quyết định có nạp lên hay không.

2.4.5.4. Network Service Catalog

Network Service Catalogue là kho chứa tất cả các tập tin mẫu mô tả các dịch vụ mạng Network Service Descriptor (NSD). Một dịch vụ mạng có thể được tạo nên bởi một hay nhiều VNF thành phần. Tương tự, một NSD cũng sẽ bao gồm các VNFD thành phần.



Hình 2-13 Mối liên hệ giữa NSD và VNFD

Một NSD ví dụ sẽ có dạng:

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
imports:
  - VNFD1
  - VNFD2
topology_template:
  node_templates:
    VNF1:
      type: tosca.nodes.nfv.VNF1
      requirements:
        - virtualLink1: VL1
        - virtualLink2: VL2
    VNF2:
      type: tosca.nodes.nfv.VNF2
    VL1:
      type: tosca.nodes.nfv.VL
      properties:
      network_name: net0
      vendor: tacker
    VL2:
      type: tosca.nodes.nfv.VL
      properties:
          network_name: net_mgmt
          vendor: tacker
```

Hình 2-14 Ví dụ một Network Service Descriptor theo chuẩn TOSCA

Tập tin NSD ví dụ này cũng tuân theo chuẩn TOSCA Simple Profile. Thông qua ví dụ, ta thấy được những thông tin cơ bản của một NSD như:

- Tên các VNFD thành phần: ở đây là VNFD1 và VNFD2
- Tên các VNF thành phần sẽ được khởi tạo từ các VNFD trên: VNF1 và VNF2
- Tên các mạng ảo sẽ sử dụng để tạo kết nối giữa các VNF: đường mạng net0 ứng với VL1 và đường mạng net_mgmt ứng với VL2.

Khi nạp một NSD lên trên kho chứa, khối MANO sẽ kiểm tra tính hợp lệ của các trường trong NSD đó như: tên các VNFD thành phần, tên các mạng ảo sẽ sử dụng,... trước rồi mới quyết định nạp NSD đó lên hay không.

Ngoài ra, để cho thuận tiện trong việc phát hành, các NSD và cả các VNFD thường được đóng gói chung lại thành một tập tin duy nhất chứa tất cả các tập tin mô tả cấu hình, đoạn mã cài đặt, image để cài đặt VM chạy VNF,...

Ví dụ: một tập tin zip theo chuẩn CSAR (Cloud Service Archive) có dạng sau:

```
├── Definitions
|   └── testNSDiperf.yaml
├── Scripts
|   ├── install.sh
|   └── (VNF NAME)
|         └── script.sh
└── TOSCA-Metadata
    ├── Metadata.yaml
    └── TOSCA.meta
```

Hình 2-15 Cấu trúc một tập tin NSD có đuôi zip theo chuẩn CSAR

Trong đó: Tập tin CSAR có thể bao gồm một NSD và các VNFD thành phần của nó hoặc chỉ bao gồm một VNFD riêng lẻ:

- Thư mục Definitions: chứa các tập tin NSD, VNFD theo chuẩn TOSCA.
- Thư mục scripts: chứa các script cài đặt cho NS, các VNF thành phần.
- Thư mục TOSCA-Metadata: chứa các tập tin mô tả thông tin về gói CSAR này.

2.5. Mối liên hệ giữa NFV và SDN

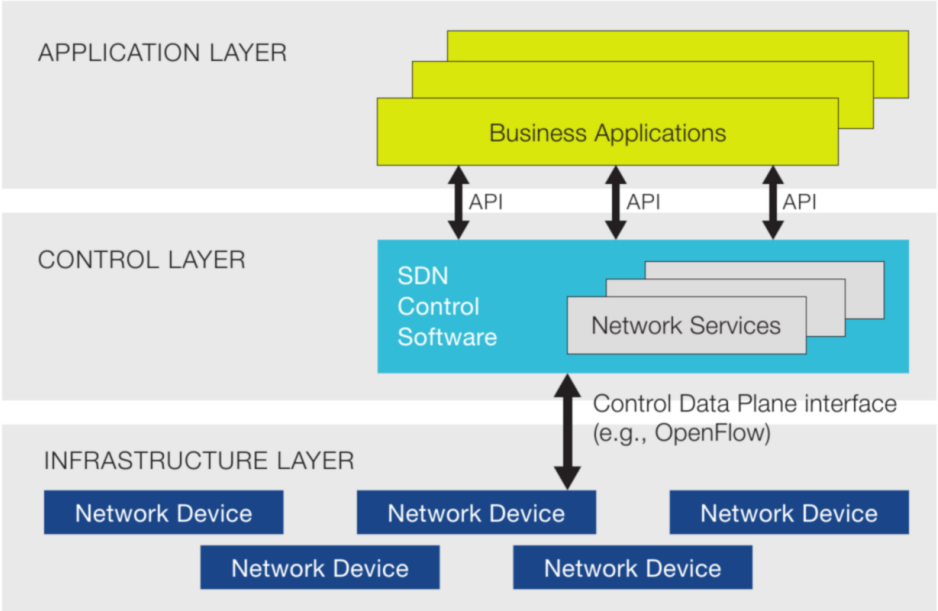
2.5.1. Sơ lược về SDN

Theo như tổ chức Open Networking Forum: *"SDN là một mô hình kiến trúc mạng mới với những đặc tính như: linh động, dễ dàng quản lý, tiết kiệm chi phi, khả năng tương thích cao. Đây chính là những đặc tính khiến cho SDN trở thành 1 mô hình lý tưởng cho các ứng dụng tiêu tốn nhiều băng thông và thay đổi liên tục trong thời đại hiện nay*".

Kiến trúc SDN phân tách rõ rệt giữa hai chức năng: điều khiển mạng (control plane) và chuyển mạch (data plane). Từ đó mở ra khả năng phân tách giữa hạ tầng mạng bên dưới và các dịch vụ, ứng dụng mạng chạy bên trên. Đồng thời, cho phép người quản trị có thể trực tiếp lập trình, điều khiển đường đi trong mạng.

Một kiến trúc SDN gồm có những thành phần sau:

- **Controller:** bộ não của hệ thống, controller cung cấp một cái nhìn toàn cảnh về hệ thống mạng cho người quản trị. Toàn bộ việc điều chỉnh các luồng traffic đều được thực hiện tại đây. Trong hình dưới, đây chính là lớp điều khiển (Control Layer).
- **Northbound API:** là các API dùng để tương tác với các ứng dụng của người dùng ở trên. Các API này hỗ trợ người quản trị lập trình và điều khiển các dịch vụ mạng. Trong hình dưới, đây chính là các API ở giữa lớp điều khiển (Control Layer) và lớp ứng dụng (Application Layer).
- **Southbound API:** đây là những API dùng để truyền tải lệnh điều khiển từ controller xuống cho các switch và router vật lý ở dưới. Trong hình dưới, đây chính là các API ở giữa lớp điều khiển và lớp hạ tầng (Infrastructure Layer). Đại diện cho các API này là OpenFlow. Đây cũng chính là thuật toán phổ biến nhất hiện nay dùng để tương tác với các thiết bị mạng. Giao thức OpenFlow chính là nền tảng cho các giải pháp SDN.



Hình 2-16 Kiến trúc tổng quát SDN

Hướng tiếp cận phân tách giữa Control Plane và Data Plane của SDN đem lại rất nhiều lợi ích cho các bên liên quan:

- Đối với các nhà sản xuất phần mềm, việc phân chia này giúp cho họ có thể tập trung hơn vào sản phẩm của mình, không còn quá phụ thuộc vào phần cứng của một hãng nhất định.
- Cho phép người sử dụng được tự do lựa chọn, kết hợp giữa nhiều loại phần mềm điều khiển mạng và phần cứng khác nhau nhằm đạt được hiệu suất cao nhất.
- Cho phép quản lý tập trung toàn bộ hạ tầng mạng, điều này đặc biệt quan trọng đối với các nhà cung cấp dịch vụ viễn thông có số lượng thiết bị lớn, đặt tại nhiều data center khác nhau.

2.5.2. Mối liên hệ giữa NFV và SDN

Nếu như khái niệm NFV xuất phát từ nhu cầu của các nhà cung cấp dịch vụ muốn giảm bớt chi phí đầu tư các thiết bị phần cứng bằng cách ảo hóa các thiết bị mạng để có thể triển khai các dịch vụ mạng trên phần cứng phổ thông, thì SDN lại xuất phát từ các trường đại học, viện nghiên cứu, data center muốn tách bạch việc điều khiểu

mạng khỏi các thiết bị vật lý, để dễ dàng cấu hình, quản lý tập trung một lượng lớn các thiết bị này.

Về bản chất, hai công nghệ này là độc lập với nhau. Bên này có thể áp dụng được vào thực tiễn mà không cần phụ thuộc vào bên kia. Thật khó để nói được SDN hay NFV, công nghệ nào tốt hơn. Vì như đã so sánh rõ ở trên, hai công nghệ này phục vụ cho hai mục đích hoàn toàn khác nhau.

Tiêu chí so sánh	SDN	NFV
Mục đích	Phân tách giữa control plane và data plane, quản lý tập trung, cấu hình mạng bằng cách lập trình	Chuyển dời các chức năng mạng từ phần cứng chuyên dụng sang các thiết bị phổ thông.
Đối tượng phục vụ	Các viện nghiên cứu, trung tâm dữ liệu	Các nhà cung cấp dịch vụ mạng.
Thiết bị	Máy chủ, thiết bị chuyển mạch phổ thông	Máy chủ, thiết bị chuyển mạch và lưu trữ phổ thông
Ứng dụng	Điều phối mạng.	Các thiết bị mạng như: router, firewall, CDN,…
Tổ chức chuẩn hóa	Open Networking Forum	ETSI NFV Working Group

Bảng 2-2 So sánh giữa SDN và NFV

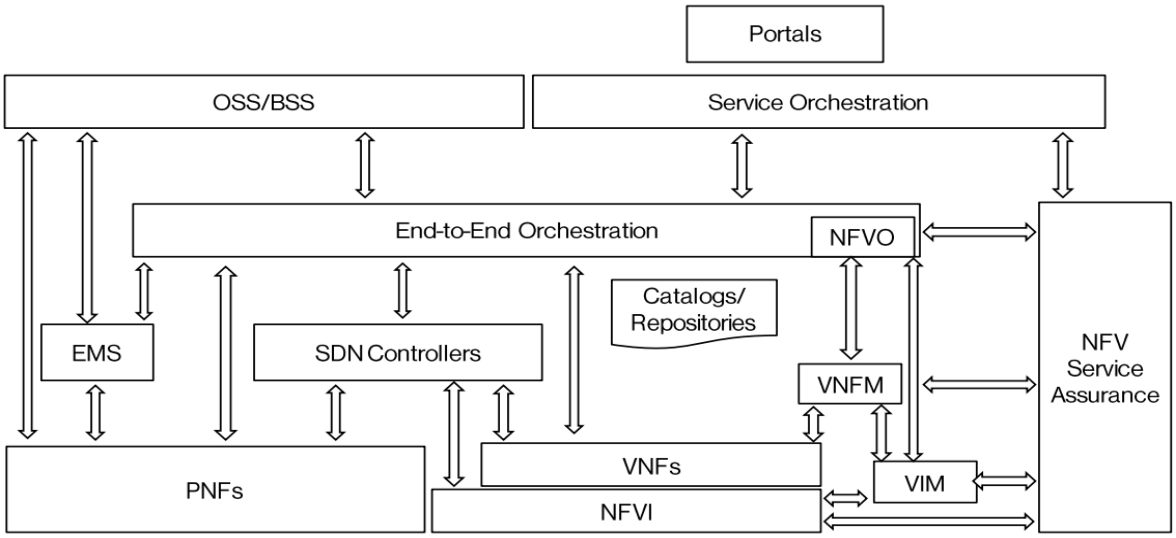
Mục tiêu chung của cả SDN và NFV là điều khiển hạ tầng mạng dễ dàng hơn, tiết kiệm chi phí và hạn chế việc tương tác trực tiếp với các thiết bị phần cứng. Như vậy, ta có thể thấy rằng hai công nghệ này không hề đối chọi với nhau mà còn lại bổ sung, hoàn thiện lẫn nhau, tạo nên một giải pháp hoàn chỉnh cho ngành viễn thông.

Việc quản lý tập trung của SDN kết hợp với khả năng ảo hóa các thiết bị mạng của NFV sẽ đem lại những lợi ích vô cùng lớn với hạ tầng viễn thông. Đặc biệt là việc chuẩn bị cho công nghệ 5G sắp tới thì việc ứng dụng hai công nghệ này tạo nên nền tảng cho mạng 5G.

2.5.3. *Kết hợp SDN vào NFV*

Mục tiêu chính của công nghệ NFV chính là khởi tạo và cấu hình các thiết bị mạng một cách nhanh chóng. Tuy công nghệ NFV có thể điều chỉnh luồng dữ liệu khi khởi tạo các thiết bị mạng nhưng khó có thể điều chỉnh lại các luồng dữ liệu đã được thiết lập này.

Chưa kể đến một số tính năng nâng cao như lọc gói tin, header, QoS thì các giải pháp NFV hiện nay vẫn còn thiếu rất nhiều tính năng và không thể so sánh được với công nghệ SDN, vốn chuyên dùng để điều chỉnh các luồng dữ liệu.



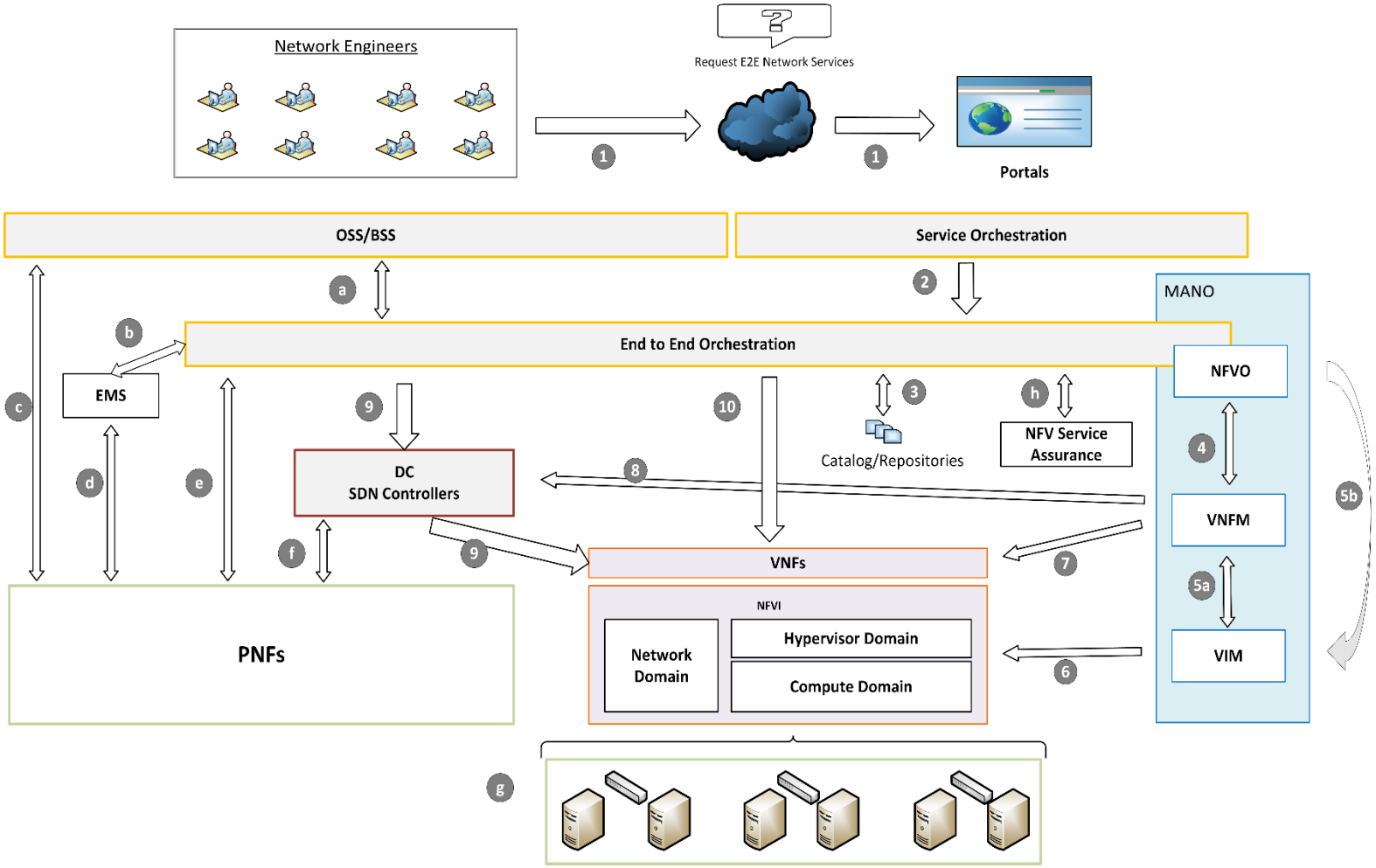
Hình 2-17 Mô hình kiến trúc tổng quan 1 hạ tầng NFV + SDN do Verizon đề ra

Chính vì vậy, kết hợp SDN vào hạ tầng NFV chính là lời giải đáp hoàn chỉnh cho vấn đề này.

Trong mô hình này việc điều chỉnh luồng dữ liệu sẽ được chia làm 2 giai đoạn:

- **Giai đoạn 1:** là giai đoạn khởi tạo ban đầu do khối MANO của NFV đảm nhận. Cụ thể, thông tin về đường mạng, các liên kết giữa các VNF/VNFC cũng như việc kết nối chúng lại thành NS hoàn chỉnh sẽ được thể hiện trong các tập tin đặc tả. Khối MANO sẽ đọc các tập tin này, cấp phát tài nguyên rồi khởi tạo thành dịch vụ mạng hoàn chỉnh.

- **Giai đoạn 2:** là giai đoạn sau khởi tạo. Sau khi hệ thống đã được triển khai hoàn chỉnh, việc điều chỉnh luồng lúc này sẽ được giao cho SDN Controller phụ trách. Khi xuất hiện luồng dữ liệu thỏa các bộ luật (rule) đã định nghĩa từ trước, SDN Controller sẽ điều chỉnh luồng dữ liệu đi qua các VNF theo một lộ trình đã được định sẵn. Người điều khiển hoàn toàn có thể lập trình sẵn hoặc điều chỉnh lại các tập rule để điều chỉnh luồng cho thích hợp với yêu cầu dịch vụ.



Hình 2-18: Workflow hoàn chỉnh của giải pháp NFV kết hợp SDN

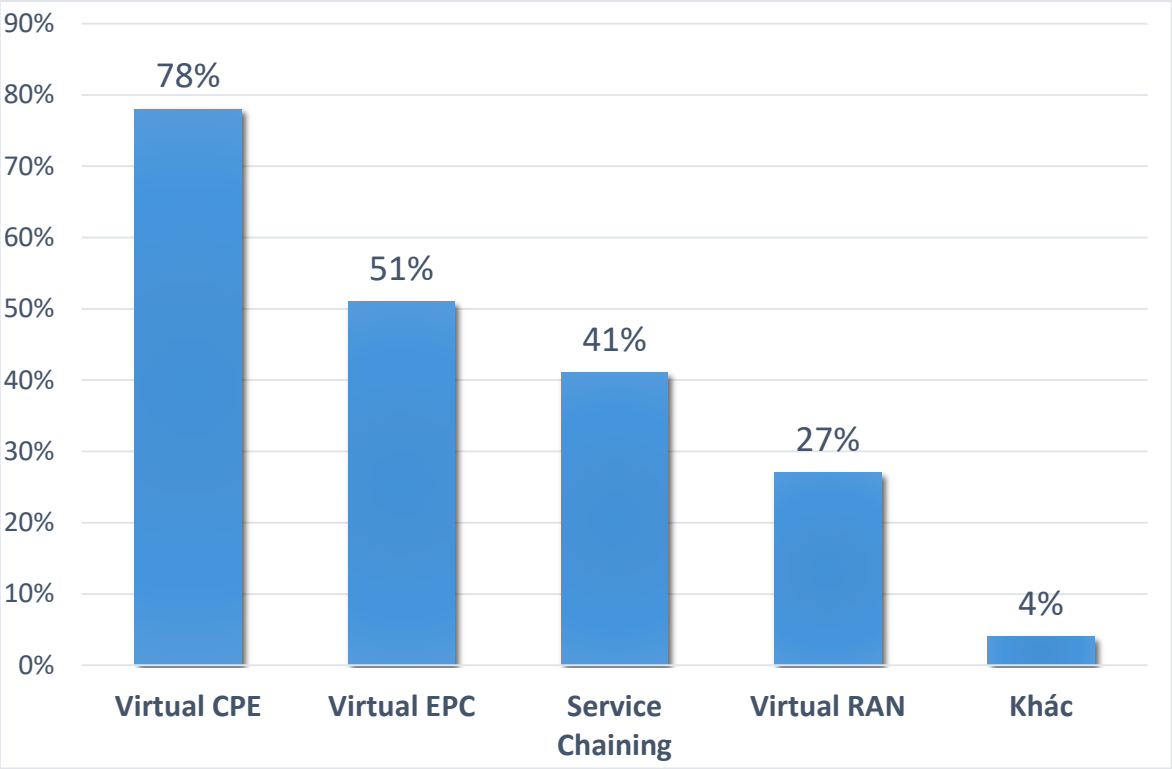
STT	Mô tả
1	Network Engineers khởi tạo một dịch vụ hoàn chỉnh thông qua portal (giao diện đồ họa hoàn chỉnh). Portal này sẽ có sẵn danh sách các dịch vụ mà hệ thống hỗ trợ (NAT, VPN, Load Balance…) cùng các thông số đi kèm.
2	Các thông tin, thông số của dịch vụ được End-to-End Orchestator (EEO) tiếp nhận.
3	EEO lấy thông tin mô tả dịch vụ (End-to-End Service Descriptor) từ các file template được định nghĩa trước. Việc này nhằm xác định lượng tài nguyên cần thiết và vị trí cài đặt các VNF mới.
4	NFVO (lúc này là một thành phần trong EEO) và VNFM liên lạc với nhau để thực hiện các bước chuẩn bị để khởi tạo VNF như: kiểm tra lượng tài nguyên khả dụng, cấp phép cấp phát tài nguyên,..
5a	VNFM liên hệ với VIM để yêu cầu tạo các *máy ảo* cần thiết để cho VNF chạy lên. *(VNFM driven*).
5b	NFVO liên hệ với VIM để yêu cầu tạo các *máy ảo* cần thiết để cho VNF chạy lên. *(NFVO driven*).
6	VIM liên hệ với NFVI để yêu cầu tài nguyên phần cứng cần thiết nhằm khởi tạo VM.
7	VNFM tiến hành cài đặt VNF lên các VM vừa được tạo.
8	Đối với những VNFs nào gồm nhiều VM, VNFM sẽ yêu cầu SDN Controller để tạo kết nối giữa các VM trong cùng một VNF.
9	EEO yêu cầu SDN Controller để tạo kết nối giữa các VNF lại thành một chuỗi dịch vụ hoàn chỉnh.
10	EEO cung cấp các thông tin đặc thù của khách hàng xuống cho các hàm dịch vụ mạng. Ví dụ như tập rule đối với Firewall, IDS,..

Bảng 2-3 Bảng chú thích cho hình 2–18

STT	Mô tả
a	OSS/BSS liên hệ với EEO để yêu cầu dịch vụ
b	EEO thông qua EMS để quản lý các thiết bị mạng vật lý (*PNFs*)
c	OSS/BSS quản lý trực tiếp các *PNFs* bên dưới
d	EMS quản lý PNFs
e	EEO quản lý trực tiếp *PNFs* bên dưới
f	SDN Controller tạo kết nối giữa PNFs - PNFs hoặc giữa PNFs - VNFs. Tận dụng lại hạ tầng mạng sẵn có.
g	Phần cứng bên dưới của NFVI gồm các switch, server có hiệu năng cao, ko phụ thuộc vào công nghệ của các hãng (high volume servers, switches).
h	Service Assurance (SA) thu thập thông tin alarm, monitor hệ thống. Những thông tin này có thể được dùng để sửa lỗi, phân tích tình hình…

Bảng 2-4 Bảng phụ lục cho hình 2-18

2.6. Các ứng dụng thực tiễn tiêu biểu của NFV.



Hình 2-21 Danh sách các ứng dụng phổ biến nhất của NFV năm 2016 (Nguồn: ADVA)

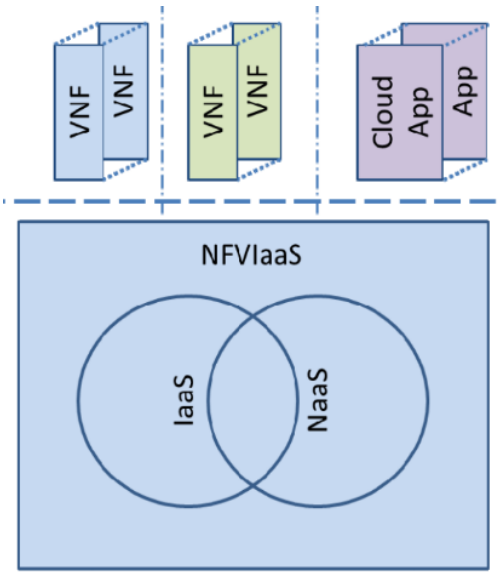
Từ những lợi ích đã đề cập, ETSI đã đề xuất ra một số ứng dụng thực tiễn cụ thể cho công nghệ NFV. Trong phạm vi luận văn này, nhóm sẽ tóm tắt lại thành một số ứng dụng thực tiễn lớn như sau:

2.6.1. Theo ETSI

2.6.1.1. Dịch vụ hạ tầng ảo hóa chức năng mạng

Khi tiến hành xây dựng hạ tầng ảo hóa để triển khai NFV, các nhà cung cấp dịch vụ thường sẽ nghĩ đến việc xây dựng các hạ tầng ảo hóa của mình theo mô hình đám mây nội bộ (Private Cloud). Tuy vậy, vấn đề là sẽ khó có nhà cung cấp dịch vụ nào có thể trải rộng hạ tầng của mình đủ rộng khắp để phục vụ khách hàng ở mọi nơi trên thế giới. Hay với các nhà cung cấp dịch vụ có qui mô nhỏ, việc đầu tư xây dựng các hạ tầng ảo hóa qui mô sẽ là bài toán lớn.

Giải pháp ở đây sẽ là các nhà cung cấp dịch vụ lớn đủ tiềm lực hoặc ở vị trí địa lý phù hợp có thể tự xây dựng hạ tầng ảo hóa NFV (NFV Infrastructure) của mình rồi cung cấp lại cho các cung cấp dịch vụ nhỏ hơn hoặc ở vùng địa lý khác sử dụng lại. Mô hình này được gọi là NFV Infrastructure as a Service (NFVIaaS). NFVIaaS sẽ bao gồm cả dịch vụ hạ tầng tính toán, lưu trữ (Infrastructure as a Service - IaaS) và dịch vụ hạ tầng mạng (Network as a Service - NaaS).



Hình 2-19 Một mô hình NFVIaaS tiêu chuẩn

Lấy ví dụ nhà cung cấp dịch vụ A sẽ cho nhà cung cấp dịch vụ B sử dụng hạ tầng ảo hóa của mình để chạy một phần hệ thống CDN của B. Nhà cung cấp dịch vụ B sẽ chỉ có thể quản lý các VNF CDN của B và hoàn toàn không thể tác động gì tới các hạ tầng ở bên dưới của A.

Ngoài ra, mô hình này còn có thể mở rộng bằng cách nhà cung cấp dịch vụ sẽ cài đặt thêm vào hạ tầng của mình những công cụ hỗ trợ để tạo thành một nền tảng cho phép các doanh nghiệp khách hàng khác (không phải là các nhà cung cấp dịch vụ mạng) sử dụng để xây dựng các ứng dụng, dịch vụ của mình. Mô hình mở rộng này có thể coi như một dịch vụ điện toán đám mây theo mô hình dịch vụ nền tảng (Platform as a Service - PaaS).

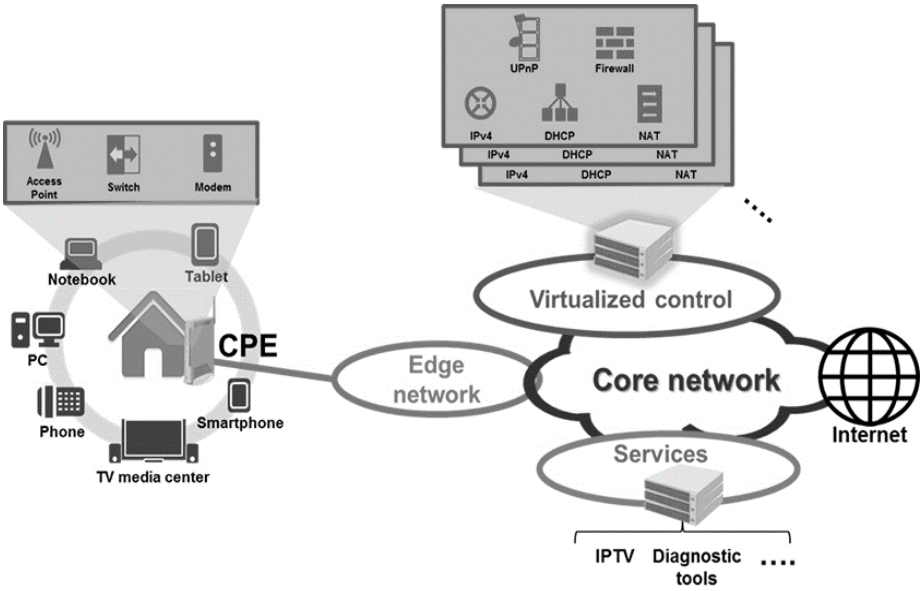
2.6.1.2. Ảo hóa hạ tầng viễn thông và di động

Hạ tầng viễn thông và di động truyền thống đòi hỏi rất nhiều thiết bị mạng chuyên dụng phức tạp và đắt đỏ. Đó có thể là các thiết bị trong kiến trúc Evolved Packet Core (EPC) của mạng di động lõi (Mobile Core Network), các trạm phát sóng tín hiệu di động (Mobile Base Station),…

Việc ứng dụng NFV cho phép các nhà mạng viễn thông, di động thay thế một phần hoặc toàn bộ các thiết bị chuyên dụng cũ bằng các máy chủ, thiết bị chuyển mạch hay lưu trữ phổ thông hơn. Từ đó, các nhà mạng sẽ tiết kiệm được rất nhiều chi phí đầu tư và vận hành, tiết kiệm điện năng tiêu hao cũng như tăng hiệu suất, độ linh hoạt, tính sẵn sàng và ứng phó tốt trước các sự cố (có thể có) của hệ thống.

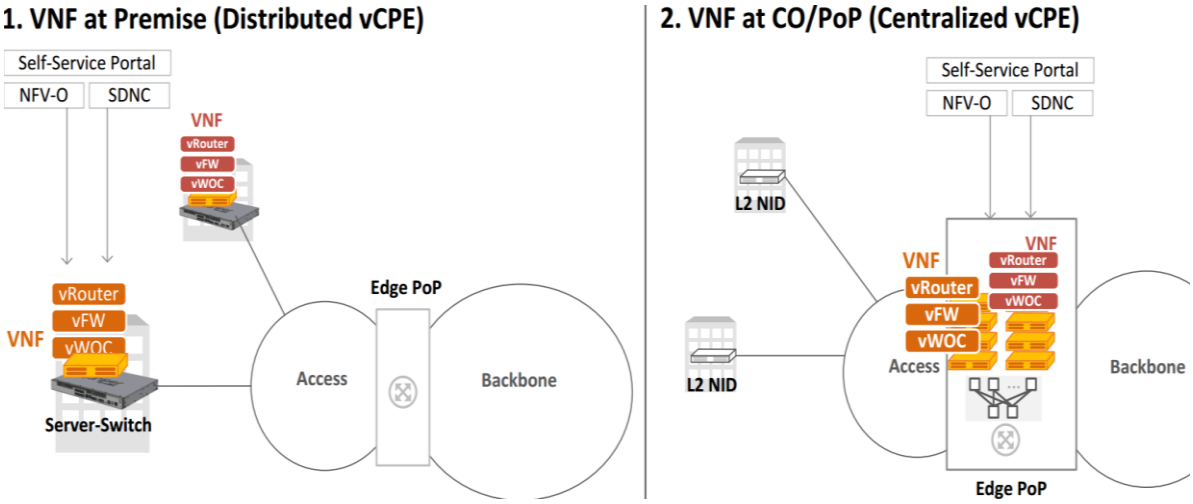
Một trong những ứng dụng tiêu biểu nhất chính là vCPE (virtual Customer Premises Equipment). CPE là các thiết bị mạng được lắp đặt tại phía khách hàng như: Router Internet, hay Setup Box,… Khi áp dụng công nghệ NFV, các thiết bị này sẽ được ảo hóa hoàn toàn hoặc một phần và trở thành các VNF. Điều này giúp các nhà cung cấp dịch vụ cắt giảm chi phí đầu tư, vận hành, bảo dưỡng, nâng cấp cũng như đơn giản hóa việc quản trị.

Bên cạnh đó, khách hàng không còn cần phải quan tâm đến các bước cấu hình phức tạp các dịch vụ như Internet, điện thoại, truyền hình Internet,... Thay vào đó, khách hàng sẽ được cung cấp một cổng portal để có thể truy cập và theo dõi các dịch vụ đang sử dụng cũng như dễ dàng yêu cầu mở rộng hay kết thúc dịch vụ.



Hình 2-20 Ảo hóa mạng tại căn hộ của người sử dụng

Ở đây nhà cung cấp sẽ có hai cách tiếp cận về vị trí đặt của thiết bị chạy vCPE:



Hình 2-21 Distributed vCPE và Centralized vCPE

- **VNF at Premise (Distributed vCPE)**

Lúc này, thiết bị chạy vCPE được đặt tại chính vị trí của khách hàng và được tối ưu hóa để có thể chạy các VNF theo nhu cầu sử dụng. Khách hàng có thể sử dụng các tiện ích của các hàm ảo hóa này mà không cần can thiệp gì về phần cứng và phần mềm. Việc quản lý phiên bản phần mềm, nâng cấp, mở rộng đều được nhà cung cấp dịch vụ cấu hình từ xa.

Cách làm này phụ thuộc rất nhiều vào sự ổn định của mạng truy cập cũng như giới hạn phần cứng mà thiết bị được thiết kế ban đầu. Bên cạnh đó, các khách hàng không thể chia sẻ tài nguyên ảo hóa với nhau dẫn đến gây lãng phí lớn. Nhưng nhìn chung, chiến lược này khả thi và có thể thực hiện dễ dàng vì sẽ không thay đổi quá nhiều kiến trúc tổng thể của cả hạ tầng mạng mà chỉ cung cấp cho khách hàng những thiết bị CPE mới tối ưu hơn về giá thành và hiệu năng.

- **VNF at CO/PoP (Centralized vCPE)**

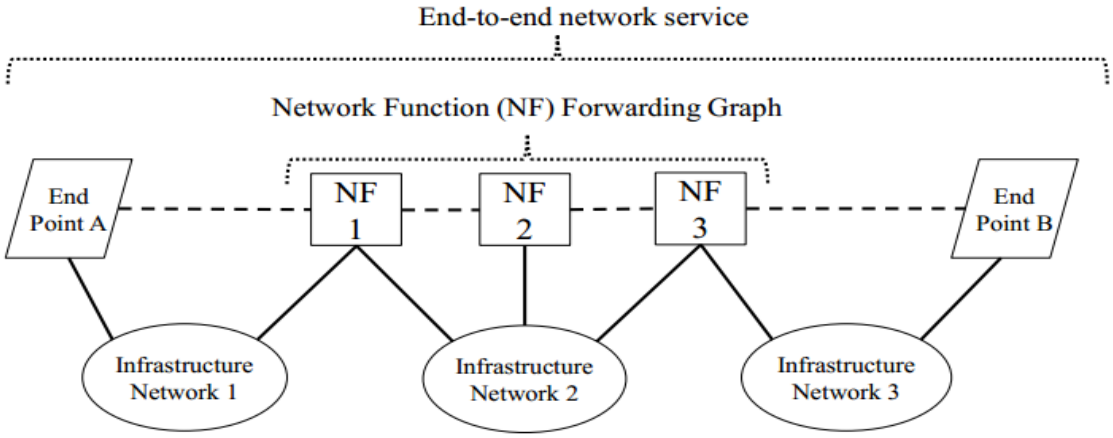
Thiết bị vCPE đặt tại trung tâm dữ liệu của nhà cung cấp dịch vụ**.** Lúc này, ứng với mỗi khách hàng, nhà cung cấp dịch vụ sẽ có một vCPE ở vùng biên (virtual Edge CPE) cung cấp các dịch vụ mạng cho khách hàng đó. Lúc này, khách hàng chỉ còn cần phải có một thiết bị thu/phát và khuyếch đại tín hiệu đơn giản để kết nối vào hệ thống mạng của nhà cung cấp dịch vụ là đã có thể sử dụng dịch vụ mong muốn.

Cách làm này giúp giảm rất nhiều chi phí đầu tư và vận hành thiết bị CPE và thuận tiện hơn cho việc quản trị cũng như tính cước. Tuy nhiên, nhược điểm của cách làm này đó là nhà cung cấp dịch vụ cần phải tốn khá nhiều chi phí xây dựng ban đầu để nâng cấp hoặc xây mới các trung tâm dữ liệu của mình.

Trong một số kiến trúc thực tế, ta sẽ thấy có sự kết hợp thêm của công nghệ SDN trong việc ứng dụng vCPE để tối ưu hơn nữa về hiệu năng cũng như quản lý luồng dữ liệu.

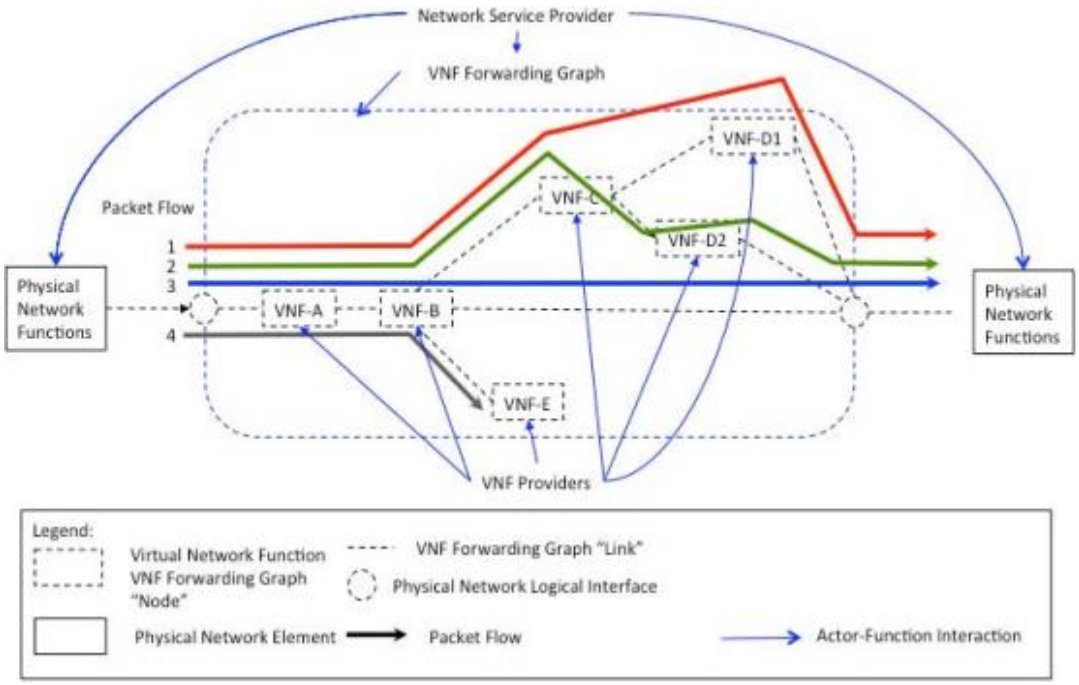
2.6.1.3. Kết nối các dịch vụ mạng (Service Chaining)

Dù bản thân một hàm chức năng mạng (Network Function - NF) đã có thể trở thành một dịch vụ mạng nhưng để tối đa hóa lợi ích của các thiết bị ảo này, các nhà cung cấp dịch vụ thường xâu chuỗi các NF lại với nhau (Function Chaining hay Service Chaining) để tạo nên các dịch vụ phức tạp cho các hệ thống mạng viễn thông của mình. Một chuỗi các hàm chức năng mạng như vậy được gọi là đồ thị chuyển tiếp hàm chức năng mạng (Network Function Fowarding Graph).



Hình 2-22 Một Network Service gồm 3 Network Function

Tương tự khi áp dụng NFV, các VNFs có thể kết nối lại với nhau và với các PNFs sẵn có để tạo thành một dịch vụ hoàn chỉnh. Một chuỗi các hàm chức năng mạng như vậy được gọi là đồ thị chuyển tiếp chức năng mạng ảo (Virtual Network Function Fowarding Graph - VNFFG).



Hình 2-23 Một VNF Forwarding Graph phức tạp gồm nhiều luồng khác nhau

Ưu điểm dễ nhận thấy nhất của cách làm này là do các hàm chức năng này chạy trên môi trường ảo hóa nên tính sẵn sàng và độ linh hoạt rất cao. Người quản trị có thể triển khai, co dãn, thu hồi hay phục hồi khi có sự cố các VNF một cách nhanh chóng.

Bên cạnh đó, người quản trị cũng có thể lập trình để thay đổi đường đi của luồng dữ liệu qua các hàm chức năng này một cách tự động, linh hoạt và tập trung.

2.6.2. Tiềm năng ứng dụng NFV tại Việt Nam

Trong môi trường thực tế tại Việt Nam, nhóm nghiên cứu nhận thấy việc ứng dụng NFV vào thực tiễn có thể giải quyết các bài toán cụ thể.

2.6.2.1. Các nhà cung cấp dịch vụ Internet và viễn thông

Một số đại diện tiêu biểu cho lĩnh vực này ở Việt Nam là: Viettel, VNPT, FPT, CMC… Đối với những doanh nghiệp này thì lợi ích mà NFV mang lại là rất lớn:

- Áp dụng ảo hóa chức năng mạng vào hạ tầng mạng viễn thông và di động: sẽ giúp việc triển khai, quản lý các dịch vụ trở nên dễ dàng, nhanh chóng, linh hoạt và tối ưu hơn trên trên các thiết bị phần cứng phổ thông. Đặc biệt là khi kết hợp với SDN, các nhà mạng còn có thể tối ưu hóa việc truyền dẫn cũng như cắt giảm chi phí nếu có thể chia sẻ tài nguyên về hạ tầng kết nối với nhau.
- Áp dụng dịch vụ hàm chức năng mạng ảo: giúp quản lý tốt hơn dịch vụ Internet, Thoại, Truyền hình Internet,... đến người dùng. Bên cạnh đó, họ còn có thể cắt giảm chi phí triển khai, bảo trì, sửa chữa, đào tạo con người cho các thiết bị CPE ở tại nhà khách hàng.

2.6.2.2. Các công ty, tổ chức, đơn vị sự nghiệp

Đối với những công ty kinh doanh dựa trên môi trường Internet như: FilmPlus, K+, VNG, hoặc các trường đại học... Ứng dụng công nghệ NFV vào hạ tầng của những công ty này cũng đem lại những lợi ích không nhỏ:

- Sử dụng dịch vụ hàm chức năng mạng ảo: như Load balancing, Firewall của các nhà cung cấp dịch vụ viễn thông, di động giúp họ tiết kiệm chi phí đầu tư và vận hành các thiết bị CPE.
- Ảo hóa hạ tầng phần cứng để tiết kiệm chi phí mua sắm thiết bị mới và tận dụng hạ tầng đã có.

Chương 3 Các giải pháp triển khai

Trên thị trường hiện nay hiện có rất nhiều các giải pháp về NFV, cả nguồn mở lẫn thương mại. Trong phạm vi của luận văn này, nhóm sẽ tập trung tìm hiểu, khảo sát nhanh và giới thiệu một số giải pháp nguồn mở về MANO tiêu biểu nhất. Từ đó chọn ra các giải pháp để tiến hành triển khai thử nghiệm.

Còn đối với khối NFVI và VIM, nhóm sẽ sử dụng giải pháp OpenStack. Đây là một trong những giải pháp xây dựng hạ tầng điện toán đám mây tiêu biểu và đã được sử dụng rộng rãi. Bên cạnh đó, nhóm cũng sẽ tìm hiểu giải pháp xây dựng nền tảng NFV trọn gói và tự động là OPNFV.

3.1. Khối hạ tầng ảo hóa – NFVI

3.1.1. OpenStack

Openstack là một trong những giải pháp xây dựng hạ tầng ảo hóa tiêu biểu nhất. Một số dự án con của OpenStack có thể kể đến như:

- Compute (Nova): Quản lý tài nguyên tính toán (CPU, RAM) của các máy ảo.
- Network (Neutron, networking-sfc): Quản lý tài nguyên mạng, khả năng liên kết giữa các máy ảo.
- Storage (Cinder): Quản lý tài nguyên lưu trữ (HDD, SDD) của các máy ảo.
- Orchestration (Heat): Điều phối các ứng dụng điện toán đám mây thông qua các template được định nghĩa sẵn.
- Workflow (Mistral): Quản lý và thực hiện qui trình các tác vụ cho Tacker và các dịch vụ khác.
- Telemetry (Ceilometer): Đo lường, thu thập dữ liệu từ các dịch vụ khác trong hệ thống OpenStack.

Nền tảng OpenStack cung cấp một nền móng vững chắc cho kiến trúc NFV, phù hợp cho mục đích triển khai, điều phối và quản lý các VNF. OpenStack cũng cho phép người dùng quản lý nhiều trung tâm dữ liệu từ một giao diện duy nhất. Chính kiến

trúc mở, khả năng tương thích với nhiều dự án khác theo dạng mô đun là nền tảng để các nhà mạng viễn thông và doanh nghiệp thiết kế được hệ thống NFV theo đúng nhu cầu của mình. Ngoài ra, Openstack còn có các ưu điểm như:

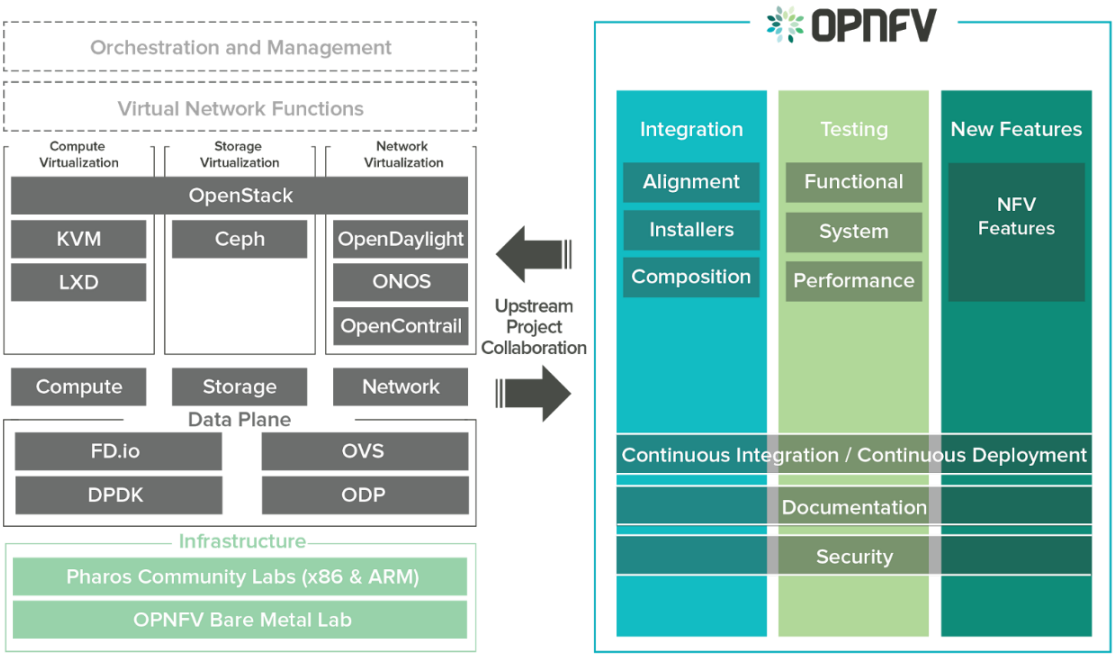
- Khả năng của OpenStack đã được chứng thực trên rất nhiều hệ thống điện toán đám mây lớn lớn trên thế giới như NASA, CERN, AT&T, China Mobile, SK Telecom, Ericsson,…
- Dự án Neutron (Networking) của OpenStack đã phát triển ổn định và được dùng trong nhiều hệ thống thực tế.
- Đa phần các giải pháp MANO có sẵn trên thị trường đều hỗ trợ OpenStack.

3.1.2. OPNFV

Vào tháng 9/2014, tổ chức Linux Foundation cho ra đời nền tảng nguồn mở Open Platform for NFV Project (OPNFV) nhằm tạo nên một nền tảng hợp nhất có thể đáp ứng được các tiêu chuẩn khắt khe của các nhà cung cấp dịch vụ, từ đó tăng tốc độ đưa các sản phẩm và dịch vụ mới liên quan đến NFV ra thị trường một cách nhanh chóng. OPNFV có mối quan hệ chặt chẽ với ETSI và nhiều tổ chức quốc tế khác nhằm thiết lập nên các tiêu chuẩn mở một cách thống nhất.

Xét theo mô hình kiến trúc của ETSI, OPNFV bao gồm khối hạ tầng ảo hóa NFVI và trình quản lý hạ tầng ảo hóa VIM trong khối MANO. Cụ thể, OPNFV tích hợp nhiều dự án lại với nhau như: OpenStack, KVM, LXD, Ceph, OpenDayLight,… và bao gồm cả một số giải pháp về tối ưu tác vụ chuyển tiếp gói tin như: FD.io, DPDK và ODP để có thể tạo ra một nền tảng hạ tầng ảo hóa hoạt động được trên các kiến trúc phần cứng phổ thông như: x86 của Intel, ARM,...

OPNFV có khả năng tương thích với với nhiều giải pháp điều phối và quản lý MANO khác nhau như: Tacker, OpenBaton, OSM,...



Hình 3-1 Những mục tiêu chính của dự án OPNFV

Những mục tiêu chính của dự án OPNFV là:

- Tích hợp, kiểm thử và phát triển một nền tảng nguồn mở để từ đó có thể phát triển các tính năng của NFV, tăng tốc độ phát triển của các sản phẩm mới.
- Tham gia đóng góp và phát triển các dự án nguồn mở được tích hợp trong OPNFV, đảm bảo tính đồng nhất, tương thích và hiệu năng tổng thể của cả nền tảng.
- Thiết lập một hệ sinh thái cho các giải pháp NFV dựa trên các tiêu chuẩn và phần mềm nguồn mở.
- Thúc đẩy OPNFV thành 1 nền tảng NFV được nhiều người sử dụng.

Bằng cách sử dụng OPNFV, tất cả các thành phần liên quan đã được đóng gói trong một tập tin ISO duy nhất để triển khai một môi trường NFV. Các công việc cấu hình đã được tự động hóa bằng Ansible và Puppet, cho phép các kỹ sư giảm thiểu thời gian thiết lập và cấu hình hệ thống mà dành nhiều thời gian hơn vào việc kiểm tra, thử nghiệm các giải pháp NFV.

Tuy nhiên, sau khi khảo sát và thử nghiệm, nhóm nhận thấy yêu cầu về phần cứng của OPNFV là khá cao và chỉ phù hợp với các công ty hoặc phòng nghiên cứu lớn.

Phần cứng	Yêu cầu
Số node	Tối thiểu 3 node gồm: • 1 Fuel deployment master • 1 Controller • 1 Compute
CPU	Mỗi Node tối thiểu 1 CPU hỗ trợ ảo hóa (từ Intel Xeon E3 trở lên)
RAM	Mỗi Node tối thiểu 16GB
Disk	Mỗi Node tối thiểu 256GB 10kRPM HDD
Network	Mỗi Node tối thiểu: • 1 card mạng dùng để PXE Boot từ Fuel Master • 1 card mạng (cấu hình VLAN): PUBLIC, MGMT, STORAGE, PRIVATE

Bảng 3-1 Cấu hình tối thiểu để triển khai một hệ thống OPNFV

Vậy nên, nhóm quyết định sẽ không sử dụng OPNFV mà sẽ tự xây dựng hạ tầng ảo hóa NFV riêng dựa trên nền tảng chính là OpenStack.

3.2. Khối điều phối và quản lý – NFV MANO

3.2.1. OpenBaton

3.2.1.1. Tổng quan

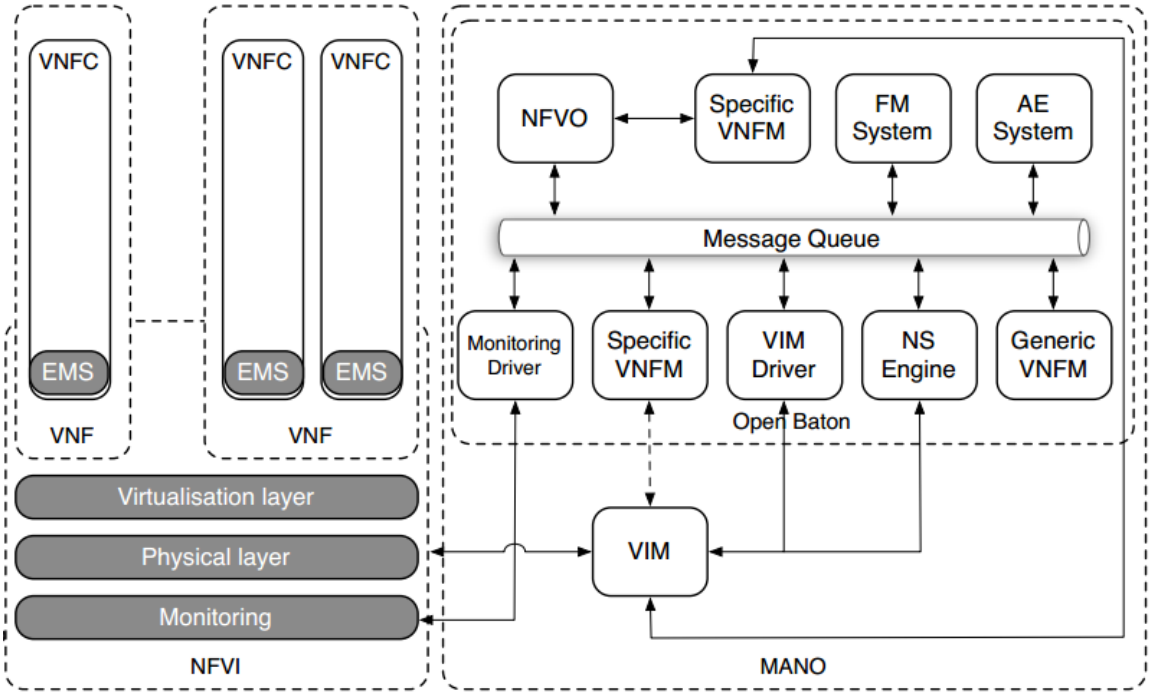
OpenBaton là một dự án mã nguồn mở về khối MANO được phát triển dựa theo các chuẩn đặc tả của ETSI. OpenBaton được phát triển bởi Viện nghiên cứu Fraunhofer Fokus (Đức). OpenBaton là lựa chọn khá phù hợp để tìm hiểu các khái niệm cơ bản về một hệ thống MANO vì các ưu điểm như: dễ dàng cấu hình, triển khai, thêm bớt các thành phần (có sẵn hoặc tự phát triển) theo dạng plugin cũng như cung cấp công

cụ quản lý trực quan, rõ ràng và thân thiện với người dùng. Bên cạnh đó, các tài liệu đặc tả, hướng dẫn của OpenBaton cũng rất cụ thể và dễ nắm bắt.

Tuy nhiên, do tuổi đời vẫn còn khá non trẻ, dự án vẫn còn nhiều tính năng chưa hoàn thiện. Một trong số đó là dự án hỗ trợ khởi tạo một VNF Forwarding Graph hoàn chình. Bên cạnh đó, do qui mô cộng đồng và nhóm phát triển tương đối nhỏ (do một trường đại học tại Đức chủ trì) nên độ ổn định và khả năng hỗ trợ người dùng còn hạn chế.

3.2.1.2. Kiến trúc

OpenBaton được phát triển dựa theo tài liệu đặc tả khối MANO của ETSI. Hình bên dưới biểu thị vị trí của OpenBaton ứng với các thành phần trong mô hình do ETSI đề xuất.



Hình 3-2 Kiến trúc của OpenBaton

OpenBaton cung cấp:

- **Một trình điều phối NFVO** để điều phối các hoạt động của hệ thống bao gồm: quản lý thông tin các VNF/NS, tương tác với một hoặc nhiều trình quản lý VNFM và VIM khác nhau.
- **Một trình quản lý VNF phổ thông** (generic VNFM) được OpenBaton cung cấp sẵn. Generic VNFM có thể được mở rộng để hỗ trợ nhiều loại VNF khác nhau. Đi kèm với generic VNFM này là generic EMS có chức năng thực thi các đoạn script cấu hình bên trong các máy ảo để chạy các VNF.
- **Trình điều khiển tương tác với VIM**: OpenStack, OpenVim,... Hiện OpenBaton chỉ mới hỗ trợ cho việc tương tác với giải pháp OpenStack và OpenVIM.
- Hệ thống kiểm soát lỗi và tự động mở rộng: xử lý các tác vụ quản trị VNF cũng như kiểm soát lỗi trong quá trình hoạt động.
- Một giao diện quản lý trên web để dễ dàng quản trị các VNFs và hệ thống giám sát dựa trên Zabbix.

3.2.1.3. Nhận xét

Qua khảo sát sơ bộ, nhóm nhận thấy tài liệu hướng dẫn của OpenBaton là khá rõ ràng cũng như yêu cầu phần cứng của OpenBaton là không cao.

Đồng thời, OpenBaton cũng hỗ trợ VIM là OpenStack phù hợp với hạ tầng mà nhóm sẽ triển khai. Vậy nên, nhóm sẽ tiến hành triển khai thử nghiệm OpenBaton ở chương tiếp theo.

3.2.2. OpenSource MANO

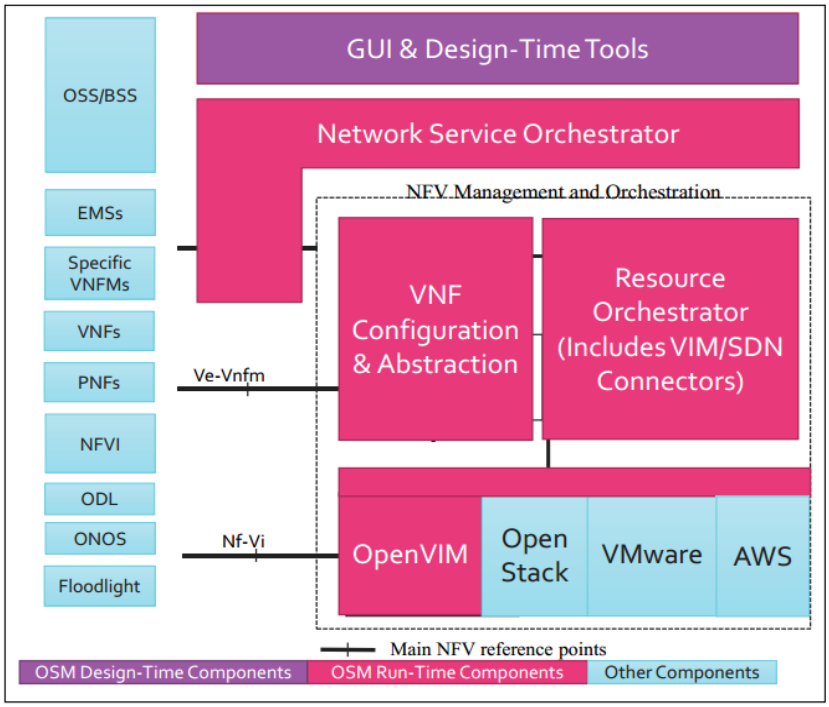
3.2.2.1. Tổng quan

Open Source Mano (OSM) là một dự án mã nguồn mở được chính tổ chức ETSI đỡ đầu nhằm thúc đẩy tạo ra các sản phẩm MANO chất lượng cao dành cho công nghệ NFV, tương thích với nhiều loại sản phẩm khác nhau, dễ dàng tiếp cận và phù hợp với mọi loại VNF.

Hình 3-3 Logo của OpenSource MANO

Về mặt chức năng, OSM cũng cung cấp một trình điều phối NFVO và một trình quản lý VNF phổ thông. Tuy nhiên, do mới chỉ phát hành tới bản OSM 2.0, các tài liệu hướng dẫn của dự án vẫn còn chưa thật sự đầy đủ.

3.2.2.2. Kiến trúc



Hình 3-4 Kiến trúc của OSM

Về mặt kiến trúc, OSM bao hàm toàn bộ khối MANO, bao gồm:

- **NFVO** bao gồm:
    - Trình điều phối Network Service: điều phối các hoạt động của hệ thống bao gồm: quản lý các NS/VNF, tương tác với các trình quản lý VNF khác nhau

- Trình điều phối tài nguyên: quản lý tài nguyên ảo hóa của hệ thống, có khả năng kết nối với các nhiều loại VIM (OpenVIM, OpenStack, VMware, AWS) cũng như các SDN Controller khác nhau (OpenDayLight, ONOS, Floodlight).

- **VNFM**: Giống như các giải pháp MANO khác, OSM cũng cung cấp một trình quản lý VNF phổ thông và hỗ trợ người dùng kết nối tới các VNFM đặc biệt riêng cho các VNF đặc biệt của người dùng.
- **VIM:** OSM cung cấp sẵn trình VIM do chính OSM tự phát triển là OpenVIM. Bên cạnh đó, OSM có thể dễ dàng tương thích với các trình VIM khác như OpenStack, VMware và đặc biệt là nền tảng đám mây công cộng Amazon Web Service. Việc hỗ trợ nhiều giải pháp điện toán đám mây sẽ mở đường cho sự phát triển của những giải pháp NFV trên nền tảng Hybrid Cloud. Đáp ứng được nhiều ơn nhu cầu của người sử dụng.

Ngoài ra, OSM còn cung cấp những công cụ để người dùng tự phát triển các tính năng của riêng mình cũng như một giao diện GUI để người điều khiển giám sát hoạt động của hệ thống.

3.2.2.3. Nhận xét

Tiềm năng của dự án OSM trong tương lai là rất lớn. Với kiến trúc mở hướng tới việc sẽ hỗ trợ nhiều loại hình VIM và SDN Controller khác nhau, OSM sẽ dễ dàng phù hợp với nhiều mô hình của khách hàng. Có thể nói đây là một dự án đầy triển vọng, nhưng vẫn cần thêm thời gian để hoàn thiện các tính năng của mình.

Tuy nhiên, thông qua khảo sát sơ bộ, nhóm nhận thấy việc yêu cầu phần cứng của OSM vẫn còn khá cao. Theo trang Wiki của OSM, cấu hình đề nghị của giải pháp MANO này là:

Phần cứng	Yêu cầu
Số node	Tối thiểu 1 node
CPU	Mỗi Node tối thiểu 4 CPU hỗ trợ ảo hóa
RAM	Mỗi Node tối thiểu 16GB
Disk	Mỗi Node tối thiểu 40GB HDD
Network	Mỗi Node tối thiểu 1 card mạng (có kết nối Internet)

Bảng 3-2 Cấu hình tối thiểu để triển khai một node OSM

Đây chỉ mới là cấu hình dành riêng cho OSM mà chưa tính đến hạ tầng ảo hóa (vốn phải chiếm rất nhiều tài nguyên để cài đặt và triển khai dịch vụ). Chính vì vậy, nhóm sẽ không sử dụng giải pháp này để tiến hành triển khai thử nghiệm.

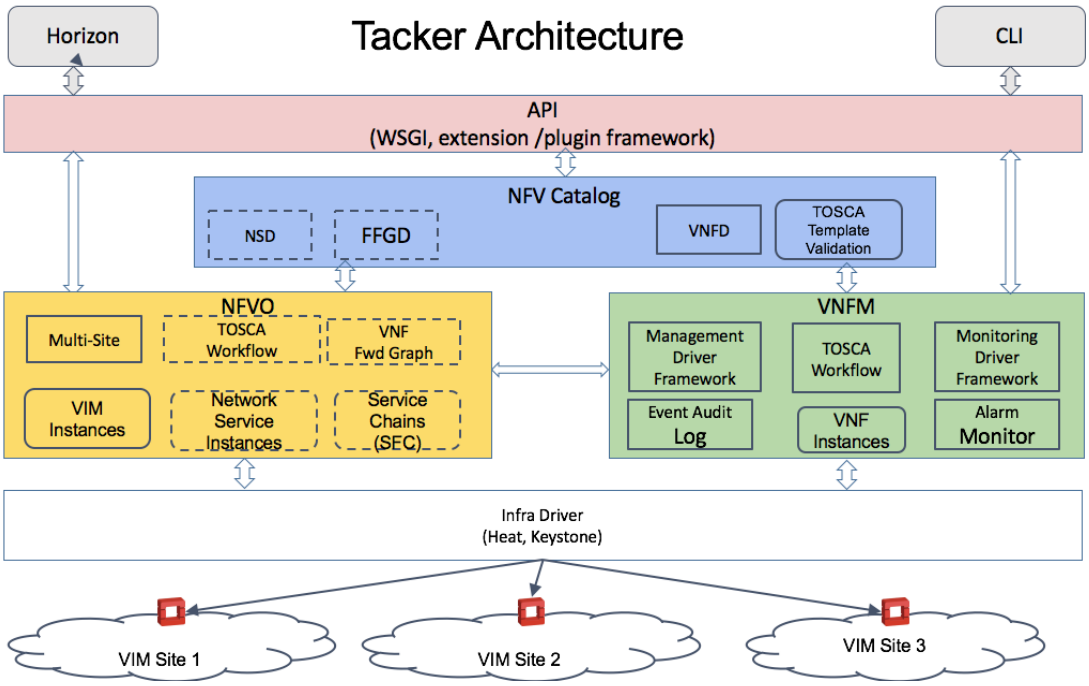
3.2.3. Tacker

3.2.3.1. Giới thiệu

Tacker là dự án chính thức của OpenStack về khối điều phối và quản lý MANO trong kiến trúc NFV. Về mặt kiến trúc, Tacker tuân theo mẫu thiết kế kiến trúc khối MANO của ETSI. Về mặt chức năng, Tacker cung cấp một trình điều phối NFVO và một trình quản lý phổ thông các VNF.

Trong bản phát hành Ocata mới nhất của OpenStack, Tacker đã có thể khả năng triển khai và điều phối dịch vụ mạng đầu cuối hoàn chỉnh. Tuy vậy, dự án Tacker cũng vẫn còn khá non trẻ với thời gian phát triển không dài cùng với phiên bản hiện tại mới chỉ là Tacker 0.7

3.2.3.2. *Kiến trúc*

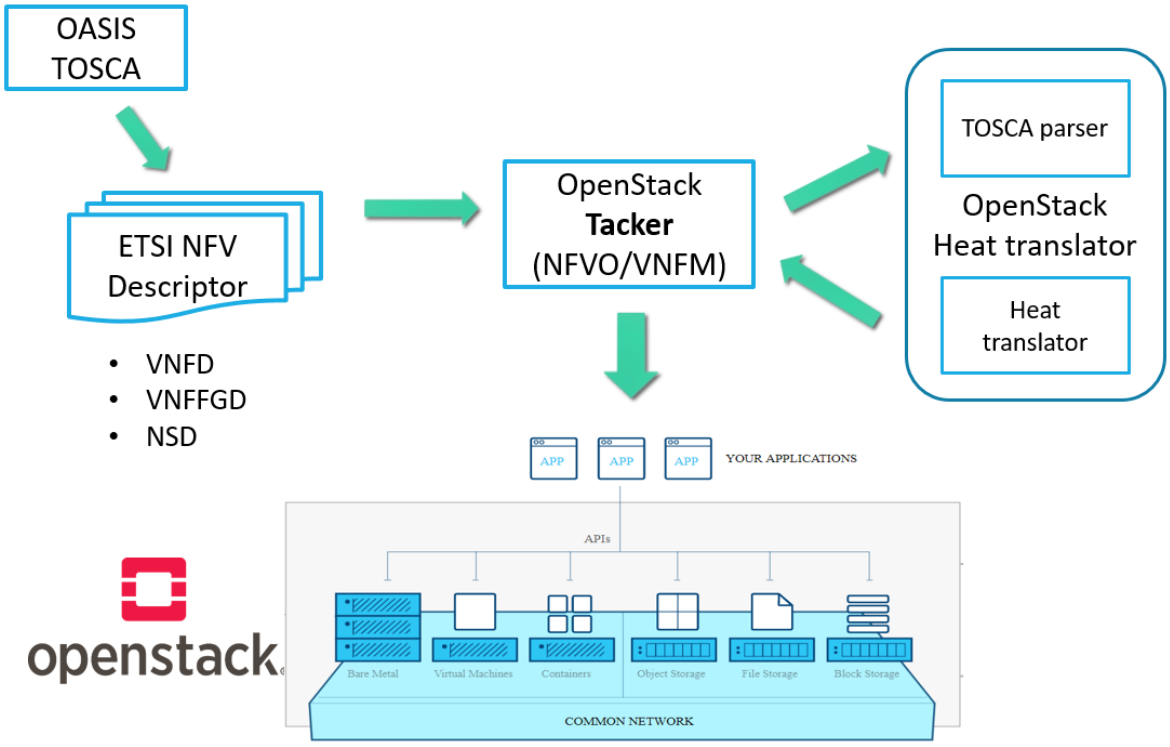


Hình 3-5 Kiến trúc của Tacker

Tacker bao gồm các thành phần chính:

- **NFVO:** điều phối hoạt động hệ thống NFV. Bao gồm quản lý các VIM Instance, Network Service Instances, VNF Fowarding Graph,...
- **VNFM:** quản lý vòng đời (khởi tạo, cập nhật, xóa), cấu hình hoạt động của các VNF Instance, cũng như tự động sửa chữa, co dãn các VNF.
- **NFV Catalog:** chứa các tập tin mô tả để khởi tạo, triển khai các VNF, Network Service và VNF Forwarding Graph.
- **Các API:** để kết nối với dashboard (horizon) và cli của người quản trị.

Tacker thông qua trình điều khiển hạ tầng như Heat và Keystone để kết nối đến các hạ tầng ảo hóa ở bên dưới. Các hạ tầng này được các dự án con khác của OpenStack cung cấp như Nova, Neutron,...



Hình 3-6 Quy trình làm việc của Tacker

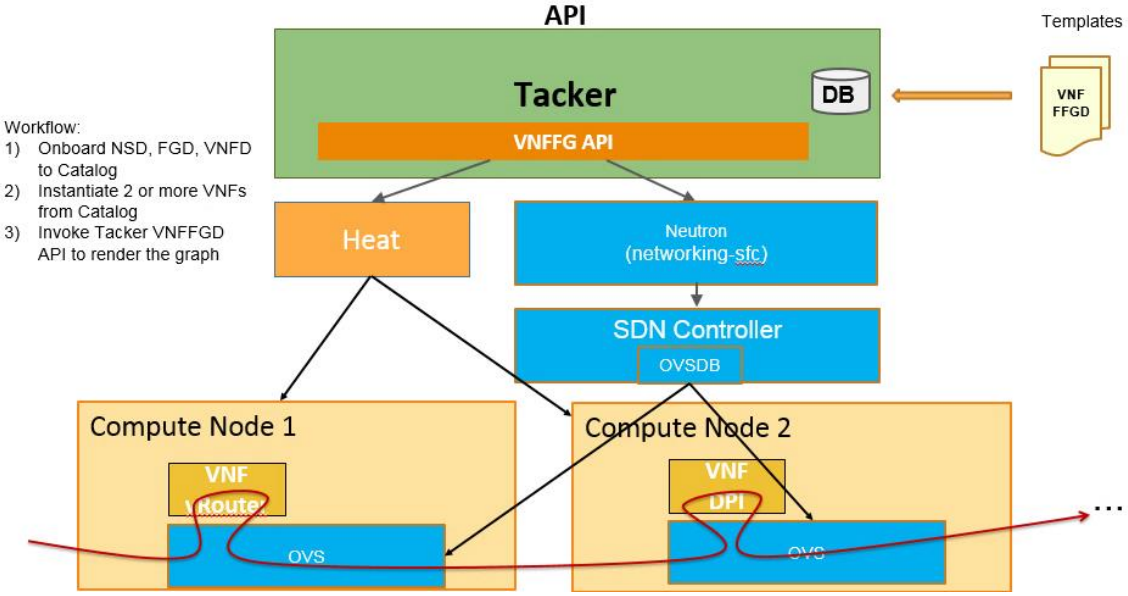
Quá trình khởi chạy một VNF của Tacker tuân thủ qui trình của một MANO điển hình:

- Từ trình điều khiển, người quản trị chọn từ ra VNFD cần thiết từ kho chứa VNFD (VNFD Catalog).
- NFVO sẽ đọc tập tin đặc tả VNFD và chuyển sang cho cho dự án Heat.
- Heat chuyển các đặc tả thành các yêu cầu cấp phát tài nguyên ảo hóa thích hợp để tạo thành các máy ảo.
- Management driver của Tacker giám sát quá trình khởi tạo của VNF.
- SFC driver của Tacker điều khiển việc khởi tạo VNFFG giữa các VNF.
    - SFC Driver sẽ gọi lại Neutron-SFC để thực hiện việc nối các máy ảo lại thành một chuỗi các máy ảo (Port Chain) để xử lý các luồng dữ liệu phù hợp với các luật (Flow Classifier) được định nghĩa sẵn trong VNFFGD.

- Monitor driver của Tacker sẽ giám sát các VNF vừa được khởi tạo. (Có thể kết hợp với ceilometer)
- Khi sự cố/nhu cầu mở rộng xảy ra, monitor driver sẽ nắm được thông tin và báo lên lại cho infra driver để sử xý (scale up, respawn).

Hiện tại, Tacker mới chỉ có thể tương tác với VIM là chính OpenStack. Đồng thời, khả năng cài đặt Tacker độc lập khỏi hạ tầng ảo hóa OpenStack bên dưới để triển khai mô hình VIM đa vị trí địa lý (multi-site VIM) vẫn còn chưa hoàn toàn ổn định. Bên cạnh đó, khả năng tương thích với các dự án quản lý SDN bên ngoài vẫn còn chưa tốt.

Trong tương lai, thông qua một số plugin đang được phát triển, Tacker sẽ có thể tương tác tốt hơn với các trình quản lý SDN để tối ưu việc điều khiển luồng dữ liệu giữa các VNF/NS. Một trong những trình quản lý SDN đang được hỗ trợ là OpenDayLight.



Hình 3-7 Quy trình khởi tạo VNFFG bằng Tacker và SDN Controller

3.2.3.3. Nhận xét

Tacker là dự án MANO của chính OpenStack nên tính tương thích với hạ tầng ảo hóa của OpenStack là không cần phải bàn cãi. Tuy vẫn còn nhiều tính năng cần hoàn thiện

nhưng Tacker vẫn là một dự án rất hứa hẹn do nhận được sự quan tâm và góp sức phát triển từ cộng đồng vốn rất đông đảo của OpenStack.

Qua khảo sát sơ bộ, nhóm nhận thấy yêu cầu phần cứng của Tacker không cao cũng như các tài liệu hướng dẫn đi kèm khá rõ ràng. Bên cạnh đó, Tacker có thể được dễ dàng cài đặt vào một hạ tầng Openstack đã được triển khai sẵn. Vậy nên, nhóm quyết định sẽ sử dụng cả Tacker để triển khai thử nghiệm mô hình.

3.2.4. Tổng kết các giải pháp MANO

Tên giải pháp	Loại VIM được hỗ trợ	Ưu điểm	Khuyết điểm
OpenBaton 3.0	- OpenStack - Hỗ trợ VIM plugin tự viết	- Tài liệu rõ ràng. - Yêu cầu phần cứng thấp. - Hỗ trợ phát triển VNFM, NFVO riêng dựa trên bộ thư viện của OpenBaton.	- Thiếu ổn định, vẫn còn nhiều lỗi. - Chức năng hiện có còn ít
OpenSource MANO 2.0	- Amazon - VMware - OpenStack - OpenVIM	- Hỗ trợ kết hợp với AWS (public cloud) - Hỗ trợ nhiều loại VIM, SDN Controller	- Mới ở giai đoạn sơ khai, còn thiếu nhiều tính năng, cần thêm thời gian để phát triển. - Yêu cầu cấu hình phần cứng khá cao, hạn chế khả năng tiếp cận của các nhóm nghiên cứu.

Tacker 0.7.0 (Ocata)	- OpenStack	-Yêu cầu phần cứng chấp nhận được. -Tài liệu tương đối rõ ràng -Tương thích tốt với OpenStack. -Đã hỗ trợ các tập tin đặc tả VNFD/VNFFGD/NSD đầy đủ.	-Chỉ hỗ trợ OpenStack làm VIM -Việc kết nối giữa Tacker và các SDN Controller còn chưa tốt.

Bảng 3-3 Bảng so sánh các giải pháp MANO

Qua khảo sát, nhóm sẽ chọn ra hai giải pháp MANO là OpenBaton và Tacker để triển khai thử nghiệm mô hình. Còn đối với VIM và NFVI, nhóm sẽ chọn nền tảng Openstack.

Chương 4 Triển khai thử nghiệm

Trong chương ngày, nhóm sẽ lần lượt tiến hành triển khai thử nghiệm hệ thống NFV sử dụng giải pháp MANO là OpenBaton và Tacker trên nền hạ tầng ảo hóa NFVI của OpenStack. Sau khi triển khai và có những đánh giá sâu hơn, nhóm sẽ chọn một giải pháp MANO phù hợp để tiếp tục triển khai thử nghiệm các hàm chức năng mạng cũng như dịch vụ mạng.

4.1. Triển khai thử nghiệm hệ thống NFV

4.1.1. Thử nghiệm giải pháp OpenBaton

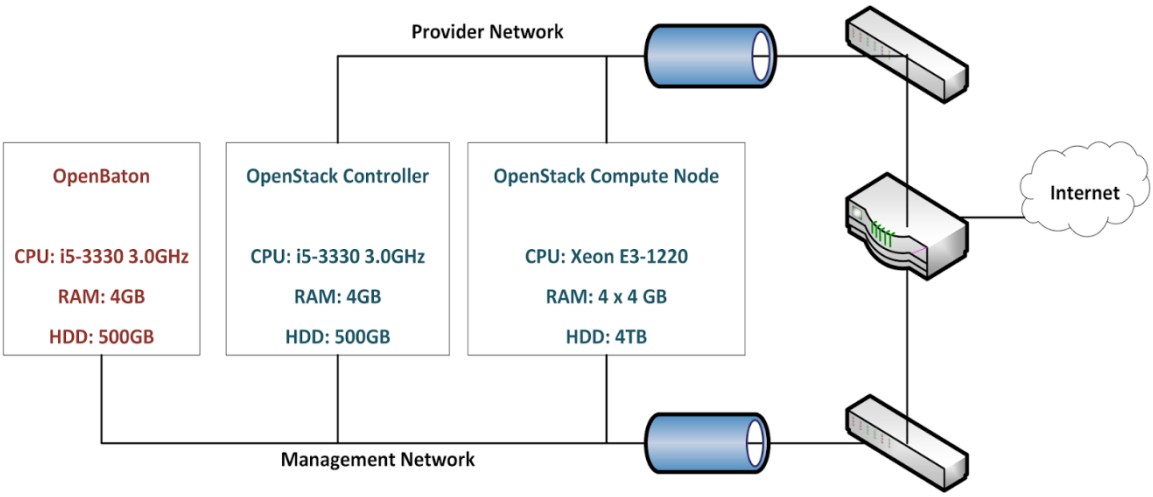
4.1.1.1. Mô hình

Tính đến Release 3.0 của mình, OpenBaton mới chỉ hỗ trợ một trình VIM duy nhất là Openstack nên nhóm thực hiện triển khai thử nghiệm mô hình OpenBaton và OpenStack như sau:

- OpenStack phiên bản Ocata (2 nodes)
    - Controller
        - Compute Server
        - Network Server
        - Glance Server
        - Heat Server
        - Ceilometer Server
    - Compute
        - OpenvSwitch Agent
        - Nova compute agent
        - Neutron-OVS Agent
- OpenBaton phiên bản 3.2.0 (1 node)

Hai node controller và compute của OpenStack sẽ quản lý tài nguyên ảo hóa (CPU, RAM, network) và cung cấp những tài nguyên này cho OpenBaton sử dụng. OpenBaton sẽ khởi tạo các VNF, NS dựa trên nền hạ tầng ảo hóa của OpenStack.

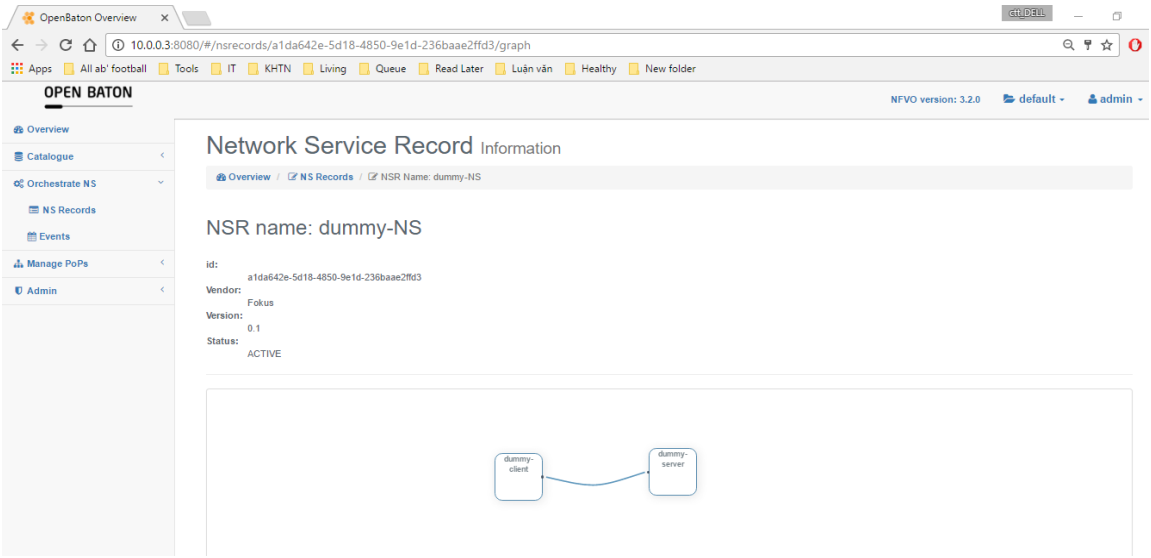
Mô hình sử dụng mạng self-service layer 3. Chi tiết như sau:



Hình 4-1 Mô hình thử nghiệm OpenBaton

4.1.1.2. Kết quả

Nhóm đã triển khai thành công mô hình OpenStack kết hợp OpenBaton và khởi tạo được các network service demo.



Hình 4-2 Network service demo trong OpenBaton

4.1.1.3. Nhận xét OpenBaton

Ưu điểm:

- Dễ dàng cài đặt mô hình thử nghiệm.
- Yêu cầu phần cứng đơn giản.
- Tài liệu hướng dẫn rõ ràng.

Khuyết điểm:

- Độ ổn định chưa cao.
- Chưa hỗ trợ tập tin đặc tả mạng VNF FGD.
- Khả năng phát triển NS/VNF riêng còn tương đối khó khăn

4.1.2. Thử nghiệm giải pháp Tacker

4.1.2.1. Mô hình

Dự án Tacker cũng mới chỉ hỗ trợ một trình VIM duy nhất là chính OpenStack. Mô hình thử nghiệm của nhóm sẽ sử dụng phiên bản OpenStack Ocata như sau:

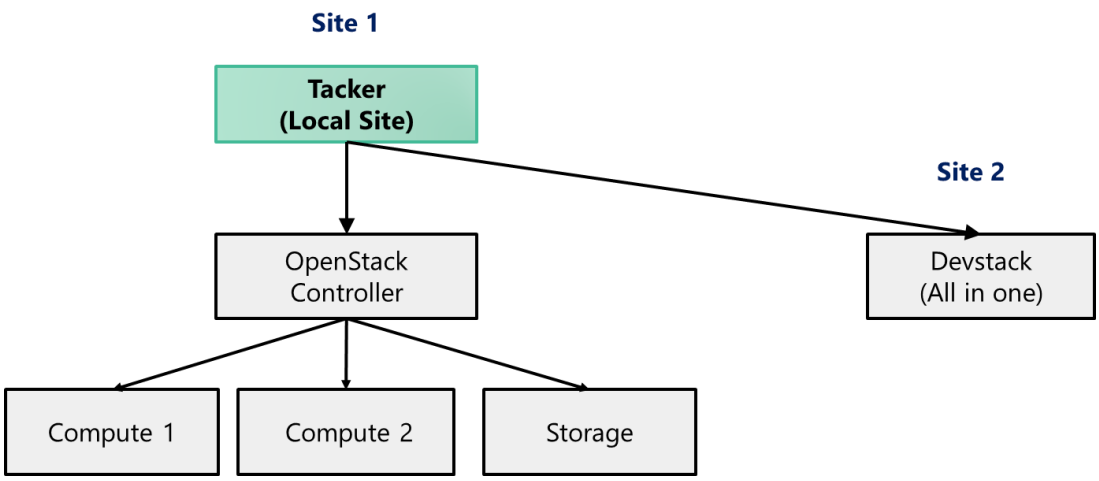
- Site 1: OpenStack phiên bản Ocata (4 nodes)
    - 1 Controller Node
        - Compute Server
        - Network Server (networking-sfc)
        - Glance Server
        - Tacker Server
        - Heat Server
        - Ceilometer Server
        - Mistral Server
        - Cinder
    - 2 Compute Node
        - OpenvSwitch Agent
        - Nova compute Agent
        - Neutron-OVS Agent

    - 1 Storage Node
        - Cinder-Volume
- Site 2: Devstack All-in-one (1 node).
    - Compute Server và Nova compute Agent
    - Network Server(networking-sfc, OpenvSwitch Agent, Neutron-OVS Agent)
    - Glance Server
    - Ceilometer Server
    - Heat Server

Ở site 1, Controller node lúc này sẽ được cài đặt thêm dịch vụ là Tacker (VNFM, NFVO). Như vậy, controller node lúc này sẽ trở thành khối MANO gồm đầy đủ các thành phần. Đồng thời, một node Compute và một Node Storage được cài đặt thêm để tăng cường khả năng tính toán và lưu trữ của hệ thống.

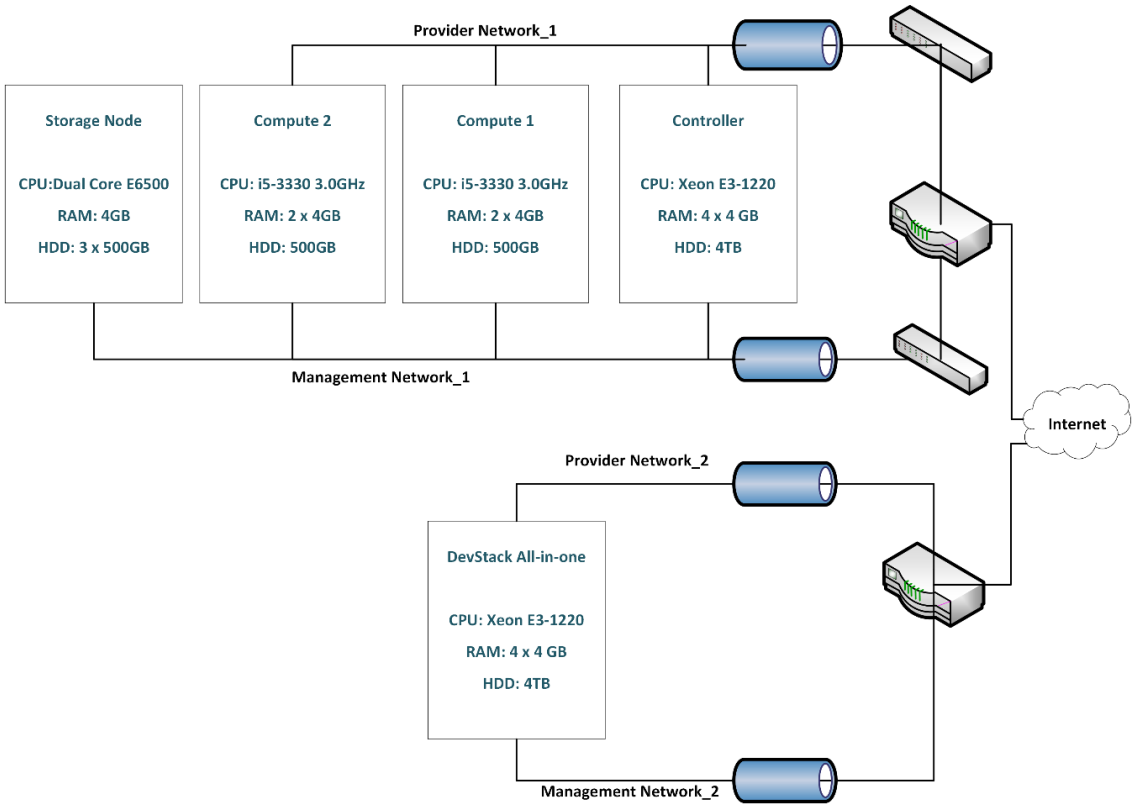
Ở site 2, một hệ thống DevStack All-in-one được triển khai trên 1 node. Tuy là rút gọn, nhưng hệ thống Devstack vẫn có đầy đủ các dịch vụ cần thiết như: Nova, Neutron, Glance, Heat, Ceilometer,… như hạ tầng ở site 1.

Hạ tầng ảo hóa ở cả 2 site đều được điều khiển bởi Tacker ở trên máy Controller của site 1.



Hình 4-3 Kịch bản thử nghiệm Tacker

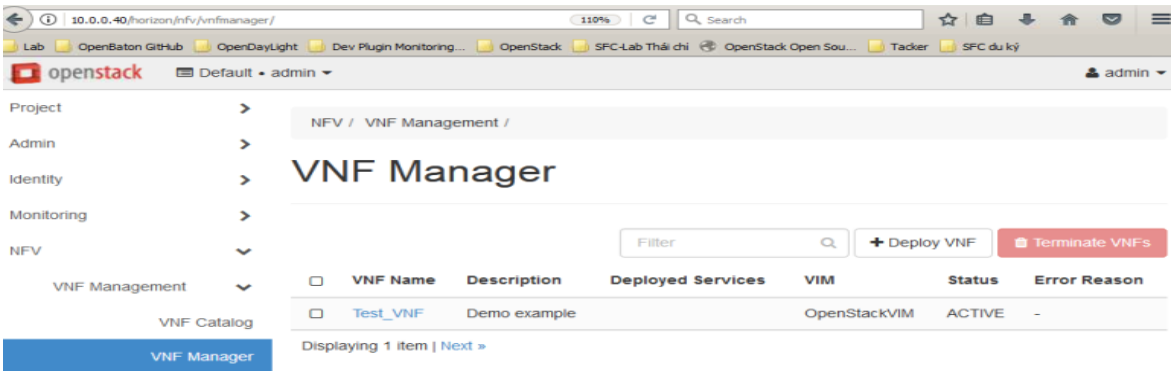
Mô hình sử dụng mạng self-service layer 3. Chi tiết như sau:



Hình 4-4 Mô hình thử nghiệm Tacker

4.1.2.2. Kết quả

Nhóm đã triển khai thành công mô hình Openstack sử dụng Tacker làm MANO và triển khai thử nghiệm các VNF demo.



Hình 4-5 Một VNF demo trong Tacker

4.1.2.3. Nhận xét

Ưu điểm:

- Tài liệu hướng dẫn của Tacker khá chi tiết.
- Yêu cầu phần cứng chấp nhận được.
- Đã hỗ trợ đầy đủ các tập tin đặc tả bao gồm VNFFGD và NSD (riêng NSD chỉ mới có ở phiên bản mới nhất là OpenStack Ocata).

Khuyết điểm:

- Độ ổn định chưa cao.
- Khả năng cài đặt độc lập Tacker thành Node MANO riêng còn chưa ổn định.
- Vẫn còn xuất hiện nhiều lỗi trong lúc vận hành.

4.1.3. So sánh và đánh giá hai giải pháp thử nghiệm

Qua thử nghiệm, nhóm nhận thấy phương án thử nghiệm OpenBaton tuy tương đối đơn giản và dễ cài đặt, nhưng lượng tính năng hỗ trợ cũng như độ ổn định và khả năng phát triển tùy biến VNF/VNFFG/NS chưa cao.

Trong khi đó, phương án Tacker tuy cũng vẫn còn non trẻ nhưng lại tương thích tốt với hạ tầng OpenStack sẵn có. Đồng thời, khả năng mở rộng của Tacker là rất cao thông qua các API nguồn mở cũng như đã tương đối hỗ trợ phát triển tùy biến các VNF/VNFFG/NS. Vậy nên, nhóm quyết định sẽ sử dụng Tacker để tiếp tục triển khai dịch vụ mạng thử nghiệm.

4.2. Triển khai thử nghiệm VNF

4.2.1 Nhu cầu

Sau khi đã xây dựng thành công hệ thống NFV sử dụng MANO là Tacker kết hợp với hạ tầng ảo hóa của OpenStack, nhóm xin đề xuất một số trường hợp thực tế có thể áp dụng được đối với thực trạng tại Việt Nam như sau:

Trường hợp 1: Khó khăn trong việc triển khai các dịch vụ mạng phức tạp như IMS

IMS là thuật ngữ viết tắt cho IP Multimedia Subsystem. Đây là một kiến trúc mạng nhằm tạo sự sự thuận tiện khi kết nối mọi người với nhau nhờ hỗ trợ nhiều phương thức truy cập vô tuyến cũng như hữu tuyến sử dụng giao thức IP. Nền tảng IMS hỗ trợ được rất nhiều dịch vụ viễn thông như: gọi thoại, fax, email, truy cập Internet, dịch vụ Web,...

Tuy nhiên việc triển khai hệ thống IMS là khá phức tạp do những yêu cầu về phần cứng chuyên biệt, kiến trúc đặc thù của hệ thống này. Ngoài ra là các hệ thống đi kèm như các giải pháp về quản lý cước, giám sát hệ thống. Việc này làm giảm tốc độ triển khai dịch vụ mới của các nhà cung cấp dịch vụ mạng viễn thông đến khách hàng. Bên cạnh đó, các doanh nghiệp không chuyên về mạng viễn thông hay các đơn vị sự nghiệp giáo dục như trường đại học muốn xây dựng hệ thống nhỏ để sử dụng nội bộ là khá khó khăn.

Trường hợp 2: Vấn đề quản trị hạ tầng mạng của các đơn vị nhỏ và vừa

Các đơn vị mà nhóm muốn nhắc đến là các doanh nghiệp không chuyên về mạng viễn thông hay các đơn vị hành chính, giáo dục như trường Đại học Khoa học Tự nhiên,... Một trong những vấn đề của các đơn vị này là việc phải thường xuyên đầu tư mua sắm, thay thế các thiết bị mạng được sử dụng. Giá thành các thiết bị mạng này là tương đối đắt đỏ. Và khi các thiết bị này khi hết hạn giấy phép sử dụng hoặc hỏng hóc thì thường phải mua sắm mới rất tốn kém.

Mặt khác, khả năng linh hoạt tùy biến cũng như co dãn qui mô dịch vụ là có giới hạn. Trong những khoảng thời gian cao điểm trong ngày, các thiết bị này có thể gặp tình trạng quá tải dẫn đến gián đoạn dịch vụ. Còn trong các khoảng thời gian thấp điểm như ngoài giờ hành chính, các thiết bị này lại gây lãng phí điện năng. Bên cạnh đó, bất kì sự thay đổi nào của các thiết bị như: thêm, bớt, gặp sự cố hỏng hóc,... đều có khả năng làm ảnh hưởng đến toàn bộ dịch vụ. Người quản trị buộc phải cho dừng lại để xử lý thiết bị, đấu nối lại dây dẫn cho phù hợp rất mất thời gian. Điều này sẽ ảnh

hưởng đến người sử dụng cuối hoặc tệ hơn là với khách hàng của doanh nghiệp (trong trường hợp các doanh nghiệp hoạt động chủ yếu trên môi trường Internet).

4.2.2 Đề xuất

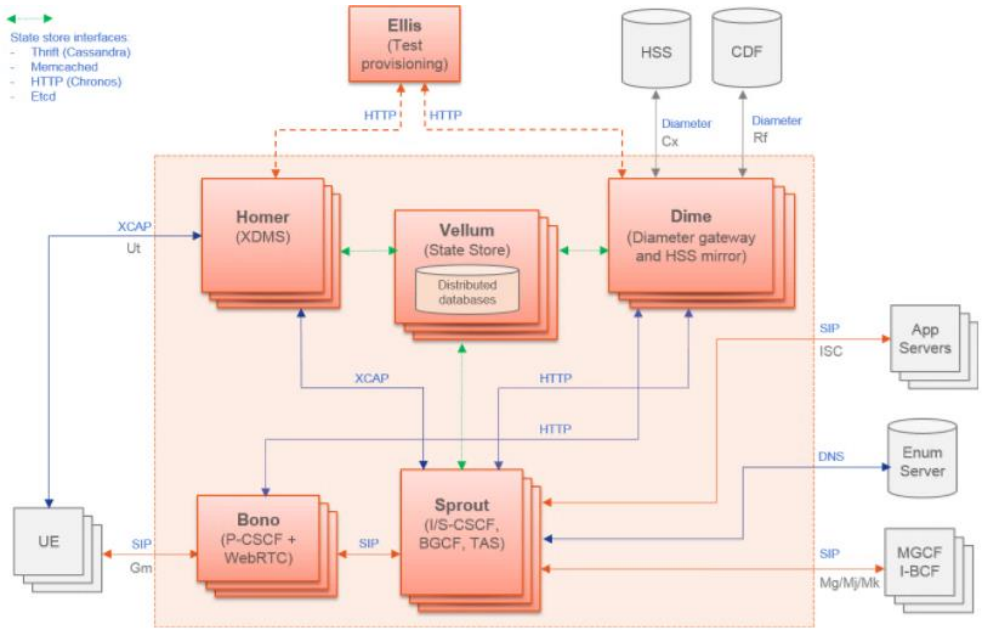
Ứng với từng trường hợp, nhóm xin đề xuất những hướng giải quyết như sau:

Trường hợp 1:

Để tăng tốc cho việc triển khai các dịch vụ mạng phức tạp này, nhóm đề xuất cấu hình sẵn dịch vụ vIMS mẫu rồi đóng gói thành các tập tin đặc tả mẫu. Khi cần triển khai dịch vụ, người quản trị chỉ việc sử dụng tập tin đặc tả VNFD tương ứng là có thể nhanh chóng triển khai dịch vụ.

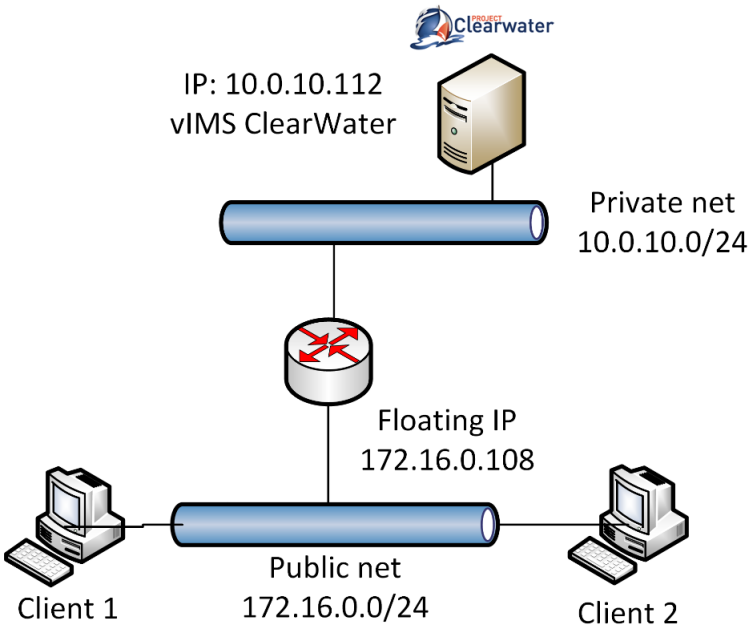
Hệ thống vIMS cũng hỗ trợ các chức năng như gọi thoại, gọi video,... như một hệ thống IMS tiêu chuẩn thông thường. Cách làm này sẽ đơn giản và dễ dàng hơn cho người quản trị trong việc triển khai cũng như khâu quản lý.

Để demo cho trường hợp này, nhóm sẽ sử dụng giải pháp IMS Clearwater. Kiến trúc của Clearwater cũng là khá phức tạp. Tuy nhiên do đã đóng gói thành template mẫu nên việc triển khai đã trở nên đơn giản rất nhiều.



Hình 4-6 Kiến trúc của IMS ClearWater

Đồ hình mạng của demo này như sau:



Hình 4-7 Đồ hình mạng Demo vIMS

Kịch bản:

- Một máy ảo IMS ClearWater:
  - IP Private: 10.0.10.112
  - Floating IP: 172.16.0.108 được cấu hình tự động.
- Hai client nằm trên public net và được cài đặt phần mềm VoIP client là Jitsi. Sau khi server IMS được triển khai thành công, hai client sẽ truy cập vào server và sử dụng các dịch vụ do server cung cấp.

Trường hợp 2:

Để tối ưu việc quản trị hạ tầng mạng của các đơn vị này, nhóm đề xuất

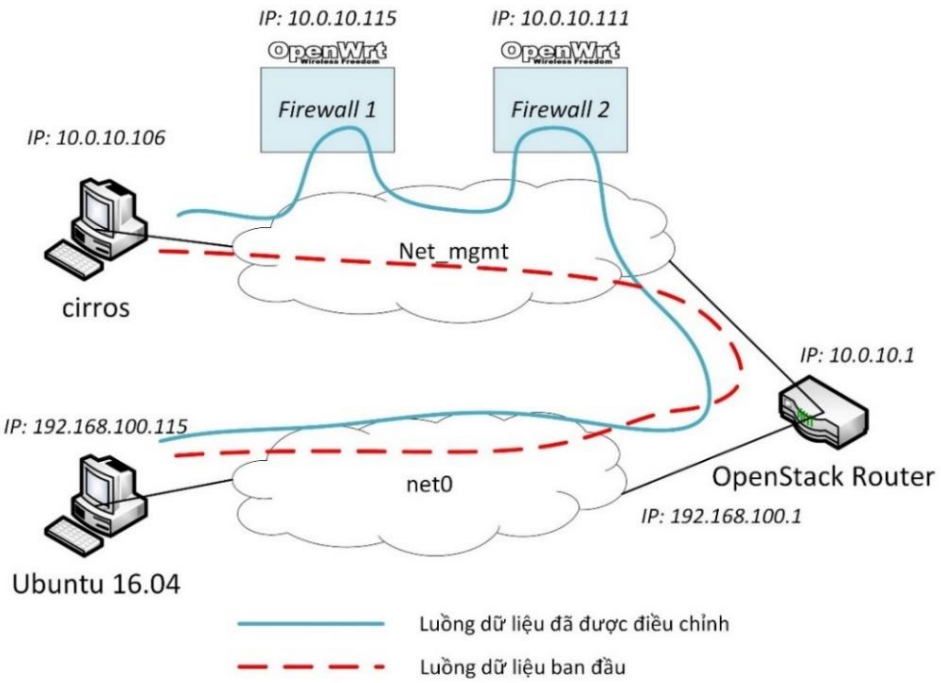
- Sử dụng các thiết bị phần cứng phổ thông cũ (máy chủ tính toán, lưu trữ, thiết bị chuyển mạch) để xây dựng một hệ thống NFV hoàn chỉnh. Việc này sẽ giúp tiết kiệm được chi phí đầu tư phần cứng cũng như chủ động được về giấy phép phần mềm. Bên cạnh đó, hạ tầng ảo hóa nhóm đã sử dụng hoàn toàn có thể

được tận dụng để xây dựng thành một hệ thống tính toán đám mây phục vụ thêm nhiều lợi ích khác nữa của đơn vị như : quản lý, lưu trữ tập trung,...

- Ứng dụng usecase ***Đồ thị Chuyển tiếp luồng dữ liệu giữa Các hàm chức năng mạng ảo hóa (VNF Forwarding Graph)*** để linh hoạt hơn trong việc điều chỉnh luồng dữ liệu các dịch vụ mạng. Người quản trị có thể linh hoạt: thêm, bớt hay điều chỉnh luồng dữ liệu theo các chính sách thích hợp để tạo ra một tập hợp những dịch vụ mạng phù hợp một cách nhanh chóng và giảm thiểu tối đa thời gian gián đoạn của dịch vụ mạng. Ứng dụng use case này sẽ giúp người quản trị dễ dàng ứng phó khi có sự cố xảy ra: DDoS, thiết bị nhiễm mã độc, hư hỏng phần cứng…

Để demo cho use case này, nhóm sẽ sử dụng các VNF chạy phần mềm OpenWRT. Đây là một hệ điều hành mã nguồn mở, khả năng tùy biến cao, đóng vai trò là những router ảo trong hệ thống.

Những VNF này được tạo tự động bằng cách sử dụng các tập tin đặc tả VNFD. Các VNF này sau đó sẽ được xâu chuỗi lại thành một dịch vụ mạng phức tạp hơn thông qua việc ứng dụng VNF Forwarding Graph. Luồng dữ liệu của người dùng từ máy Source tới máy Dest sẽ bị buộc phải đi qua các VNF chạy OpenWRT và phải chịu những tập luật (rule) được cài đặt trên các VNF này. Đồ hình mạng của demo này sẽ như sau:



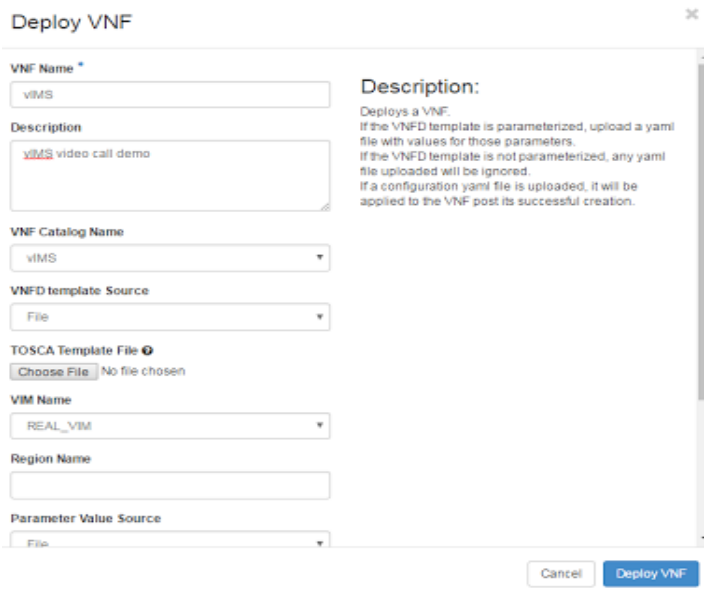
Hình 4-8 Luồng dữ liệu của demo VNFFG

- Máy Source: đóng vai trò client
  - IP: 10.0.10.106/24
  - OS: Cirros
  - Network: net_mgmt
- Máy Dest: đóng vai trò Server
  - IP: 192.168.100.115
  - OS: Ubuntu Cloud Image 16.04
  - Network: net0
- VNF1: đóng vai trò Firewall 1
  - IP: 10.0.10.115/24
  - OS: OpenWRT
  - Network: net_mgmt
- VNF2: đóng vai trò Firewall 2
  - IP: 10.0.10.111/24
  - OS: OpenWRT
  - Network: net_mgmt

4.2.3 Kết quả thử nghiệm

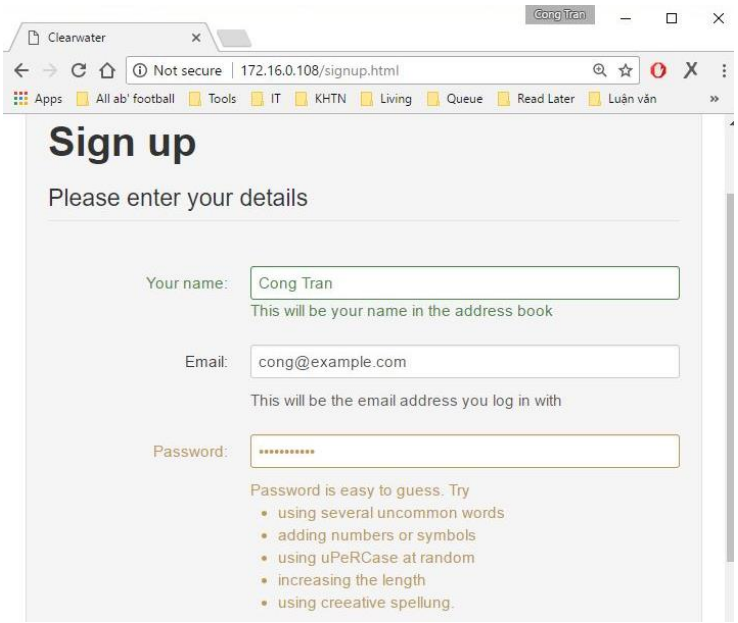
Trường hợp 1:

Nhóm đã triển khai nhanh thành công VNF vIMS Clearwater



Hình 4-9 Tạo VNF vIMS ClearWater từ dashboard

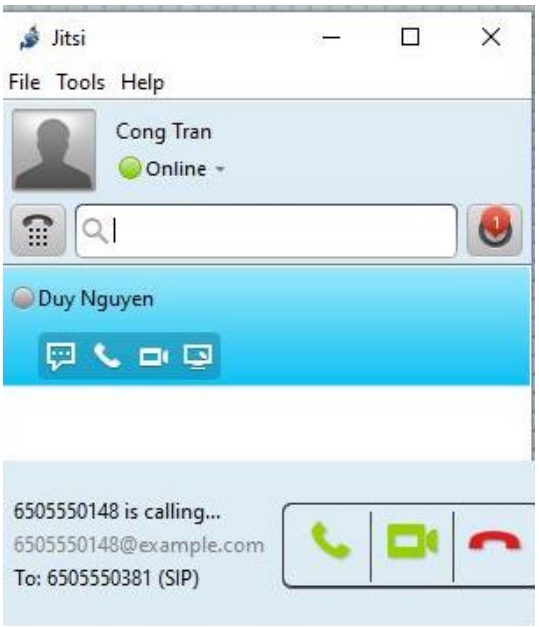
Sau khi khởi tạo, người dùng có thể truy cập và tạo tài khoản để sử dụng dịch vụ.



Hình 4-10 Người dùng đăng kí tài khoản tại tổng đài vIMS

Người dùng 1 có ID là 6505550148. Người dùng 2 có ID là 6505550381. Sau khi tạo tài khoản, người dùng có thể sử dụng các phần mềm VoiceOverIP để kết nối tới hệ thống. Ở demo này, nhóm sẽ sử dụng phần mềm Jitsi làm client.

Để kiểm tra, hai người dùng sẽ thực hiện cuộc gọi video với nhau.



Hình 4-11 Người dùng 1 thực hiện cuộc gọi tới người dùng 2

Hình 4-12 Hai người dùng thực hiện được cuộc gọi video với nhau

Trường hợp 2:

Nhóm đã triển khai tự động các VNF 1 và VNF2 bằng tập tin VNFD.

```
root@controller:~# tacker vnf-show vnf1
+----------------+-----------------------------------------------------------------------------------+
| Field          | Value                                                                             |
+----------------+-----------------------------------------------------------------------------------+
| attributes     | {"heat_template": "heat_template_version: 2013-05-23\ndescription:                |
|                | 'Demo example\n\n  '\nparameters: {}\nresources:\n  VDU1:\n    type:              |
|                | OS::Nova::Server\n    properties:\n      user_data_format:                           |
|                | SOFTWARE_CONFIG\n      availability_zone: nova\n      image: OpenWRT\n              |
|                | flavor:\n        get_resource: VDU1_flavor\n      networks:\n      -              |
|                | port:\n          get_resource: CP11\n      config_drive: false\n                  |
|                | CP11:\n    type: OS::Neutron::Port\n    properties:\n                              |
|                | port_security_enabled: false\n      network: net_mgmt\n  VDU1_flavor:\n           |
|                | type: OS::Nova::Flavor\n    properties:\n      vcpus: 1\n      disk:               |
|                | 1\n      ram: 512\noutputs:\n  mgmt_ip-VDU1:\n    value:\n                           |
|                | get_attr: [CP11, fixed_ips, 0, ip_address]\n"}                                    |
| created_at     | 2017-06-25 08:09:30                                                               |
| description    | Demo example                                                                      |
| error_reason   |                                                                                   |
| id             | f0b0e28a-69d4-41e5-9bca-655882df4c00                                              |
| instance_id    | 669a5d9c-2b5c-4220-8787-922894504831                                              |
| mgmt_url       | {"VDU1": "10.0.10.115"}                                                           |
| name           | VNF1                                                                              |
| placement_attr | {"vim_name": "REAL_VIM"}                                                          |
| status         | ACTIVE                                                                            |
| tenant_id      | f8883ea7742f44ac8c5e6f75c050571c                                                  |
| updated_at     |                                                                                   |
| vim_id         | bbc84763-e4de-4e96-a9c3-6ffc4c272276                                              |
| vnfd_id        | b3df5522-a65b-40d4-a07a-257832f02811                                              |
+----------------+-----------------------------------------------------------------------------------+
```

Hình 4-13 Thông tin chi tiết VNF1

```
root@controller:~# tacker vnf-show vnf2
+----------------+-----------------------------------------------------------------------------------+
| Field          | Value                                                                             |
+----------------+-----------------------------------------------------------------------------------+
| attributes     | {"heat_template": "heat_template_version: 2013-05-23\ndescription:                |
|                | 'Demo example\n\n  '\nparameters: {}\nresources:\n  VDU1:\n    type:              |
|                | OS::Nova::Server\n    properties:\n      user_data_format:                           |
|                | SOFTWARE_CONFIG\n      availability_zone: nova\n      image: OpenWRT\n              |
|                | flavor:\n        get_resource: VDU1_flavor\n      networks:\n      -              |
|                | port:\n          get_resource: CP21\n      config_drive: false\n                  |
|                | CP21:\n    type: OS::Neutron::Port\n    properties:\n                              |
|                | port_security_enabled: false\n      network: net_mgmt\n  VDU1_flavor:\n           |
|                | type: OS::Nova::Flavor\n    properties:\n      vcpus: 1\n      disk:               |
|                | 1\n      ram: 512\noutputs:\n  mgmt_ip-VDU1:\n    value:\n                           |
|                | get_attr: [CP21, fixed_ips, 0, ip_address]\n"}                                    |
| created_at     | 2017-06-25 08:09:46                                                               |
| description    | Demo example                                                                      |
| error_reason   |                                                                                   |
| id             | 2a04012e-c690-4a36-8dc4-6148aa076550                                              |
| instance_id    | 26d4d6a8-67d0-4c7e-86e2-cd1385b5387f                                              |
| mgmt_url       | {"VDU1": "10.0.10.111"}                                                           |
| name           | VNF2                                                                              |
| placement_attr | {"vim_name": "REAL_VIM"}                                                          |
| status         | ACTIVE                                                                            |
| tenant_id      | f8883ea7742f44ac8c5e6f75c050571c                                                  |
| updated_at     |                                                                                   |
| vim_id         | bbc84763-e4de-4e96-a9c3-6ffc4c272276                                              |
| vnfd_id        | d3393795-498a-4567-a675-74845273d032                                              |
+----------------+-----------------------------------------------------------------------------------+
```

Hình 4-14 Thông tin chi tiết VNF2

Các VNF này đã được bật sẵn rule cho phép Forward gói tin.

```
config defaults
        option syn_flood        1
        option input            ACCEPT
        option output           ACCEPT
        option forward          REJECT
# Uncomment this line to disable ipv6 rules
#       option disable_ipv6     1

config zone
        option name             lan
        list   network          'net0'
        option input            ACCEPT
        option output           ACCEPT
        option forward          ACCEPT
```

Hình 4-15 VNF OpenWRT đã được cho phép forward gói tin.

Trước khi tạo VNFFG, máy Source traceroute tới máy Dest chỉ phải đi qua Router giữa hai đường mạng (10.0.10.1/24).

```
$ traceroute 192.168.100.115
traceroute to 192.168.100.115 (192.168.100.115), 30 hops max, 46 byte packets
 1  10.0.10.1 (10.0.10.1)  0.921 ms  0.598 ms  0.604 ms
 2  192.168.100.115 (192.168.100.115)  1.437 ms  0.956 ms  0.965 ms
$
```

Hình 4-16 Traceroute từ máy Source tới máy Dest trước khi tạo VNFFG

Ta tiến hành tạo VNFFG để hướng luồng dữ liệu khi traceroute của máy Dest tới máy Source phải đi qua các VNF đã khởi tạo.

```
root@controller:~# tacker vnffg-create --vnffgd-name vnffgd-no-param --vnf-mapping VNF1:'VNF
1',VNF2:'VNF2' VNFFG1
Created a new vnffg:
+------------------+--------------------------------------------------------------------+
| Field            | Value                                                              |
+------------------+--------------------------------------------------------------------+
| description      |                                                                    |
| forwarding_paths | b86531f5-bd6f-4289-a3c8-fa2ee5689c12                               |
| id               | 7088f2e3-fe67-428b-b253-fa434a99c8e7                               |
| name             | VNFFG1                                                             |
| status           | PENDING_CREATE                                                     |
| tenant_id        | f8883ea7742f44ac8c5e6f75c050571c                                   |
| vnf_mapping      | {"VNF2": "2a04012e-c690-4a36-8dc4-6148aa076550", "VNF1": "f0b0e28a-|
|                  | 69d4-41e5-9bca-655882df4c00"}                                      |
| vnffgd_id        | d375be99-b9df-4456-a43c-ab9f6384b93b                               |
+------------------+--------------------------------------------------------------------+
```

Hình 4-17 Tạo VNFFG tên là VNFFG1

VNFFG1 có thông tin VNFFGD chi tiết như sau:

```
description: Sample VNFFG template

node_templates:

  Forwarding_path1:
    type: tosca.nodes.nfv.FP.Tacker
    description: creates path (CP11->CP21)
    properties:
      id: 51
      policy:
        type: ACL
        criteria:
          - network_src_port_id: e81cea38-8992-4038-93ad-896312a8ab72
          - destination_port_range: 1024-65535
          - ip_proto: 17
      path:+
        - forwarder: VNF1
          capability: CP11
        - forwarder: VNF2
          capability: CP21

groups:
  VNFFG1:
    type: tosca.groups.nfv.VNFFG
    description: Traceroute from Dest to Source
    properties:
      vendor: tacker
      version: 1.0
      number_of_endpoints: 2
      dependent_virtual_link: [VL11,VL21]
      connection_point: [CP11,CP21]
      constituent_vnfs: [VNF1,VNF2]
    members: [Forwarding_path1]
```

Hình 4-18 Chi tiết tập tin VNFFGD của VNFFG1

VNFFGD này qui định các gói tin UDP (IP Protocol 17) từ máy Dest (Source Port ID như trên) phải đi qua các Forwader là VNF1 và VNF2. Đây là tên của hai VNFD của chính VNF1 và VNF2. Khi tạo VNFFG1, ta mapping VNFD trong VFFGD và VNF đã tạo trước đó.

Sau khi tạo VNFFG, gói tin máy Source traceroute tới máy Dest bắt buộc phải đi qua VNF1 và VNF2 trước rồi mới đến Router và máy Dest.

```
$ traceroute 192.168.100.115
traceroute to 192.168.100.115 (192.168.100.115), 30 hops max, 46 byte packets
 1  10.0.10.115 (10.0.10.115)  0.884 ms  0.637 ms  0.469 ms
 2  10.0.10.111 (10.0.10.111)  1.860 ms  1.280 ms  1.020 ms
 3  10.0.10.1 (10.0.10.1)  1.683 ms  1.332 ms  0.855 ms
 4  192.168.100.115 (192.168.100.115)  1.621 ms  1.748 ms  1.568 ms
$
```

Hình 4-19 Traceroute từ máy Source tới máy Dest sau khi tạo VNFFG

Chương 5 Phát triển công cụ hỗ trợ giám sát hệ thống NFV

Nội dung chương này là về công cụ hỗ trợ việc giám sát các VNF và VIM cho dịch vụ Tacker do nhóm tự phát triển dựa trên các API có sẵn của OpenStack.

5.1. Thực trạng

Hiện tại, dự án Tacker của OpenStack vẫn còn ở giai đoạn sơ khai, tuy đã hỗ trợ tương đối tốt việc điều khiển luồng và khởi tạo các VNF nhưng hệ thống theo dõi của Tacker vẫn chưa hoàn thiện, chỉ dừng ở mức biết được VNF/VIM có còn hoạt động hay không bằng kỹ thuật hearbeat đơn giản.

Displaying 2 items

☐	Name	Description	VIM Id	Auth URL	Regions	User	Project	Status
☐	VIM_DEVSTACK		4670c618-0acf-4d7d-bda7-148143466be5	http://10.0.0.50/identity/v3	RegionOne	nfv_user	nfv	REACHABLE
☐	REAL_VIM		bbc84763-e4de-4e96-a9c3-6ffc4c272276	http://10.0.0.40:5000/v3/	RegionOne	admin	admin	REACHABLE

Displaying 2 items

Hình 5-1 Giao diện quản lý VIM của Tacker

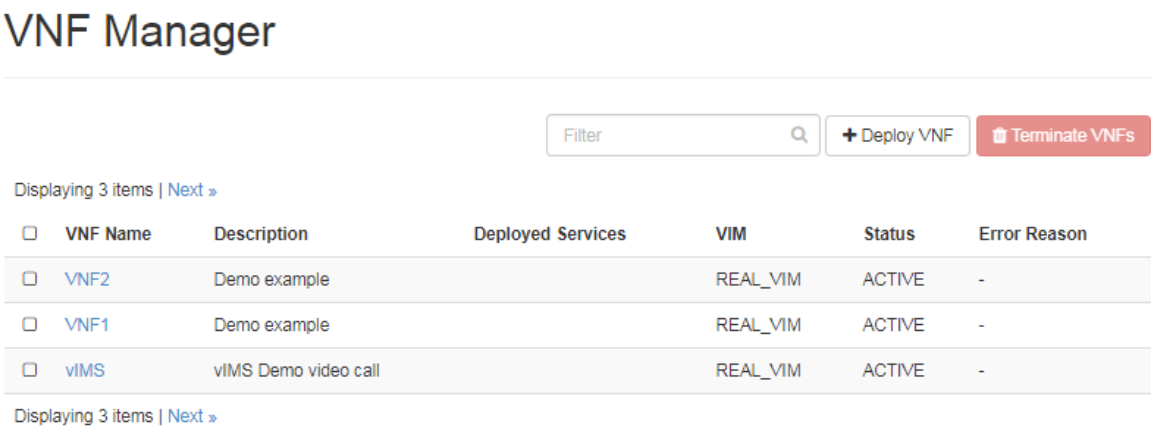
Khi ta đi sâu vào thông tin chi tiết của 1 VIM, ta cũng chỉ nhận được những thông tin theo dõi vô cùng đơn giản, chỉ gồm các event như: **CREATE** (khi đăng ký VIM), **MONITOR** (hệ thống đang hoạt động ổn định), **ERROR** (báo lỗi)…

Displaying 2 items | Next »

Event ID	Resource State	Time Since Event	Event Type	Event Details
3	PENDING	2017-06-25 08:08:46	CREATE	
4	REACHABLE	2017-06-25 08:08:46	MONITOR	

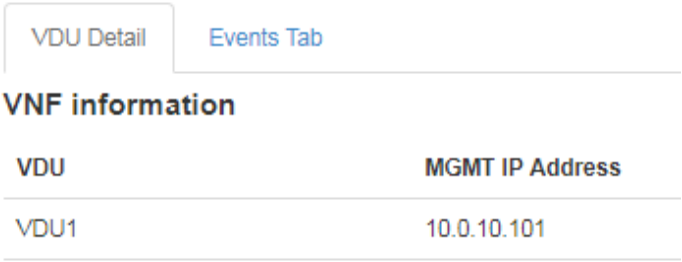
Displaying 2 items | Next »

Hình 5-2 Giao diện quản lý chi tiết một VIM Instance của Tacker



Hình 5-3 Giao diện quản lý các VNF của Tacker

Không chỉ có VIM, giao diện quản lý của VNF Manager cũng gặp phải tình trạng thiếu thông tin như vậy. Chỉ gồm các thông tin thông báo tình trạng hoạt động của VNF (**ACTIVE**/**ERROR**) và thông tin về IP quản lý của VNF đó.



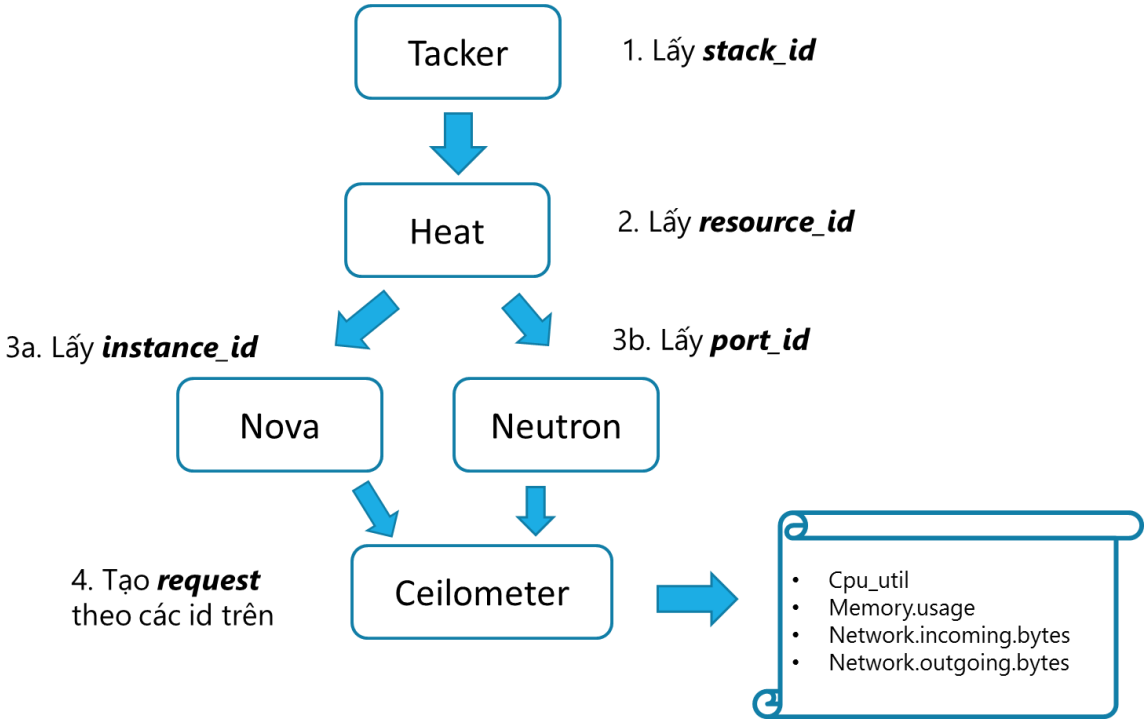
Hình 5-4 Giao diện quản lý chi tiết một VNF

Chính vì thiếu các tính năng chuyên sâu trong việc theo dõi hệ thống, người quản trị sẽ gặp rất nhiều khó khăn trong việc quản lý tài nguyên cũng như giám sát hoạt động của các VNF/VIM. Chính vì vậy, nhóm đã nghiên cứu và phát triển một công cụ hỗ trợ việc giám sát các VNF và VIM cho Tacker, tận dụng API từ những dự án khác trong hệ thống.

5.2. Kiến trúc

Công cụ của nhóm cho phép giúp người quản trị đơn giản hóa qui trình giám sát hệ thống và có cái nhìn trực quan hơn thông qua các đồ thị thời gian thực trên giao diện web nhờ tích hợp với dự án Horizon.

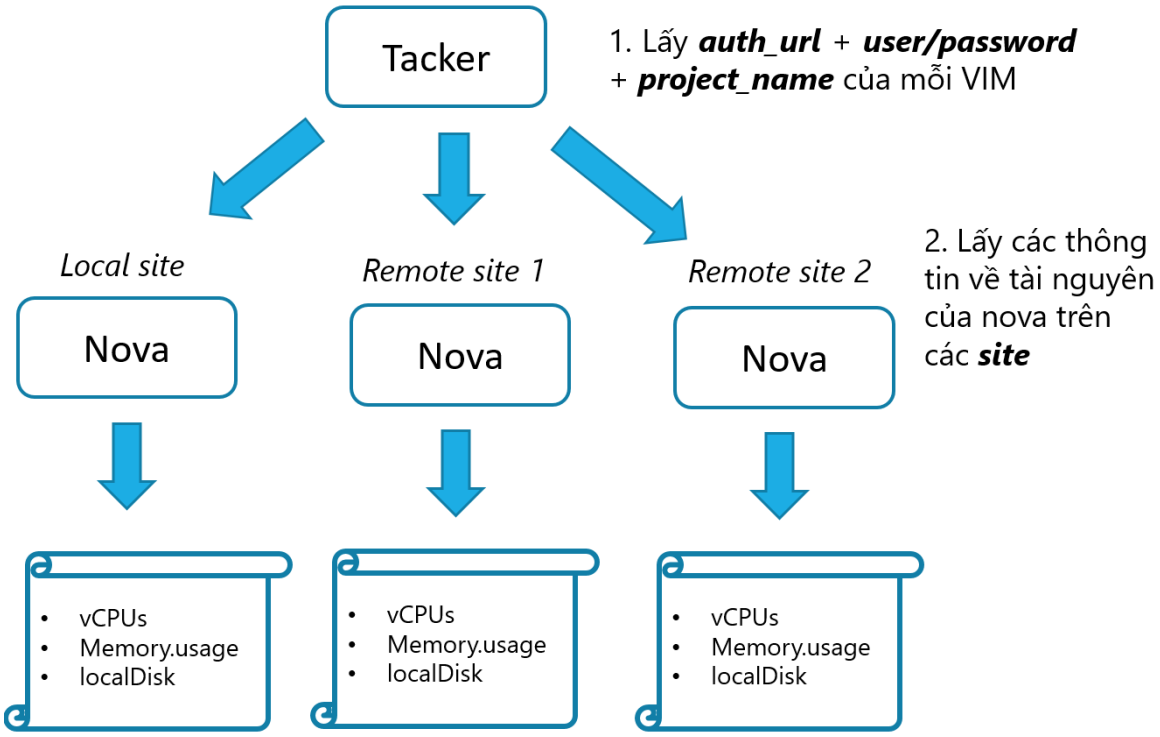
Hiện tại, công cụ của nhóm tập trung vào các thông tin như: lượng vCPU/RAM/Disk hiện có và khả dụng đối với VIM cũng như lưu lượng mạng vào/ra của các VNF, tốc độ truyền gói tin đối với VNF. Công cụ hoàn toàn có thể mở rộng để lấy được các thông tin khác như: số gói tin bị lỗi/bị hủy, tốc độ đọc/ghi của ổ cứng,…



Hình 5-5 Quy trình theo dõi các VNF

Về quy trình theo dõi VNF, nhóm dựa trên các API do Tacker cung cấp, lấy ***stack_id*** của VNF. Sau đó, dựa trên ***stack_id*** này, thông qua dự án Heat, chương trình sẽ lần lượt lấy tiếp ***resource_id*** và ***instance_id*** (Nova) cũng như ***port_id*** (Neutron).

Cuối cùng, dựa trên các id vừa lấy được, chương trình sẽ gởi một request cho Ceilometer để truy xuất các thông tin như ***Network.incoming.bytes, Network.outgoing.bytes, cpu_util, memory.usage*** của VNF đó.



Hình 5-6 Quy trình theo dõi VIM tại các site khác nhau

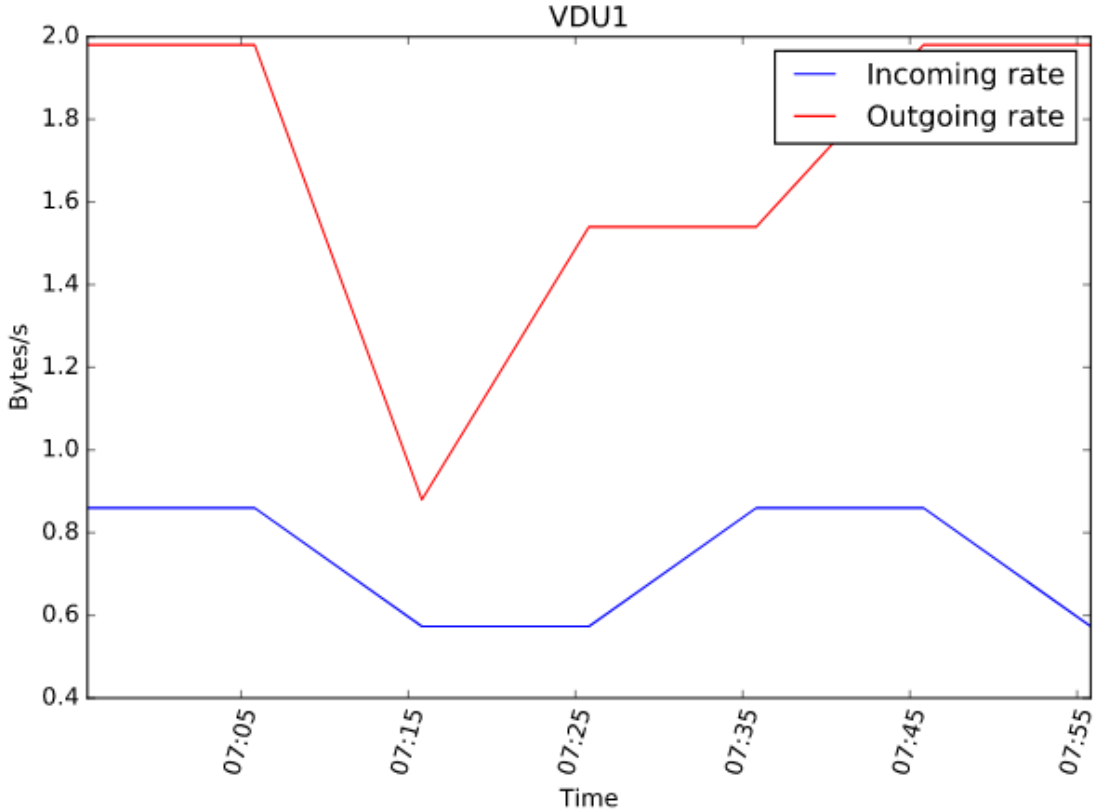
Đối với quy trình theo dõi các VIM của Tacker, do đặc thù của một hệ thống NFV là sẽ có thể có rất nhiều site đặt tại nhiều vị trí địa lý khác nhau nên cần phải có một hệ thống nhằm quản lý tập trung các VIM này lại với nhau.

Nhóm dựa trên những API sẵn có của Tacker, lấy được các thông tin về ***auth_url, user/password, project_name*** của từng VIM và dựa trên những thông tin này để truy xuất các thông tin về hệ thống của từng site: số lượng vCPU của từng VIM, dung lượng bộ nhớ hiện tại, dung lượng đĩa cứng đang có…

Nắm được những thông tin này sẽ giúp người quản trị dễ dàng hơn trong việc lên kế hoạch cấp phát tài nguyên, khởi tạo các VNF tại vị trí phù hợp nhất.

5.3. Kết quả

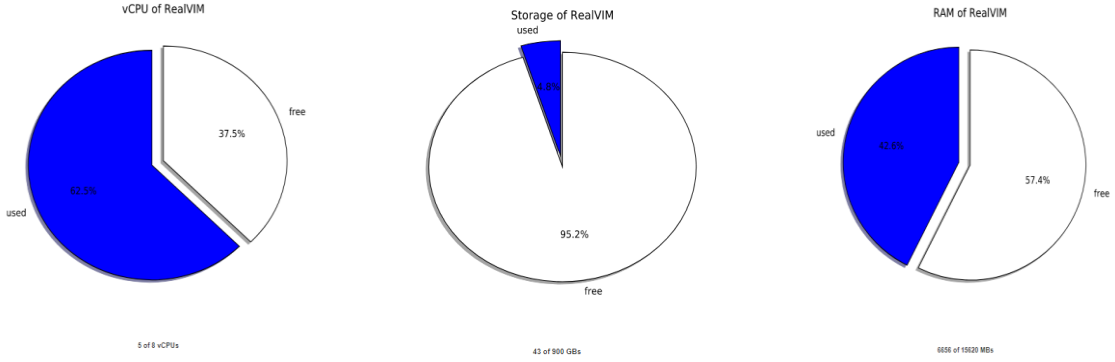
Nhóm đã xây dựng thành công công cụ hỗ trợ giám sát hệ thống với kiến trúc đã trình bày ở trên. Công cụ thể hiện được các thông tin về lưu lượng mạng của VNF theo thời gian thực.



Incoming Bytes	Outgoing Bytes	Total Bytes
2669748.0	925212.0	3594960.0

Hình 5-7 Công cụ theo dõi các VNF

Biểu đồ cho biết thông lượng mạng đang chạy qua VNF/nhóm VNF tại nhiều thời điểm. Công cụ này hoàn toàn có thể được mở rộng để phục vụ các ứng dụng thực tiễn như use case vCPE, vRAN,… Nhà cung cấp dịch vụ Internet có thể theo dõi được tốc độ sử dụng thực tế của người dùng mạng, lượng dữ liệu được sử dụng,.. Từ đó, nhà cung cấp có thể quản lý chất lượng dịch vụ, tính cước tốt hơn.



Hình 5-8 Công cụ theo dõi các VIM

Tương tự với công cụ theo dõi thông tin của các VIM Instance. Giao diện này giúp việc quản lý tài nguyên của các VNF dễ dàng hơn, tiện lợi cho người quản trị khi cấu hình các ứng dụng mạng phức tạp. Bên cạnh đó, trình theo dõi VIM cung cấp một cái nhìn tổng quan hơn về hạ tầng hệ thống đặc biệt là với hệ thống NFV có sử dụng nhiều VIM ở các vùng địa lý khác nhau.

Chương 6 Tổng kết và hướng phát triển

6.1. Kết quả đạt được

Về tìm hiểu lý thuyết, nhóm đã đạt được kết quả sau:

- Hiểu được tổng quan về công nghệ ảo hóa chức năng mạng, quá trình phát triển, kiến trúc, các lợi ích, các usecase và tiềm năng phát triển ở Việt Nam
- Hiểu được mối liên hệ giữa SDN và NFV.
- Tìm hiểu, thử nghiệm và đánh giá các giải pháp NFV như OpenStack, OPNFV, Tacker, OpenSourceMANO, OpenBaton.

Về mặt triển khai hệ thống thử nghiệm, nhóm đã đạt kết quả sau:

- Xây dựng thành công hạ tầng NFV trên nền tảng OpenStack
- Thử nghiệm thành công hai mô hình NFV với hai giải pháp MANO khác nhau là OpenBaton và Tacker.
- Triển khai thử nghiệm dịch vụ mạng vIMS.
- Triển khai thử nghiệm usecase Đồ thị chuyển tiếp VNF.
- Phát triển plugin giao diện quản lý NFV cho Tacker với khả năng theo dõi các VIM và VNF của hệ thống.

6.2. Hướng phát triển của đề tài

Vì thời gian nghiên cứu có hạn, đồng thời NFV là một công nghệ mới và vẫn đang trong giai đoạn hoàn thiện, chính vì thế nhóm còn rất nhiều vấn đề chưa thể thực hiện trong phạm vi của luận nay. Sau đây là một số hướng phát triển cho đề tài về NFV được nhóm đề ra:

- Nghiên cứu chuyên sâu hơn về các công nghệ tối ưu hiệu năng của hạ tầng ảo hóa chức năng mạng như: DPDK, SR-IOV, Huge Pages, CPU Pinning…
- Nghiên cứu, triển khai thử nghiệm NFV kết hợp với SDN.
- Ứng dụng công nghệ Container vào việc triển khai các VNF.
- Phát triển Portal để người trị có thể quản lý, triển khai dịch vụ mạng theo giao diện GUI và thao tác kéo thả.

Danh mục tài liệu tham khảo

[1]. "Accelerating NFV Delivery with Openstack" – OpenStack Foundation

[2]. "Cisco Visual Networking Index 2015-2020" – Cisco

[3]. "ETSI NFV Management and Orchestration" – ETSI

[4]. "NFV Architectural Framework" – ETSI

[5]. "NFV for Beginners" – Redhat

[6]. "NFV Plugtest Report 2017" – ETSI

[7]. "NFV Use case" – ETSI

[8]. "NFV White Paper 2012, 2013, 2014" – ETSI

[9]. "OSM Whitepaper Release TWO" – OpenSource MANO

[10]. "Performance evaluation of Data Path Improvements in NFV OpenStack Deployment" - De Langhe, Kris.

[11]. "Report on SDN Usage in NFV Architecture" – ETSI

[12]. "SDN/NFV: Strategies and Progresses in Network Operators" - Dr. Harrison J. Son (Netmanias)

[13]. "Tìm hiểu OpenStack và triển khai hệ thống thử nghiệm" – Nguyễn Duy Vũ, Đặng Quốc Pháp – Khóa luận tốt nghiệp 2015, ĐH-KHTN

[14]. "Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0" – OASIS

[15]. "Understanding OPNFV" - OPNFV

[16]. "Verizon SDN – NFV Reference Architecture" – Verizon

PHỤ LỤC

Alarm monitoring framework của Tacker

Tacker hiện có hỗ trợ một framework dùng để theo dõi các VNF dựa trên cơ chế alarm-trigger dựa trên monitor driver của Tacker.

Khi người dùng muốn sử dụng chức năng này của Tacker, cần tạo thêm mục policies trong file VNFD trước khi khởi tạo một VNF

```
policies:
- vdu1_cpu_usage_monitoring_policy:
    type: tosca.policies.tacker.Alarming
    triggers:
      resize_compute:
          event_type:
              type: tosca.events.resource.utilization
              implementation: ceilometer
          metrics: cpu_util
          condition:
              threshold: 50
              constraint: utilization greater_than 50%
              period: 65
              evaluations: 1
              method: avg
              comparison_operator: gt
          actions: [respawn]
```

Cấu hình tính năng alarm-trigger trong Tacker.

Theo policy được mô tả trong hình, alarm này có tác dụng tạo lại máy ảo (respawn) khi lượng CPU được sử dụng nhiều hơn 50% trong vòng 65 giây.

Hiện alarm framework này của Tacker đã hỗ trợ các thao tác như: ***scaling, log, respawn, log_and_kill***. Người quản trị có thể tùy chọn các hành động này trong file TOSCA với những điều kiện khác nhau.

Tuy chưa hoàn chỉnh, nhưng framework này có khả năng ứng dụng rất cao trong thực tế nhờ cơ chế kích hoạt tự động trigger theo policy.

Viết một monitor driver mới cho Tacker

Tacker còn hỗ trợ người dùng tự phát triển thêm một driver để theo dõi các trạng thái của VNF trong quá trình triển khai cũng như nhằm quản lý các VNF này.

Để viết một module theo dõi, ta cần phải gọi lại lớp:

tacker.vnfm.monitor_drivers.abstract_driver.VNFMonitorAbstractDriver

Đây chính là lớp mẫu chứa các hàm cơ bản cần phải định nghĩa lại.

Lấy ví dụ một driver hiện đã có là Ping. Ping sẽ được định nghĩa như sau:

tacker.tacker.monitor_drivers =

ping = tacker.vnfm.monitor_drivers.ping.ping:VNFMonitorPing

Trong đó, Ping sẽ phải kế thừa và viết lại các hàm cơ bản của lớp *VNFMonitorAbstractDriver* như sau:

def get_type(self)

Trả về loại của monitor driver (ví dụ như: ping).

def get_name(self)

Trả về tên của của plugin này.

def get_description(self)

Trả về thông tin miêu tả của driver này.

def monitor_get_config(self, plugin, context, vnf)

Trả về thông tin cấu hình của driver dưới dạng dictionary trong python.

def monitor_url(self, plugin, context, vnf)

Trả về url dùng để theo dõi VNF.

def monitor_call(self, vnf, kwargs)

Hàm này sẽ trả về thông tin trạng thái 'True' nếu VNF hoạt động tốt và trả về 'failure' hoặc 'calls-capacity-reached' nếu có lỗi.

Đối với hàm def monitor_call(self, vnf, kwargs) nếu hàm này trả về những kết quả khác ngoài 'True' thì ta cần phải nối những kết quả đó với một hành động cụ thể, tương tự như cấu hình tập tin TOSCA ở trên.

Lấy ví dụ như sau:

```
vdu1:
 monitoring_policy:
   ping:
     actions:
       failure: respawn
```

Trong ví dụ này, vdu1 được cấu hình sử dụng driver theo dõi là ping. Ping sẽ thực hiện hành động ***respawn*** vdu1 này kếu ***monitor_call*** trả về sự kiện ***failure***.

Một ví dụ khác:

```
vdu2:
 monitoring_policy:
   http-ping:
     monitoring_params:
       port: 8080
       url: ping.cgi
     actions:
       failure: respawn
```

Trong ví dụ này, vdu1 được cấu hình sử dụng driver theo dõi là http-ping. Tương tự ping, http-ping sẽ thực hiện hành động ***respawn*** vdu2 này kếu ***monitor_call*** trả về sự kiện ***failure***. Bên cạnh đó, http-ping còn được cấu hình thêm các thông tin theo dõi là **port 8080** và **url ping.cgi.**