

UNIVERZITA HRADEC KRÁLOVÉ
FAKULTA INFORMATIKY A MANAGEMENTU
KATEDRA INFORMATIKY A KVANTITATIVNÍCH METOD

Orchestrace a management virtuálních síťových
funkcí

DIPLOMOVÁ PRÁCE

Autor: Bc. Ondřej Smola

Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Vladimír Soběslav, Ph.D.

Hradec Králové

duben, 2016

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a uvedl jsem všechny použité prameny a literaturu.

V Hradci Králové dne 18. dubna 2016

Ondřej Smola

Poděkování

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean placerat. Duis pulvinar. Maecenas lorem. Mauris tincidunt sem sed arcu. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt.

Anotace

Tato diplomová práce pojednává o aktuálním tématu, kterým je Virtualizace síťových funkcí (Network function virtualization).

Annotation

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean placerat. Duis pulvinar. Maecenas lorem. Mauris tincidunt sem sed arcu. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Phasellus rhoncus. Praesent vitae arcu tempor neque lacinia pretium. Mauris suscipit, ligula sit amet pharetra semper, nibh ante cursus purus, vel sagittis velit mauris vel metus. Etiam posuere lacus quis dolor. Curabitur bibendum justo non orci. Praesent in mauris eu tortor porttitor accumsan. Nullam lectus justo, vulputate eget mollis sed, tempor sed magna. Donec quis nibh at felis congue commodo. Integer tempor. Maecenas libero.

Obsah

1	Úvod	1
2	Základní problematika virtualizace síťových funkcí	3
2.1	Potřeba virtualizaci síťových funkcí (NFV)	3
2.2	Základní princip virtualizace	5
2.3	Cloud Computing	6
2.3.1	Distribuční modely	7
2.3.2	Modely nasazení	8
2.4	Souvislost NFV a SDN	9
2.4.1	Service Chaining	10
2.5	Architektura NFV a VNF	11
2.5.1	Infrastruktura NFV	13
2.5.2	Virtuální síťová funkce	14
2.5.3	Management a orchestrace NFV	15
2.5.3.1	TOSCA	17
2.6	Možné technologie pro řešení	17
2.6.1	Hypervisory	17
2.6.2	VNF	18
2.6.3	Cloud platforma	18
2.6.3.1	OpenStack	19
2.6.3.2	VMware vCloud	20
2.6.4	SDN	21
2.6.4.1	OpenContrail	21
2.6.4.2	VMware NSX	22
3	Tvorba VNF v cloudovém prostředí	24
3.1	Požadavky na NFV řešení	24
3.2	Architektura navrženého řešení	24
3.3	Load balancer as a Service	26
3.3.1	Scénář použití LbaaS	27
3.3.2	Neutron LbaaS	27
3.3.3	LbaaS heat template	29
3.4	Firewall as a Service	33
3.4.1	Servisní instance v OpenContrailu	33
3.4.2	Scénář NAT	35
3.4.3	Scénář HA firewall	35

3.4.4 FwaaS template	35
4 Ukázka praktického spuštění jednotlivých VNF	40
4.1 Testovací topologie	40
4.2 Testované síťové funkce	41
4.3 Testování LbaaS	41
4.4 Testování FwaaS	42
4.4.1 Fortigate VM	43
4.4.2 PFSense	43
5 Shrnutí poznatků a možná budoucí rozšíření	47
6 Závěr	48
Literatura	49
Přílohy	I

1 Úvod

V dnešní době dochází v datových centrech k nasazování nových moderních technologií. Jednou z nich je například virtualizace a to především v oblasti výpočetního výkonu a úložišť. Je již běžnou praxí, že v datových centrech vše běží na jedné fyzické infrastruktuře, která je abstrahovaná na jeden souvislý blok výpočetního výkonu a jeden souvislý blok úložiště. Dalším takovýmto funkcionálním blokem v datových centrech jsou počítačové sítě. Avšak v počítačových sítích byl, oproti dvěma zmíněným oblastem, pomalejší vývoj inovací a není zde tolik využívána virtualizace. Pro zvýšení efektivity je proto nutné, aby se počítačové sítě staly programovatelnými a mohli být spravovány z jednoho centrálního místa.

Dnes je však zatím síťové funkčnosti soustředěno ve fyzických proprietárních zařízeních jako jsou routery, firewally či load balancery. To znamená, že provozovatelé počítačových sítí se při spouštění nových síťových služeb musí na tyto zařízení spoléhat. Což může vést k zdlouhavému nasazování, zvýšené spotřebě energií a investici do školení pracovníků pro dané proprietární zařízení. Zároveň zde není možnost, aby síť mohla být dynamicky ovládána dle aktuálních požadavků uživatelů sítě. Například vývojář nemůže hned nasadit aplikaci do produkce. Musí nejprve čekat na síťový tým než patřičně nakonfiguruje síťové prvky pro správné a bezpečné fungování celé infrastruktury.

Virtualizace síťových funkcí se zaměřuje na transformaci způsobu, jakým síťový architekti přistupují k oblasti počítačových sítí a to pomocí stávajících a neustále se vyvíjejících virtualizačních technologií. Snaha je tedy přesunout mnoho typů síťového příslušenství z fyzických síťových prvků do standardních průmyslově používaných serverů a úložišť, které mohou být umístěny v datových centrech či přímo u koncových zákazníků. Tímto lze dosáhnout virtuálních síťových funkcí, které mají naprosto stejnou funkcionalitu jako síťové funkce umístěné v síťových prvcích.

Cílem této diplomové práce je analyzovat aktuální stav v oblasti virtualizace síťových funkcí. Dále je cílem navrhnout jednoduchý framework pro virtualizaci síťových funkcí spolu s ukázkou několik příkladů síťových funkcí. Celý framework by měl sloužit k možnosti rychlého a jednoduchého nasazení vybraných síťových funkcí. Současně by k jeho vytvoření měli být použity aktuálně dostupné technologie. Toto řešení musí být univerzální, nezávislé na vendorech a flexibilní.

Celá struktura této práce je rozdělena na několik částí. V druhé kapitole jsou vysvětleny hlavní pojmy a problematika oblasti virtualizace síťových funkcí. Třetí je popis návrhu řešení frameworku pro virtualizaci síťových funkcí a popisu použitých použitých technologií pro tento návrh. Ve čtvrté kapitole je věnována testování a ukázce, jak navržený framework funguje. Na konci této práce dojde k závěrečnému shrnutí.

Závěrečná práce byla zpracována ve spolupráci s firmou tcp cloud a.s., která poskytuje implementace jednoho z nejlepších cloudových řešení na světě. Firma umožnila využít jejich stávající infrastrukturu v nejmodernějším datovém centru v České republice, které je v budově Technologického centra Písek s.r.o.

2 Základní problematika virtualizace síťových funkcí

Jak již vyplývá z názvu, tak tato kapitola se zabývá základní analýzou a popisem problematiky spojené s oblastí virtuální síťových funkcí. Budou zde vysvětleny hlavní důvody pro virtualizaci síťových funkcí, základní principy virtualizace, cloud computing a referenční architektura frameworku pro virtualizaci síťových funkcí. Poté budou zmíněny příklady některých dostupných technologií a řešení, které k virtualizaci síťových funkcí lze použít.

Pro lepší pochopení a přehlednost celé této práce zde budou rozlišeny následující pojmy, se kterými se lze také setkat v odborné literatuře a které budou dále v této práci používány.

- Síťová funkce (Network function - NF) - Toto je komponenta síťové infrastruktury, která má dobře definované funkční chování, jako například směrování, NAT, Load balancing, Intrusion detection, atd.
- Virtuální síťové funkce (Virtual network function - VNF) - Je stejná jako NF, ale zde je funkčnost implementována pomocí softwaru a je nezávislá na hardwaru, na kterém běží.
- Virtualizace síťových funkcí (Network Functions Virtualization - NFV) - Zde se jedná o označení celého konceptu či frameworku.

2.1 Potřeba virtualizaci síťových funkcí (NFV)

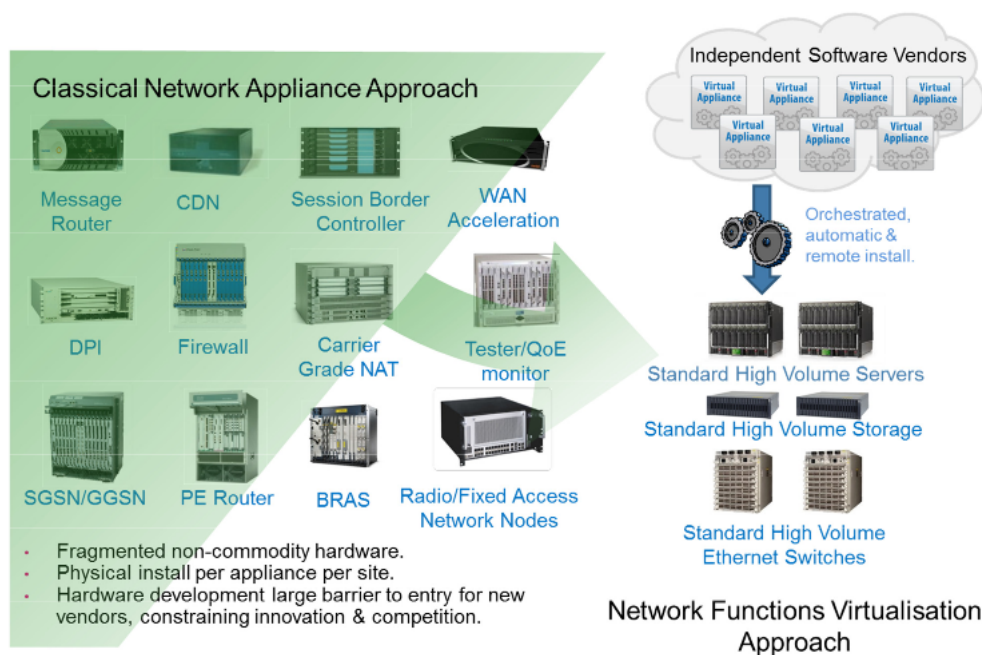
Produkční vývoj v telekomunikačním průmyslu se tradičně řídil přísnými standardy kvůli stabilitě a kvalitě komunikace. Přestože tento model v minulosti fungoval, tak vedl nevyhnutelně k dlouhým produkčním cyklům, pomalému tempu vývoje a spoléhání se na proprietární či specializovaný hardware. S příchodem výrazné konkurence v komunikačních službách, od rychle postupujících organizací operujících ve velkém měřítku na veřejném internetu, podnítil poskytovatele služeb pro hledat nových způsobů, jak změnit dosavadní způsob produkčního vývoje.

Pro vyřešení toho problému bylo navrženo v publikacích [1] a [2] skupinou několika telekomunikačních provozovatelů ETSI řešení ve formě virtualizace síťových funkcí (network functions virtualization). Později vzniklo více projektů zabývajících se touto oblastí jako například OPVNF [3] Hlavní cíle tohoto řešení jsou zlepšit následující aspekty provozu telekomunikačních sítí:

- Smíření investičních nákladů – snížení potřeby nákupu jednoúčelových hardwarových zařízení, možnost platby pouze za využití kapacity a snížení rizik přílišného předimenzování kapacit
- Snížení provozních nákladů – snížení prostoru, napájení a požadavky na chlazení, zjednodušení správy a řízení síťových služeb
- Urychlení Time-to-market – zkrácení doby pro nasazení nových síťových služeb, chopení se nových příležitosti na trhu, vyhovění potřebám zákazníka
- Doručit agilitu a flexibilitu – možnost rychle škálovat (rozšiřovat nebo zmenšovat služby) dle měnících se požadavků od zákazníka. Podpora služeb, které mají být dodány pomocí softwaru na libovolném standardním serverovém hardwaru

Jak je uvedeno v [4] a [5], tak celá myšlenka je založena na tom, že dojde k separování softwarové funkcionality v síťových prvcích od proprietárního hardwaru, na kterém běží. To umožní se síťovými funkcemi zacházet jako s klasickými softwarovými aplikacemi, které mohou běžet na standardním komerčně dostupných serverech jenž organizace v současnosti používají. Tím bude zároveň umožněno flexibilní nasazování těchto síťových funkcí a jejich dynamický provisioning. Díky tomu, že jsou síťové funkce odděleny od hardwaru, tak je také možné jejich vhodnější umístění v topologii. To znamená dle požadavků na umístění mohou být nasazeny v datových centrech, síťových uzlech či přímo v uživatelské koncové bodě. Hlavní koncept virtualizace síťových funkcí znázorňuje obrázek č. 2.1.

Za zmínění stojí poznámka v [4], kde je řečeno, že obecný koncept oddělení síťové funkce od hardwaru ještě nutně neznamená potřebu využití virtualizace. Protože budou síťové funkce dostupné jako software, tak mohou být nainstalovány a provozovány přímo na fyzickém stroji. Ovšem rozdíl je, že tento stroj již nebude speciální hardware, ale klasický server. Tento scénář může být do jisté míry použit při nasazování síťových funkcí v malém měřítku např. v uživatelských koncových bodech. Avšak pro plné využití všech výše zmíněných výhod, které jsou třeba ve velkých datových centrech, je třeba s použitím virtualizace počítat. To vše umocňuje fakt, že většina datových center v současnosti již využívá cloud computing.



Obrázek 2.1: Koncept virtualizace síťových funkcí (NFV)

2.2 Základní princip virtualizace

Základním problémem v tradičním modelu IT infrastruktury je, že servery podporují pouze jeden operační systém v čase. Na tomto systému obvykle běží pouze jedna aplikace. Přestože by na tomto systému popř. serveru mohlo běžet více aplikací, tak je lepší držet aplikace odděleně na různých systémech z důvodu minimalizace potenciálních bodů selhání. Pokud například nastane s aplikací problém, tak častým řešením je restartování systému. Pokud by na systému bylo více aplikací, znamenalo by to jejich vyřazení z provozu po dobu restartu, který může trvat velice dlouho. [6]

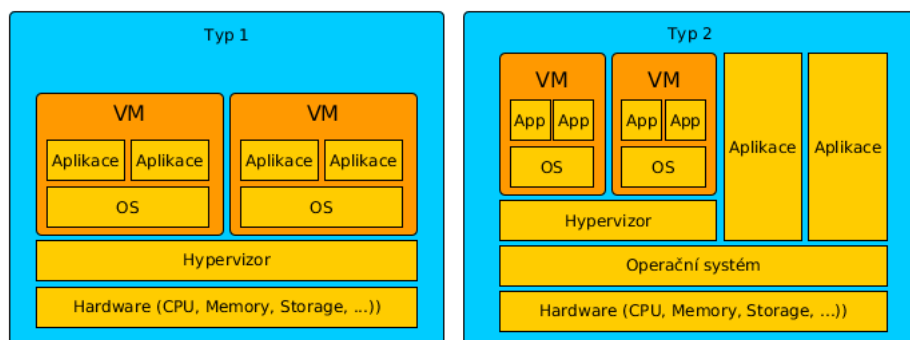
Z výše zmíněných důvodů došlo k velkému rozvoji a nasazování virtualizace. Virtualizací obecně označujeme techniky, které umožňují k dostupným hardwarovým zdrojům přistupovat jiným způsobem, než jakým fyzicky existují. Je tomu díky softwaru, který tento hardware abstrahuje a vytvoří tím virtuální prostředí. Virtualizované prostředí se dá snadněji přizpůsobit potřebám uživatelů, případně skrýt pro uživatele nepodstatné detaily (jako např. rozmístění hardwarových prostředků). Tím je tedy umožněno na jednom fyzickém serveru provozovat více od sebe oddělených virtuálních strojů, které mají každý svůj vlastní operační systém s aplikacemi. Software pro virtualizaci se nazývá hypervisor.[6]

Jak zmiňuje [7], tak existují tyto dva základní typy hypervisorů:

- Typ 1 (Nativní) - Tento hypervisor běží přímo na fyzickém hardwaru. Tím umožň-

ňuje provozovat více operačních systémů na jednom fyzickém stroji. Příkladem takového hypervisoru je VMware ESXi a XEN.

- Typ 2 (Hostovaný) - Na rozdíl od předchozího případu tento typ hypervisoru běží v prostředí operačního systému. Příkladem je například KVM či Microsoft Hyper-V



Obrázek 2.2: Schéma hypervisorů

Obrázek 2.2 zobrazuje schématický popis obou hypervisorů a jejich rozdíl. Problematika virtualizace je velice rozsáhlá a více informací o ní poskytují zdroje [6] a [7].

2.3 Cloud Computing

Jak již bylo zmíněno, tak cloud computing, nebo někdy také označován pouze jako cloud, je oblast, ve které je velký potenciál pro využití virtuálních síťových funkcí. Z tohoto důvodu bude v této části jeho problematika přiblížena.

Cloud je technologie, kterou v poslední době začala provozovat většina větších organizací, jak ukazuje [8]. Cloud Computing má mnoho definic. Dle definice uvedené v [9] ho lze charakterizovat jako poskytování služeb, programů a výpočetních zdrojů servery dostupnými z internetu s tím, že uživatelé k nim mohou přistupovat vzdáleně.

Z technického hlediska tvoří cloud veškeré služby poskytované přes Internet a zároveň i infrastruktura, která tyto zdroje poskytuje. Tuto infrastrukturu tvoří velké množství fyzických serverů, které jsou vzájemně propojeny. Na těchto serverech běží hypervisor, který vytvoří virtuální infrastrukturu. Pro vytváření cloudových služeb zde musí ještě existovat cloudová platforma, která dokáže celou tuto virtuální infrastrukturu spravovat. [9]

Dle [9] existuje 5 základních atributů, kterými se cloud computing vyznačuje. Jsou to tyto:

- Služby dostupné na požádání
- Všudypřítomný síťový přístup
- Sdílení zdrojů
- Vysokou elasticitu
- Měření využitých zdrojů

2.3.1 Distribuční modely

Dle [10] lze cloudové služby lze rozdělit do 3 základních kategorií. V [11] jsou k těmto kategoriím přiřazeny příklady užití z oblasti virtualizace síťových funkcí.

- Infrastructure as a Service (IaaS) - Nejzákladnější model poskytování cloudových služeb. IaaS cloudové platformy nabízejí například výpočetní výkon, virtuální disky, blokové a souborové úložiště či virtuální síť. Poskytovatelé IaaS cloudových platform poskytnou tyto zdroje na vyžádání ze svých datových center. Toto je možné díky skupině hypervisorů v rámci cloudu, které mohou provozovat velké množství virtuálních strojů a mají schopnost škálovat poskytované služby v závislosti na měnících se požadavcích přicházejících od zákazníků. Tento model může tedy sloužit i pro poskytnutí všech potřebných zdrojů celé infrastruktury pro virtualizaci síťových prvků, neboli Network Function Virtualization Infrastructure as a Service. Zde má uživatel pod nejvíce možností, jak navrhnout a spravovat virtuální síťové funkce, protože v zásadě dokáže nasazovat i vlastně navržené síťové funkce a nejen ty, které mu poskytuje provozovatel cloudu.
- Platform as a Service (PaaS) - V modelu Platforma jako služba (PaaS) hostují poskytovatelé cloudových služeb určitou počítačovou platformu, kterou následně poskytují koncovým uživatelům přes Internet. Tato platforma většinou bývá prostředím nějakého operačního systému, prostředím pro běh určitého programovacího jazyka, databáze a webový server. Vývojáři aplikací tím pádem mohou provozovat a případně vyvíjet svá softwarová řešení bez výrazných nákladů a složitého nákupu a konfiguraci potřebného hardwaru a softwaru. Některé PaaS platformy nastavuje výpočetní a úložné prostředky aplikace automaticky tak, aby odpovídala aktuálním požadavkům aplikace bez nutnosti zásahu zákazníka. NVF v tomto modelu může nabízet síťové služby, které se mohou skládat z více virtuálních síťových funkcí, neboli Virtual Network Platform as a Service. Zde je poskytnuta uživateli velká kontrola nad konfigurací a ovládáním celé platformy.

- Software as a Service (SaaS) - V modelu SaaS provozují poskytovatelé cloudových služeb aplikační software v cloudu a uživatelé k tomuto softwaru přistupují pomocí klientského software (např. webové prohlížeče). Uživatelé cloudu tedy nespravují infrastrukturu ani platformu, kde aplikace běží. Není proto třeba zde nic instalovat a spouštět aplikace na vlastních počítačích uživatele, což velmi zjednodušuje údržbu. Cloudové aplikace se liší od ostatních aplikací v možnostech škálování, kterého může být dosaženo díky distribuci úkolů na více virtuálních strojů, a tím reagovat na měnící se poptávku. Tento proces je pro uživatele služby transparentní, uživatel vidí pouze jeden přístupový bod pro danou aplikaci. Do této kategorie služeb může patřit poskytování virtuálních síťových funkcí, která je pouze ve formě softwarové aplikace, neboli VNFaaS. Takovéto aplikace poskytují síťovou funkci pro síťové správce a uživatele nejčastěji v privátním cloudu.

2.3.2 Modely nasazení

Existuje několik základních modelů nasazení cloud computingu resp. cloudových platforem, které uvádí [10]. V [11] lze k nim opět najít určité příklady z oblasti virtualizace síťových funkcí.

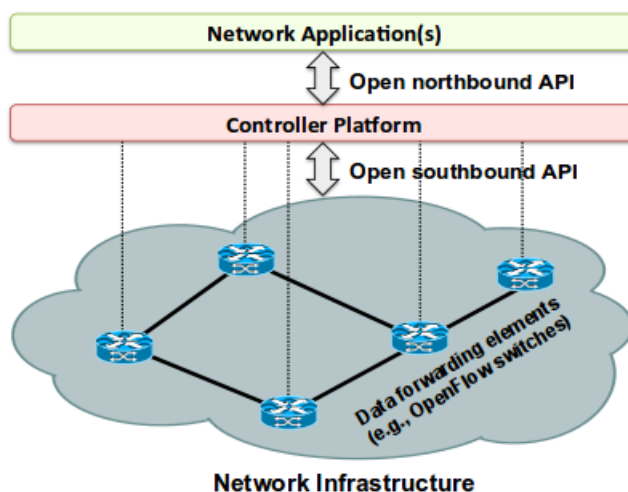
- Privátní cloud - Privátní cloud je infrastruktura provozována výhradně v rámci jedné organizace. Může být spravován interně nebo prostřednictvím třetí strany a hostování může být opět interní nebo externí. Aby mohl podnik využít privátní cloud, musí nejprve navrhnout a uzpůsobit k tomuto účelu svoji stávající infrastrukturu, která musí být virtualizována. Vlastní přechod vyvolává řadu bezpečnostních otázek, které je třeba řešit, aby se zabránilo vážným zranitelnostem celého řešení. Privátní cloud je přesně ten typ modelu, kde lze najít využití pro virtualizaci síťových funkcí.
- Veřejný cloud - Veřejné cloudové jsou cloudové služby, jako jsou aplikace, výpočetní výkon, úložiště a další, které jsou k dispozici široké veřejnosti. Služby jsou poskytovány zdarma nebo podle modelu platby za množství použitých služeb. Je zvykem, že veřejní poskytovatelé cloudových služeb, jako je Amazon AWS, Microsoft nebo Google, vlastní a provozují hardwarovou infrastrukturu a nabízejí k ní přístup pouze přes Internet. V tomto modelu není očekáváno využívání NFV.
- Hybridní cloud - Hybridní cloud je spojení dvou nebo více cloudů (soukromých, komunitních nebo veřejných), které zůstávají samostatné, ale jsou těsně propojeny. Toto složení rozšiřuje možnosti nasazení cloudových služeb a tím umožňuje

IT organizacím využít veřejné cloudové prostředky k uspokojení dočasných potřeb. Tato schopnost umožňuje hybridním cloudům škálovat přes více nezávislých cloudů. V tomto modelu může být využito NFV především na straně soukromého cloudu.

- Komunitní cloud - V rámci komunitního cloudu sdílí infrastrukturu cloudu několik organizací, které mají společné zájmy (bezpečnost, dodržování předpisů, působnost, atd.). Komunitní cloud může být spravován interně nebo prostřednictvím třetí strany. Náklady jsou rozloženy mezi méně uživatelů než na veřejném cloudu. V tomto modelu může být využito NFV, pokud se provozovatelé takového cloudu domluví.

2.4 Souvislost NFV a SDN

Softwarově definované sítě (SDN) je další z nových technologií, která se snaží vylepšit a automatizovat správu stávajících počítačových sítí. Dle [12] jde o koncept, ve které je oddělena řídicí logika (control plane) z jednotlivých routerů a switchů, které přeposílají traffic (data plane). Tím, že dojde k oddělení datové a řídicí vrstvy, se routery a switche stanou pouze přeposílající data a veškerá řídicí logika může být implementována v jednom logicky centrálním místě (SDN Controller). Z tohoto centrálního místa lze do jednotlivých routerů a switchů předávat instrukce pomocí aplikačních programovacích rozhraní (API). Samotný SDN Controller také obsahuje API, které mohou využívat aplikace a tím řídit, resp. programovat celou počítačovou síť.



Obrázek 2.3: Schéma SDN, převzato z [12]

Obrázek č. 2.3 ukazuje jednoduché schéma softwarově definovaných sítí. Celou ar-

chitekturu lze tedy rozdělit do 3 logických vrstev, které spolu komunikují pomocí API.

- Aplikační vrstva - Na této úrovni se nachází samotné síťové aplikace jako jsou například DHCP, ACL, NAT, DNS a další. Jejich vytváření by mělo být poskytováno prostřednictvím nižší vrstvy, nazývané northbound API.
- Northbound APIs - Toto API využívají aplikace pro komunikaci s SDN controllerem.
- Control vrstva - V této vrstvě je centralizována veškerá logika, které dříve byla v síťových prvcích.
- Southbound APIs - Jedná se o skupinu API protokolů, které pracují mezi vrstvou infrastruktury a control vrstvou. Jejím hlavním úkolem je komunikace, která povoluje SDN controlleru instalovat na samotné síťové prvky rozhodnutí definované v aplikační vrstvě.
- Vrstva infrastruktura - Nejnižší vrstvou je samotný hardware pro předávání datagramů na fyzické úrovni. Pro funkčnost celé architektury je nutné, aby zde byla nasazena zařízení, která umí přijímat pokyny od control plane skrze southbound API.

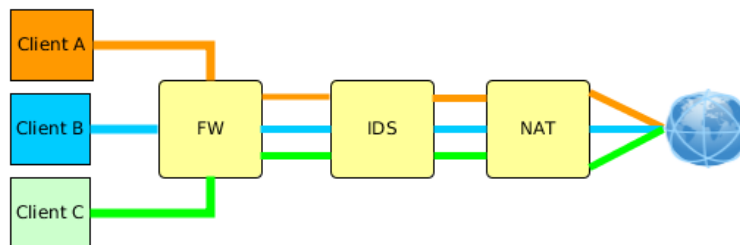
Přestože Softwarově definované sítě a virtualizace síťových funkcí jsou dvě různé technologie a koncepty, tak se navzájem se doplňují. Fakt, že SDN umožňuje programicky ovládat počítačovou síť, lze využít pro poskytnutí programovatelné konektivity mezi jednotlivými virtuálními síťovými funkcemi. Naopak SDN může využít NFV tím, že implementuje potřebné síťové funkce jako software. Může tak virtualizovat SDN Controller, který tak může běžet na co nejvhodnějším místě v datovém centru. Je vidět, že tyto dvě technologie se dobře doplňují, proto jsou často součástí jednoho řešení. [13]

2.4.1 Service Chaining

Jednou z výhod NFV je možnost využít Service Chaining. Service chaining je ve skutečnosti součástí SDN. Jde o princip jakým lze dynamicky pospojovat jednotlivé VNF a ovládat tak toky v síti. [13]

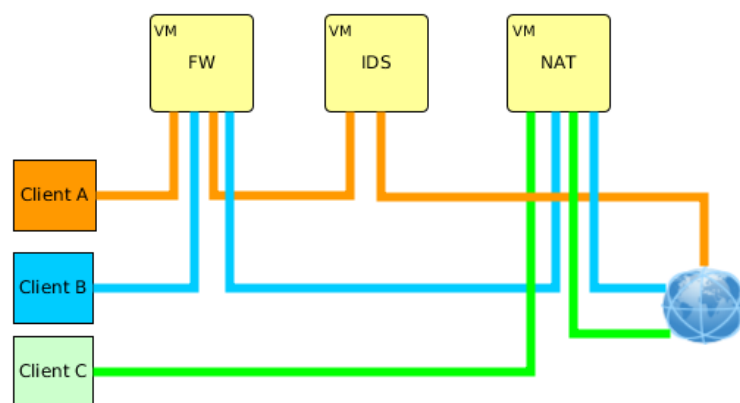
Service chaining není ve skutečnosti nic nového. V klasických počítačových sítích je používán také, ale pomocí fyzických síťových prvků. Jedná se zjednodušeně o způsob zapojení mezi jednotlivými síťovými prvky (či VNF) a způsob, jakým na sebe navazují. Příklad takového zapojení je vidět na obrázku č. 2.4. Zde se provozovatel sítě rozhodl, že odchozí data z klientských stanic musí jít přes firewall, IDS a nakonec přes NAT do Internetu. Příchozí data mají logicky obrácené pořadí. Toto zapojení funguje dobře pro

síť, kde není třeba rozlišovat cestu jakou proudí data jednotlivých uživatelských stanic. Ale není to optimální řešení pro síť s více uživateli, kde každý požaduje jinou síťovou funkci. Potřeba jednotlivých síťových služeb se samozřejmě může v čase měnit. Příklad takové sítě lze nalézt ve většině datových center.



Obrázek 2.4: Ukázka klasického service chainigu pomocí fyzických síťových prvků

Zde tedy přichází na řadu VNF spolu s SDN. Protože jednotlivé VNF existují jako virtuální stroje, tak mohou být dynamicky nasazovány dle aktuálních požadavků jednotlivých klientů a pomocí SDN mohou být tyto VM dynamicky pospojovány. Obrázek č. 2.5 ukazuje schéma zapojení, kde každý klient může mít jinou požadovanou cestu do internetu. Je možná i varianta, kde každý klient má své vlastní VNF s jinou konfigurací.

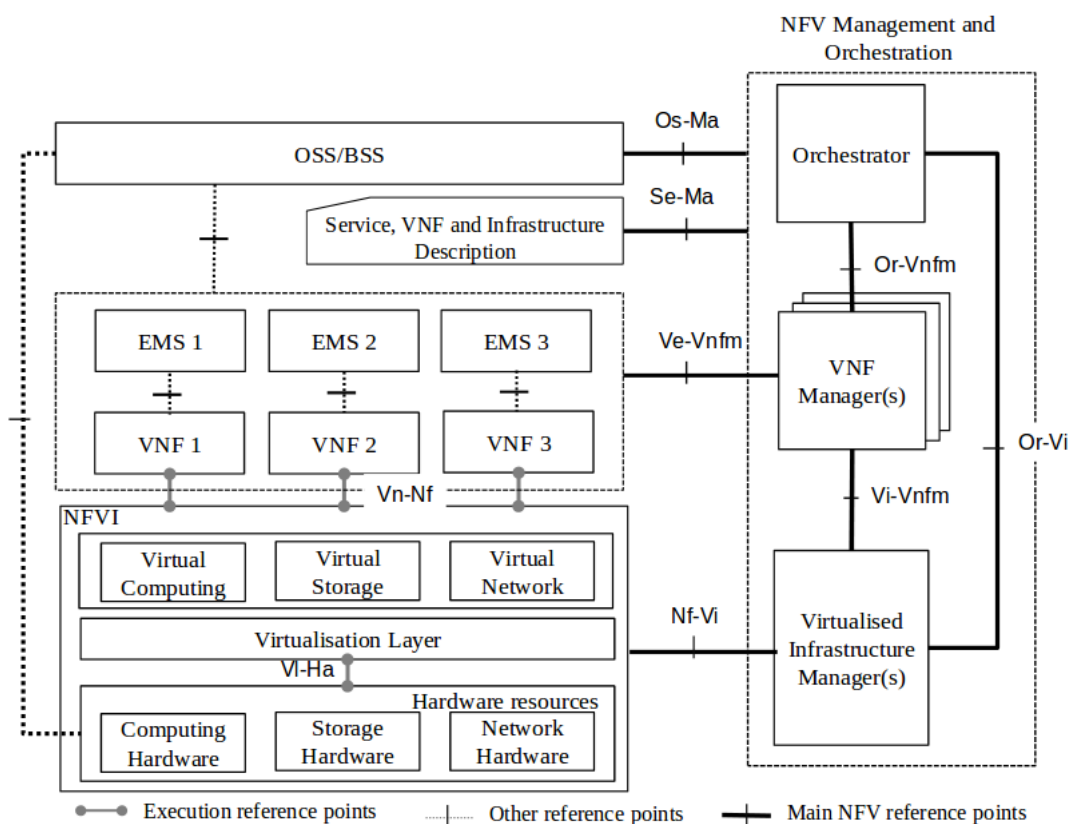


Obrázek 2.5: Ukázka VNF service chainigu

2.5 Architektura NFV a VNF

V předchozí sekci byla popsána myšlenka a motivace související s virtualizací síťových funkcí. Protože cílem této práce je navržení jednoduchého NFV frameworku, tak je nejprve nutné se seznámit s jeho obecnou architekturou. V této práci se bude vycházet z referenční architektury NVF [14], která byla navržena organizací ETSI. Jedná se

pouze o funkční návrh bez náznaků konkrétní implementace. Od této skupiny existují i podrobnější návrhy jednotlivých částí celého NFV frameworku, které v této práci budou také popsány v příslušných kapitolách.



Obrázek 2.6: NFV architektura, převzato z [14]

Jak je vidět na obrázku č. 2.6, tak celá architektura se dá rozdělit na tyto 3 hlavní části:

- **Infrastruktura virtualizace síťových funkcí (NFVI)** - Jsou všechny softwarové a hardwarové zdroje potřebné k vytvoření prostředí, ve které mohou být jednotlivé VNF být nasazeny. Tato infrastruktura může být velice rozsáhlá, proto je její součástí i síť poskytující konektivitu mezi vzdálenými lokacemi infrastruktury.[15]
- **Virtualizované síťové funkce (VNFs)** - Jsou softwarové implementace síťových funkcí, jako je např. NAT a routing, které mohou být nasazeny na NFV infrastruktuře.
- **Management a orchestrace NFV (NFV-MANO)** - zde se jedná o řízení softwarových a hardwarových zdrojů v celé infrastruktuře NFV a životního cyklu jednotlivých virtuálních síťových funkcí. Tato část se tedy zaměřuje na řízení a správu

všech úloh související v virtualizaci v NFV frameworku. [15]

Tyto funkční bloky se ještě dále dělí, proto dále v této práci budou tyto jednotlivé části popsány podrobněji a současně k nim budou uvedeny různé možnosti řešení.

2.5.1 Infrastruktura NFV

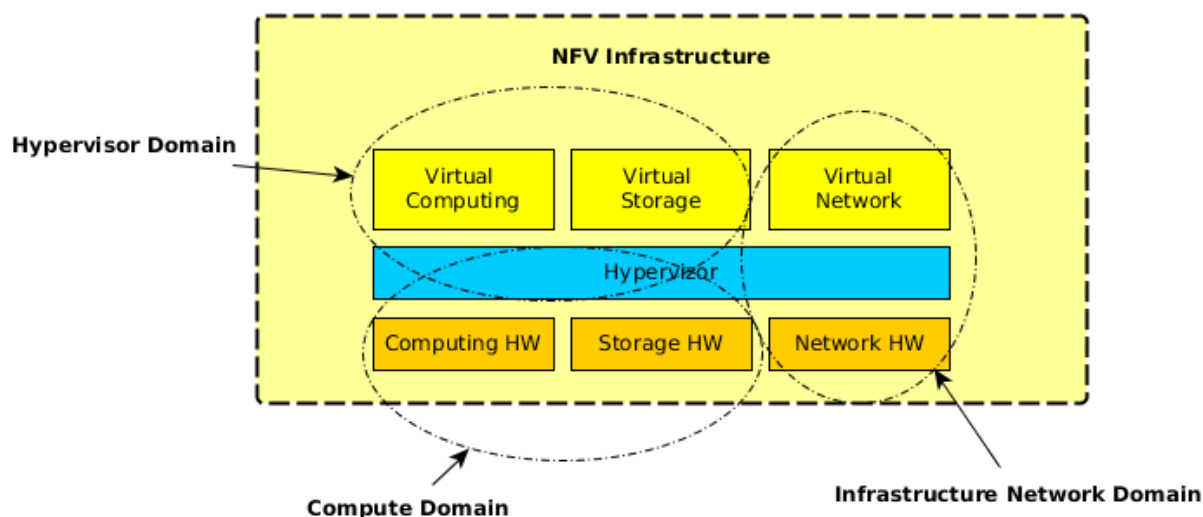
Ve zdroji [16], který detailně popisuje infrastrukturu pro virtualizaci síťových funkcí (NFVI), je uvedeno, že je v ní sdružení všech základních zdrojů potřebných pro běh virtuálních síťových funkcí (VNF). Z tohoto důvodu sem patří veškerý hardware. Do NFVI také patří některé softwarové komponenty, které jsou společné mnoha VNF a poskytují funkcionalitu potřebnou pro podporu nasazení, propojení či managementu VNF. Celou infrastrukturu může tvořit jeden či více strojů, které mají tyto potřebné funkce. Tyto stroje také mohou být umístěny v různých spolu spojených lokacích.

Pro zjednodušení lze celou NFV infrastrukturu rozdělit do 3 následujících domén:

- Compute Domain - Do této domény patří veškeré hardwarové zdroje jako jsou servery, úložiště a komponenty, které tyto zdroje obsahují, např. procesory, pevné disky, síťové karty, atd. Zároveň je zde řešen návrh fyzické topologie. [17]
- Hypervisor Domain - Toto je doména, které představuje softwarové prostředí abstrahující hardware v compute doméně a poskytuje je jako virtuální zdroje. Tyto zdroje následně mohou využívat virtuální síťové funkce. [18]
- Infrastructure Network Domain - V této doméně je řešeno veškeré propojení výše zmíněných domén. Tedy fyzické i virtuální infrastruktury. [19]

Funkci obsaženou v jednotlivých doménách znázorňuje obrázek č. 2.7. Více informací na tuto problematiku lze nalézt v [16] a ve zdrojích uvedených u každé domény.

Dá se říci, že referenční návrh infrastruktury pro NFV je podobný jako pro návrh infrastruktury pro cloud computing platformu. Měl by se tedy skládat z generických a komerčně vysoce dostupných serverů, které by měli být zapojeny do switchu a tím by měla být zajištěna konektivita. Na tyto servery je následně nasazen jeden z dostupných hypervisorů. Výběr správného hypervisoru, které jsou v současné době dostupné na trhu, je hlavní podmínka správného a funkčního návrhu této části NFV frameworku. Přehled hypervisorů je uveden v kapitole 2.6.1. V produkčním prostředí by součástí řešení bylo samozřejmě řešení síťového návrhu. Tato práce však má sloužit pouze jako ukázka a z tohoto důvodu zde nebude síťový návrh zmíněn.



Obrázek 2.7: Schéma NFV infrastruktury

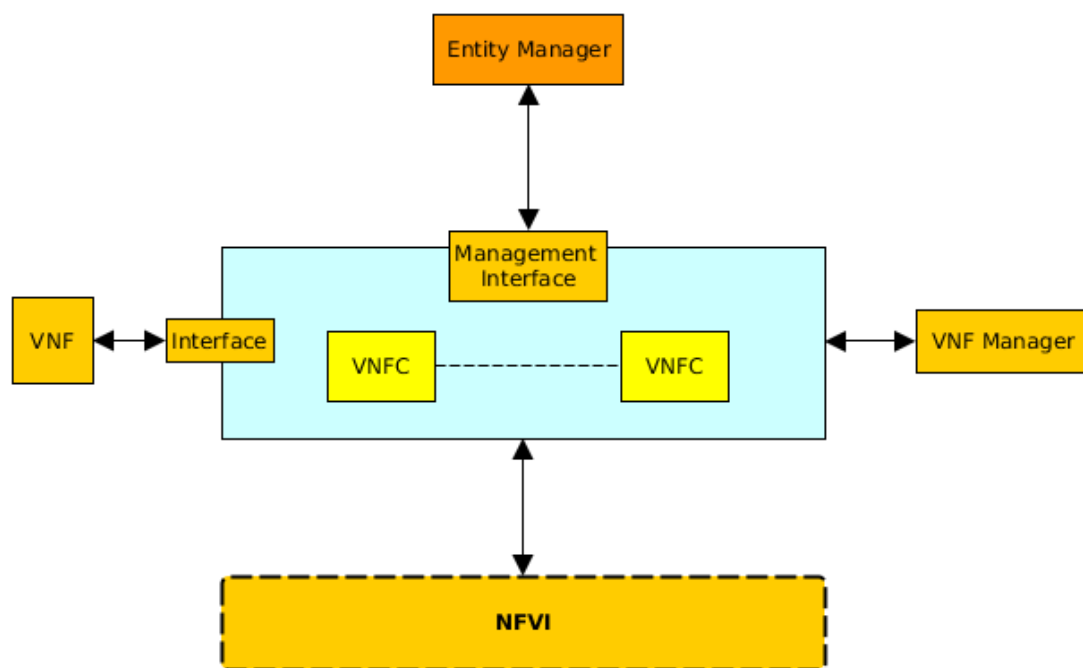
2.5.2 Virtuální síťová funkce

Virtuální síťová funkce (VNF) je dle [20] určitá síťová funkce, která běží na NFV infrastruktuře a je zároveň NFV frameworkem řízena a spravována. Zároveň musí mít dobře definované rozhraní k ostatním síťovým funkcím, k VNF Managerovi a měla by obsahovat management rozhraní či port. Jedna VNF může být obsažena v jednom virtuálním stroji nebo může být roztažena přes více virtuálních strojů.

Na obrázku č. 2.8 je vidět jednoduché schéma virtuální síťové funkce dle referenčního návrhu [20]. Celý životní cyklus VNF, což je vytvoření, spuštění, zastavení, smazání a škálování, řídí VNF Manager, který je součástí NFV managementu a orchestrace. Současně je možné dynamicky změnit aktuální konfiguraci pomocí Entity manageru (EM) přes management interface. EM může spravovat více VNF nebo právě jednu. Vnitřní struktura celé instance může být tvořena více komponentami (VNFC), které spolu mohou být navzájem provázány. Toto provázání však nemusí být viditelné zvenčí.

Pohledem na současný trh zjistíme, že VNF je prakticky poskytována ve 3 základních podobách.

- Softwarová aplikace - V tomto případě je poskytována VNF jako aplikace, která může být nainstalována na běžný operační systém jako je například GNU/Linux.
- Ucelený operační systém - Zde je poskytován přímo celý operační systém, který může být nainstalován do virtuálního stroje nebo i na fyzický server.
- Kompletní VM - Poskytovatel VNF může dát k dispozici rovnou přetvořený ob-



Obrázek 2.8: Schéma virtuální síťové funkce

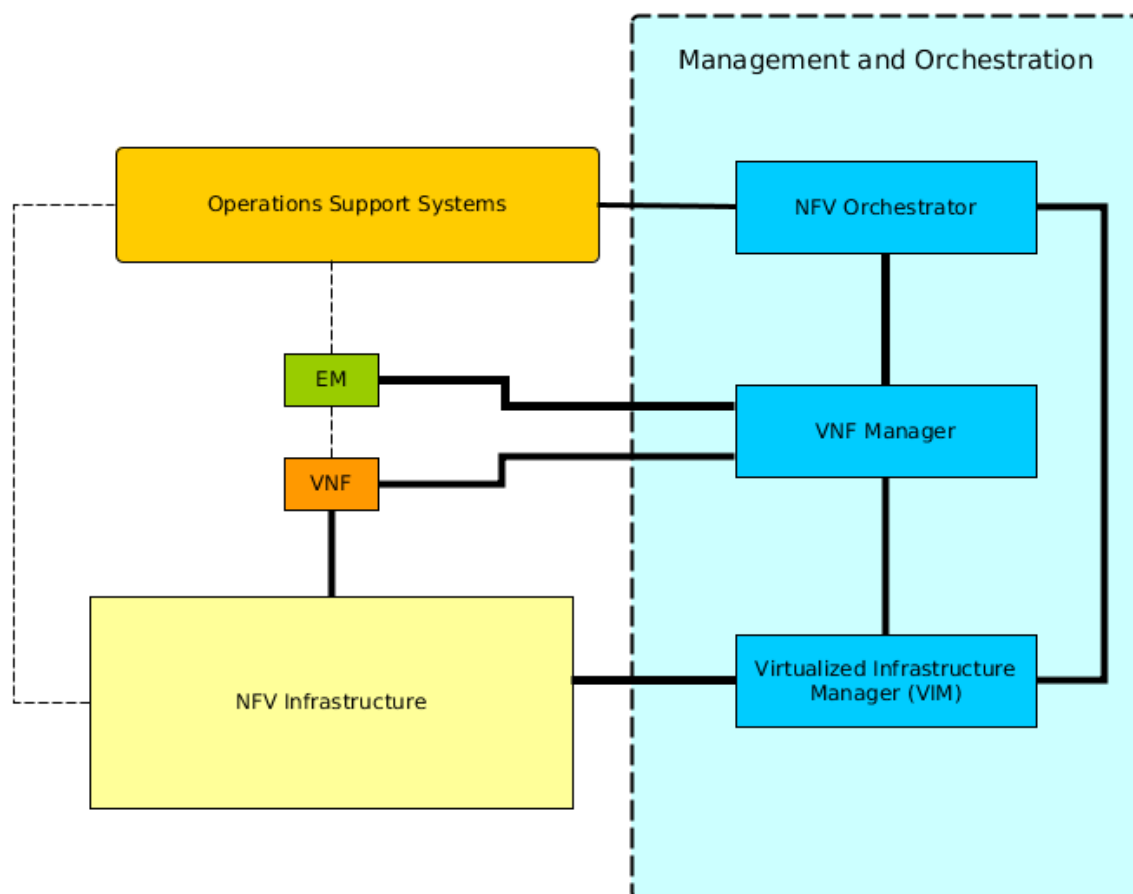
raz virtuálního stroje (image), který může obsahovat operační systém se síťovými funkcemi. Tento systém však nemusí být klasicky dostupný operační systém jako je GNU/Linux či FreeBSD, ale může se jednat o speciálně vytvořený systém od výrobce. Tento způsob budou využívat poskytovatelé, kteří mají proprietární řešení pro síťová řešení jako je například Cisco či Juniper.

2.5.3 Management a orchestrace NFV

Management a orchestrace virtualizace síťových funkcí (NFV MANO) je nejdůležitější část celého NFV frameworku. Je tomu tak, protože MANO zajišťuje správné fungování NFV infrastruktury i jednotlivých virtuálních síťových funkcí. MANO také poskytuje funkce nutné pro provisioning VNF a související operace, jako je jejich konfigurace jednotlivých VNF a infrastruktury, na které běží. Zároveň spravuje a řídí životní cyklus fyzických a virtuálních zdrojů, které slouží pro podporu VNF.

Jak vyplývá z obrázku č. 2.9, tak referenční návrh MANO dle [21] se skládá ze hlavních 3 částí, které se zabývají správnou jednotlivých vrstev NFV frameworku.

- Virtualized infrastructure manager (VIM) - Řídí a spravuje fyzické a virtuální zdroje v jedné doméně infrastruktury. Celková infrastruktura se může skládat z více domén a každá musí mít svůj VIM. Jeho typickými úlohami jsou vytváření,



Obrázek 2.9: Schéma NFV MANO

udržování a uvolňování VM na dostupných zdrojích v doméně. Zároveň musí mít přehled o všech těchto a stavu hardwarových zdrojů.

- VNF manager - Dohlíží na lifecycle management jednotlivých VNF instancí. To znamená, že vytváří, udržuje a ukončuje VNF instance, které běží na jednotlivých VM (ty však spravuje VIM). Opět může existovat více VNF managerů, kteří mohou spravovat jednu či více VNF.
- NFV orchestrator - Zjednodušeně slouží jako řízení a správu všech VIM a všech VNF managerů. Pomocí komunikace s VIM dokáže spravovat dostupné zdroje a pomocí komunikace s VNF managery dokáže řídit síťové služby. Jeho další funkcí je i přehled všech dostupných VNF, neboli katalog VNF, a registrace nových VNF do tohoto katalogu. Ten je pak dostupný uživatelům.

Celý systém je navržen tak, že by měl pracovat společně se stávajícími aplikacemi a

systémy, které potencionální uživatelé používají pro provoz své infrastruktury a podnikových procesů (Operation support system).

V oblasti NFV MANO probíhá v současnosti rozsáhlý vývoj a existuje několik projektů, které se tím zabývají. V článku [22] je nabídnut zajímavý přehled.

//TODO TOSCA

2.5.3.1 TOSCA

2.6 Možné technologie pro řešení

V předchozí části byla popsána referenční architektura, kterou navrhla ETSI. V té jsou specifikovány funkční požadavky a nastíněny potřebná rozhraní. Přesto lze tento návrh považovat za poněkud omezený v rozsahu. Není v něm například definována řízení a správa starších zařízení, což může velice zkomplikovat provoz síťové infrastruktury, která se skládá z VNF i těchto starších zařízení. Kromě toho standardy a referenční implementace VNF, infrastruktury, a MANO prozatím nejsou k dispozici.

Z tohoto důvodu následuje návrh možnosti pro každou z oblastí architektury, které v současnosti mohou sloužit jako její řešení.

2.6.1 Hypervisory

hypervisor je základná součástí cloudové platformy a frameworku pro virtualizaci síťových funkcí. Na trhu již existuje celá řada různých hypervisorů. Zde je uveden stručný přehled těch nejpoužívanějších.

- XEN - Je to hypervisor prvního typu, který pracuje na nejnižší vrstvě. Tato vrstva podporuje jeden nebo více hostovaných operačních systémů. První hostovaný systém se nazývá doménou 0 a slouží k přímému přístupu k hardwaru a jeho management. Do tohoto systému následně je možné přidávat další uživatelské domény, které mohou být linuxové systémy či Microsoft Windows.
- KVM - Jedná se o virtualizaci založenou na linuxovém jádru. Každá virtuální instance má svůj vlastní virtualizovaný hardware včetně síťové karty, disku a grafické karty. Tento typ hypervisoru vyžaduje pro správnou funkci procesor s rozšířením pro virtualizaci hardwaru.
- Microsoft Hyper-V - Zde se jedná o hypervisor od společnosti Microsoft, který lze nalézt v Windows Serverech od verze 2008. Hyper-V je hypervisorově stavěný serverový systém. To znamená, že má svůj hlavní operační systém a pomocí virtualizace se skrze něj mohou spustit další operační systémy.

- **VMware ESXi** - Varianta ESXi je odlehčená verze ESX klienta, která dovoluje běžet hostitelský systém na výměnném zařízení. Jde o tenký klient s vlastním linuxovým jádrem, který běží přímo nad hardwarovou vrstvou. Kernel ESXi funguje jako hostitelský operační systém pro vrstvení dalších služeb. Kernel na sebe váže modul vmkernel s dalšími obslužnými funkcemi, který tvoří základní stavební kámen celého řešení. Výhodou tohoto řešení je možnost alokace co největšího množství hardwarových prostředků pro hostované systémy. Tenký klient totiž zbytečně nevyužívá systémové prostředky hostujícího serveru.

2.6.2 VNF

Pokud se podíváme na trh s VNF u některých vendorů, tak zjistíme, že mnozí poskytují virtuální instance, které se dají použít pro účely VNF v této práci. Tato práce je zaměřena především na funkce firewallu a proto zde jsou uvedeny příklady pouze pro ně. Uvedeny jsou hlavně produkty největších a nejpoužívanějších poskytovatelů síťových prvků a také open-source firewall.

- **Juniper vSRX** - Jde o firewall od společnosti Juniper, který je obdobou jejich fyzického zařízení Juniper SRX. Jde o virtuální instanci poskytující funkce pro firewall, routing a pokročilé bezpečnostní funkce pro poskytovatele telekomunikačních služeb a větší společnosti. Toto VM je určené pro privátní, public i hybrid cloud.
- **Fortigate-VM** - Fortigate Virtual Appliances je řešení pro cloudové prostředí od společnosti Fortinet. Nabízí stejné funkce pro firewall jako jsou obsaženy ve Fortigate fyzických zařízeních.
- **Cisco ASAv** - Společnost Cisco nabízí Adaptive Security Virtual Appliance (ASAv), která obsahuje stejný software jako fyzické ASA zařízení a většinu funkcí pro firewall, routing a VPN.
- **PFSense** - PFSense je open-source projekt, který má za cíl poskytnout firewall postavený na operačním systému FreeBSD, který může běžet na klasické architektuře jednodeskových počítačů. Toto řešení poskytuje všechny důležité vlastnosti komerčních firewallů, má jednoduché ovládání a je to otevřené řešení.

2.6.3 Cloud platforma

Pro účely vytvoření infrastruktury, a vůbec možnost využití NFV v datovém centru, je nutná cloudová platforma. Existují několik řešení, které lze pro tyto účely použít. Dvě z nejčastějších jsou OpenStack a VMware vCloud.

2.6.3.1 OpenStack

OpenStack je open-source platformou umožňující postavit IaaS cloud, který může být nainstalován i na běžném hardwaru. Toto řešení má za cíl vytvořit dostupnou cloudovou platformu, která bude splňovat všechny potřeby privátních a veřejných cloudů nezávisle na velikosti řešení. [23]

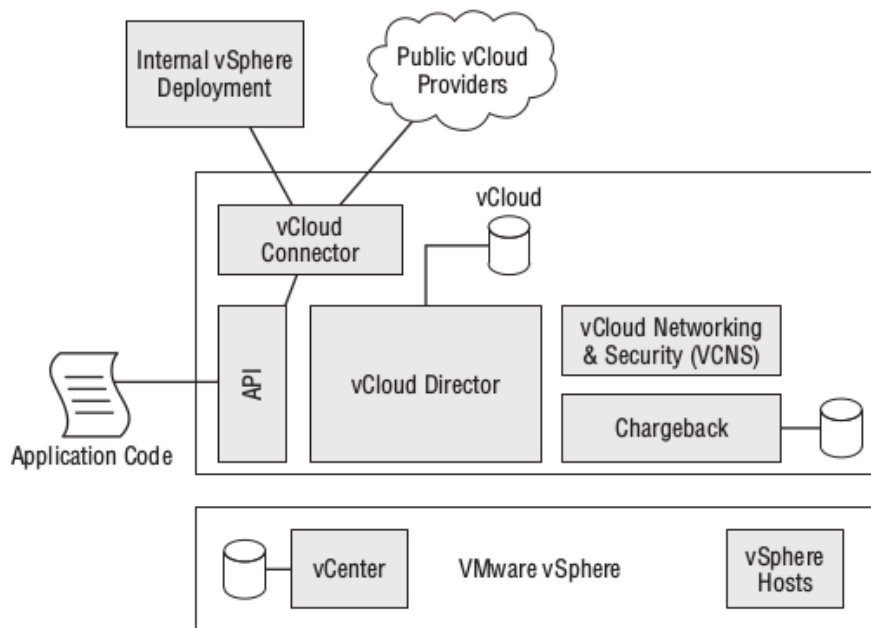
Celá stavba systému OpenStack se skládá z několika na sobě nezávislých projektů (modulů), které řeší různé oblasti cloudové platformy. Tyto projekty mezi sebou komunikují pomocí otevřených API a mohou být spravovány pomocí dashboardu. Celé administrace OpenStacku může být prováděna přes webové rozhraní, příkazovou řádku či přímo pomocí příkazů zaslaných do API. Celé toto řešení se vyznačuje jednoduchostí implementace, škálovatelností a rychlým vývojem nových vylepšení. Hlavními moduly OpenStacku jsou [23] [24]:

- Keystone - identifikační služba používaná OpenStackem pro autorizaci a autentizaci. Ověřování probíhá pomocí tokenů. Uživatel přihlášením odesílá žádost na Keystone, který tento modul zpracuje, zjistí pověření a vytvoří token. Vytvořený token je poté odesílán s žádostí do ostatních služeb. Zde dojde ke komparaci tokenu se současnou přístupovou politikou a dojde ke zjištění, zdali má uživatel dostatečná oprávnění pro provedení požadovaného úkonu.
- Glance - služba umožňující práci s virtuálními diskovými obrazy (imagy). Tyto obrazy mohou být uloženy na mnoha různých místech od lokálních systémových disků až po distribuované souborové systémy, jako je OpenStack Storage.
- Nova - tento modul poskytuje výpočetní služby. Umožňuje tedy běh několika instancí virtuálních strojů na několika hostitelských strojích, na kterých je nainstalována služba OpenStack compute. OpenStack podporuje hypervizory KVM, QEMU, VMware ESX, Hyper-V, Xen.
- Neutron - je služba pro správu všech síťových aspektů OpenStacku. Jedná se tedy o SDN komponentu. Neutron podporuje možnost rozšíření o tzv. pluginy, které umožňují využívat řešení třetích stran pro síťování.
- Cinder - poskytuje infrastrukturu pro mapování volumů v OpenStacku.
- Ceilometer - služba, která sbírá měřená data a monitoruje tak využívané zdroje.
- Heat - umožňuje automatizovanou orchestraci virtuálních strojů na základě vytvořených šablon.

- Horizon - představuje dashboard, který umožňuje cloudovým administrátorům a uživatelům spravovat různé zdroje a služby OpenStacku. Dashboard umožňuje interakci s OpenStackovým kontrolerem prostřednictvím API.

2.6.3.2 VMware vCloud

Společnost VMware poskytuje pro privátní cloudové systémy své řešení, které označuje jako VMware vCloud. Toto řešení je poskládané z jednotlivých produktů této společnosti. Celá architektura má hierarchický model, který je vidět na obrázku č. 2.10



Obrázek 2.10: Schéma VMware vCloud, převzato z [25]

Dle [25] se VMware vCloud skládá především z těchto komponent.

- VMware vCloud Director - je jedna ze základních součástí potřebných pro vytvoření privátního cloudu ve stylu VMwaru. Umožňuje vytvořit a doručovat koncovým zákazníkům infrastrukturu jako službu. Je propojen a přímo spolupracuje s VMware vSphere center.
- VMware vSphere - tento produkt slouží pro vytvoření virtualizované infrastruktury. Je to sdružení více komponent. Ty nejhlavnější jsou:
 - VMware ESX (ESXi) - hypervisor, který byl popsán v kapitole 2.6.1.
 - VMware vCenter Server - umožňuje efektivní a pokročilejší správu virtualizovaného prostředí, bez ohledu na jeho velikost. Jedná se např.o snadné

vytváření nových virtuálních počítačů, jejich klonování nebo importování z jiného úložiště.

- VMware vSphere Client - je určený pro dálkovou správu hostitelů ESXi. Připojit se můžeme prostřednictvím vCenter serveru nebo přímo přes ESXi server.
- VMware vCloud Networking and Security - poskytuje síťování a bezpečnost pro virtuální prostředí. Poskytuje mnoho síťových funkcí a poskytuje framework pro integraci řešitelů třetích stran.
- VMware vShield - představuje možnost zabezpečení v prostředí VMware VSphere. Může být konfigurován pomocí vShield Managera, který umožňuje centrální správu přes webové rozhraní, vSphere klient plug-in, nebo command line interface (CLI).
- VMware vCenter Chargeback - slouží pro monitorování virtuálních strojů, které následně může být účtováno.

2.6.4 SDN

Součástí řešení pro datové centrum je dnes i SDN. I zde existuje několik možností. Dvě z nejpoužívanější řešení jsou:

2.6.4.1 OpenContrail

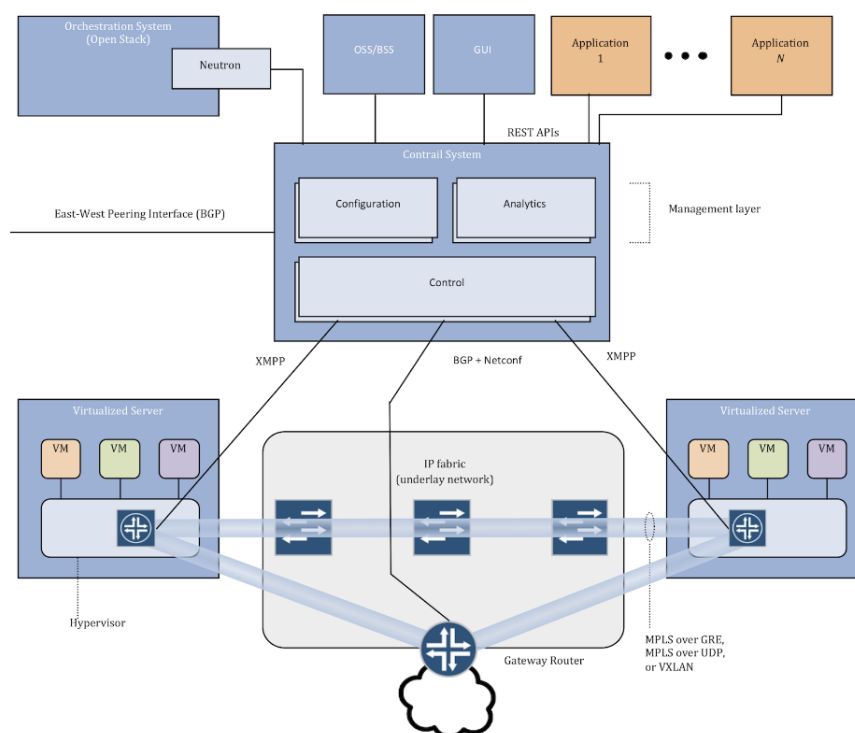
OpenContrail je systém, který může být použit v mnoha síťových scénářích jako například v cloud networkingu nebo v sítích poskytovatele síťových služeb. V privátním cloudu, ve Virtual Private Cloud (VPC) a v IaaS se vyskytuje prostředí s velkým množstvím tenantů, kde několik tenantů sdílí stejné fyzické zdroje (server, úložiště, fyzickou síť). Každý tenant má přiřazený vlastní logické zdroje (virtuální stroje, virtuální úložiště, virtuální síť). Tyto logické zdroje různých tenantů musí být od sebe odděleny. Virtuální síť v datacentrech mohou být také spojeny s fyzickou IP VPN nebo L2 VPN. [26]

OpenContrail je složen ze dvou hlavních komponent. První z nich je Controller, který je logicky centralizovaný, ale fyzicky distribuovaný. To znamená, že je složen z několika typů rolí a každý z nich má několik instancí z důvodu vysoké dostupnosti a horizontální škálovatelnosti. Tyto role mohou být fyzické servery nebo virtuální stroje. Tyto role jsou [27]:

- Configuration role - poskytuje north-bound REST API. Toto API může být použito pro konfiguraci systému nebo pro extrahování operačního stavu systému.

- Control role - implementuje logicky centralizovanou část control planu.
- Analytics role - je zodpovědný za sběr, porovnání a prezentaci analytických informací.

Další komponentou je vRouter, který má na starost přenos dat. VRouter běží na virtualizovaném serveru, na kterém běží hypervizor. Rozšiřuje fyzickou síť v datovém centru o virtuální overlay síť hostovanou ve virtualizovaných serverech. VRouter narozdíl od vSwitchů poskytuje směrování a služby vyšších vrstev. Data plane, tedy vRoutery mezi sebou, může používat různé technologie overlay jako MPLS over GRE, MPLS over UDP a VXLAN. Control plane protocol mezi Controllerem a fyzickým gateway routerem (nebo switchem) je BGP. Protokol používaný mezi Controllerem a vRoutery se nazývá XMPP. Obrázek č. 2.11 ukazuje schéma OpenContrailu.



Obrázek 2.11: Schéma OpenContrail, převzato z [26]

2.6.4.2 VMware NSX

VMware NSX představuje technologii síťové virtualizace od společnosti VMware. NSX umožňuje vytváření a správu softwarových virtuálních sítí napříč celým datovým centrem. Architektura NSX je složená z Consumption, Management plane, Control plane a Data plane. [28]

- Data plane - je představován virtuálními switchi, jako je NSX vSwitch či Open vSwitch.
- Control role - zde je komponentou NSX controller či jejich cluster. NSX controller přijímá instrukce, na konfiguraci logických sítí od tenantů, prostřednictvím NSX API. Poté pomocí OpenFlow konfiguruje vSwitche.
- Management plane a Consumption - je tvořen NSX managerem, který poskytuje centralizovanou možnost konfigurace a REST API. [28]

U tohoto řešení existují dva pohledy na síťovou infrastrukturu. První je transportní, který představuje underlay síť. Ta se skládá z síťový hardware propojující hypervizory. Druhý pohled je logický. Ten představuje množinu síťových služeb, které vidí každý tenant či virtualizovaný stroj v tomto tenantu. V multi-tenant prostředí má každý tenant svůj vlastní logický pohled na svoji síť a nemůže vidět síť ostatních tenantů. [28]

VMware NSX je dostupný ve dvou verzích - NSX-v a NSX-mh. NSX-v je verze závislá na proprietárním VMware řešení, tedy spolupracuje pouze s hypervizorem vSphere od VMware. Druhá verze je NSX-mh, kde mh znamená multi hypervizor a je tedy možné použití s různými hypervizory.

3 Tvorba VNF v cloudovém prostředí

V předešlé kapitole byla vysvětlena základní problematika, která souvisí s virtualizací síťových funkcí, cloud computingem a softwarově definovanými sítěmi. Zároveň byla popsána referenční architektura frameworku pro virtualizaci síťových funkcí. Tato kapitola bude již věnována konkrétnímu příkladu využití virtuálních síťových funkcí v cloudovém prostředí. Nejprve zde popsána navržená architektura pro privátní cloudovou platformu využívající virtualizaci síťových funkcí, kterou mohou využívat všichni její uživatelé. Pro tuto cloudovou platformu a pro její uživatele byli navrženy dva příklady virtuálních síťových funkcí. U obou příkladů jsou uvedeny scénáře a způsob jakým jsou navrženy.

//TODO Požadavky od TCP cloud //Proč jsem si vybral OpenStack??

3.1 Požadavky na NFV řešení

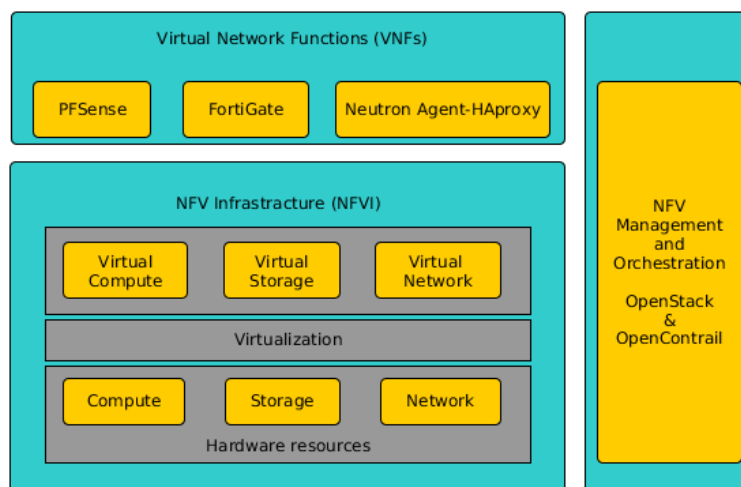
Při navrhování ukázek řešení VNF v cloudovém prostředí byli definovány některé parametry které dané návrhy museli splňovat.

- Univerzálnost - Celé řešení musí být postavené tak, aby ho mohli využívat všichni uživatelé dané cloudové platformy.
- Jednoduchost - Vytvoření a správa VNF musí být pro uživatele jednoduchá a rychlá.
- Otevřenost a Flexibilita - Vytvořené VNF by mělo být možné v případě potřeby snadno upravit uživatelem. Úpravou by měla vzniknout nová verze VNF, kterou mohou využívat další uživatelé.
- Kompatibilita se stávající síťovými prvky - Všechny VNF by měly být schopné komunikovat se všemi stávajícími fyzickými síťovými prvky. V tomto případě by měla být schopno provázat celý NFV framework se stávající infrastrukturou.

3.2 Architektura navrženého řešení

Architektura navrženého řešení byla implementována pomocí cloudové platformy OpenStack a SDN řešení OpenContrail. Obrázku č. 3.1 znázorňuje tyto technologie v

souvislosti s referenční architekturou popsanou v kapitole 2.5. Je nutné říci, že obě technologie nezapadají přímo do jedné z částí referenční architektury. Naopak v některých případech se překrývají nebo se v ní doplňují.



Obrázek 3.1: Architektura NFV řešení

OpenStack byl zvolen, protože se jedná o největší open-source cloudovou platformu na světě. OpenStack tvoří část správy infrastruktury. Hardwarová vrstva infrastruktury se může skládat z libovolných serverů, na kterých je nainstalován KVM hypervizor. Tento hypervizor tvoří virtuální vrstvu a byl vybrán, protože je nejčastěji používán společně s OpenStackem. Avšak v případě potřeby by zde mohl být použit i jiný hypervizor, pokud bude zachována kompatibilita vůči OpenStacku.

OpenStack spravuje převážně zdroje týkající se výpočetního výkonu (Compute) a uložště (Storage). Tyto zdroje následně přiděluje dle potřeby virtuálním instancím nebo v našem případě instancím, které slouží jako VNF. Bylo však nutné zvolit řešení, které se bude starat o síťování.

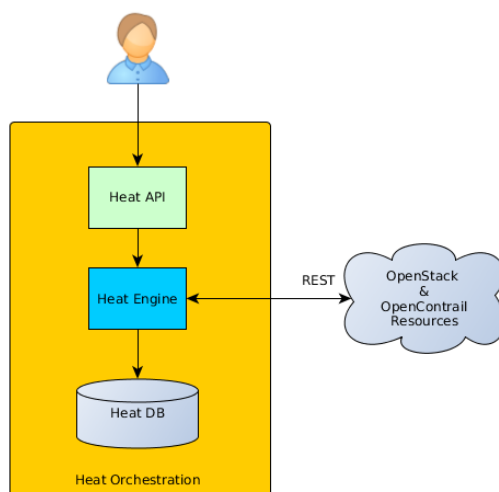
Speciálně pro vyřešení síťování v této infrastruktuře je součástí řešení OpenContrail. Díky tomu je možné vytvářet overlay sítě pomocí VXLAN či MPLS over GRE, kterými jsou dynamicky propojovány jednotlivé VM a VNF.

Jednotlivá VNF mohou být v OpenContrailu vytvořena pomocí tzv. Servisních Instance a Servisní Templatů. Ty budou v této práci použity pro vytvoření VNF sloužící jako firewally a budou podrobně popsány v kapitole věnující se vytváření této služby.

Další součástí, která musela být v architektuře navržena, je způsob řízení a správy jednotlivých VNF. Zde se muselo jednat o řešení, jakým automaticky vytvořit a popřípadě i smazat všechny potřebné části potřebné pro VNF. Pro tuto část byl zvolen Heat. Heat je část OpenStacku, která slouží pro automatickou orchestraci. Ten bude

v tomto návrhu zastávat roli VFN manažera, pomocí kterého budou jednotlivé VNF spravovány. Avšak dalo by se říci, že do této role spadá i OpenContrail, protože právě on umožňuje také spravovat jednotlivá VNF za běhu.

Heat je hlavní projekt v OpenStacku pro orchestraci. Umožňuje uživatelům popsat nasazení komplexních cloudových aplikací v jednom textovém souboru, který se nazývá Heat template. Tyto soubory se dají předat heat engine, který podle nich dokáže automaticky vytvořit požadované zdroje v OpenStacku i v OpenContrailu.



Obrázek 3.2: Popis heat orchestrace

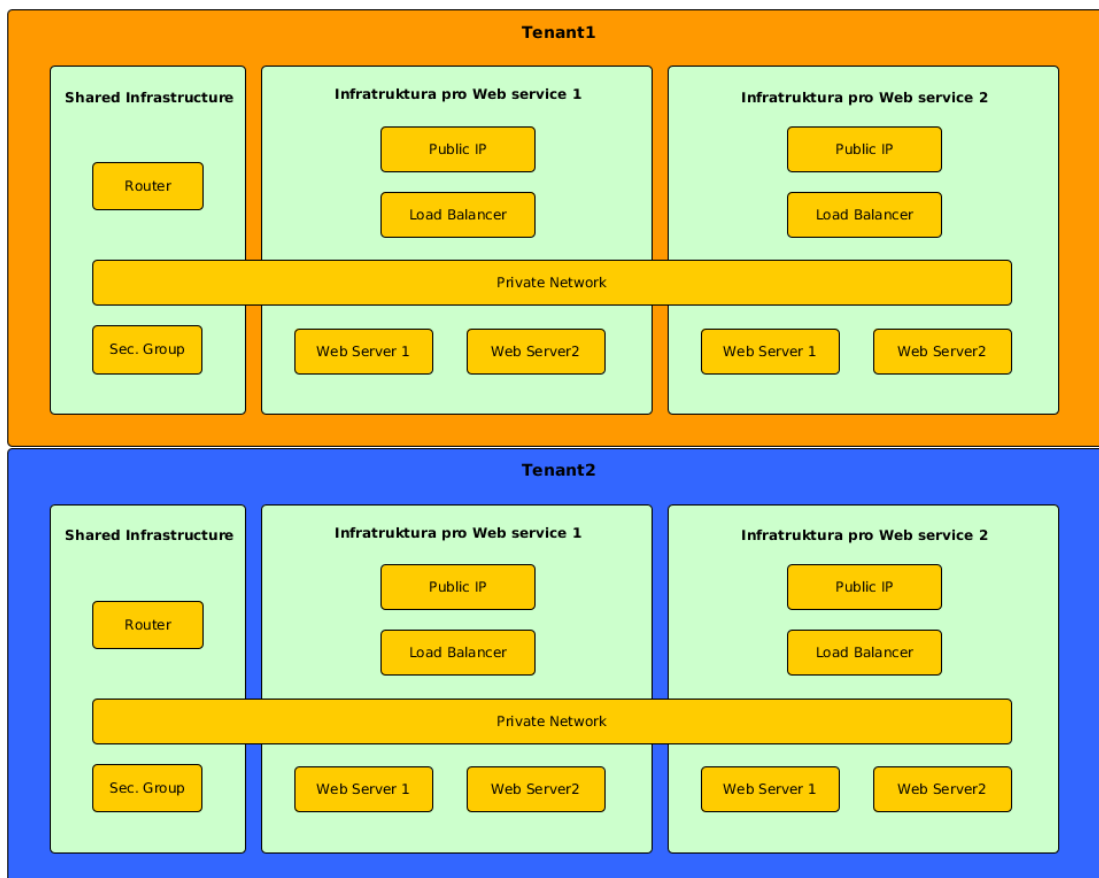
Z toho návrhu je patrné, že zde není implementovaný NFV orchestrator. Je to z důvodu toho, že pro účely řešení virtuálních síťových funkcí na cloudové platformě OpenStack s OpenContraiem, která je navržena v této práci, není tato část potřeba.

3.3 Load balancer as a Service

Jednou z nejčastějších síťových funkcí, která je v dnešních počítačových sítích a datových centrech vyžadována je load balancing. Load balancer spravuje příchozí komunikaci a distribuje ji do několika serverů či jiných síťových zdrojů. Tím je zajištěna rozloha zátěže. V cloudovém prostředí je tedy možné velice rychle, dle požadavků uživatele či automaticky dle nastavených parametrů, škálovat (přidávat či odstraňovat) webové servery. Load balancer zároveň monitoruje stav jednotlivých sleduje stav jednotlivých instancí a posílá komunika pouze správně fungujícím instancím.

3.3.1 Scénář použití LbaaS

Na obrázku č. 3.3 je vidět celý koncept load balanceru poskytovaného jako službu v privátním cloudu. Každý uživatel má možnost si dle potřeby vytvořit load balancer pro své webové servery. Pokud provozuje několik webových služeb v jednom projektu (tenantu), může pro každou tuto službu vytvořit vlastní load balancer, který bude nakonfigurován dle požadavků. Tento load balancer je dostupný pro všechny uživatele cloudu, tedy ve všech tenantech.



Obrázek 3.3: Load Balancer as a Service

3.3.2 Neutron LbaaS

Při výběru řešení pro load balancing je nutné zvážit především výkon load balanceru. Na trhu již existuje celá řada fyzických i virtuálních produktů. Fyzická řešení nabízí větší výkon a obvykle i více funkcí. Virtuální pak větší flexibilitu v nasazení a jednodušší konfiguraci. U obou však existuje jednotná množina funkcí, kterou uživa-

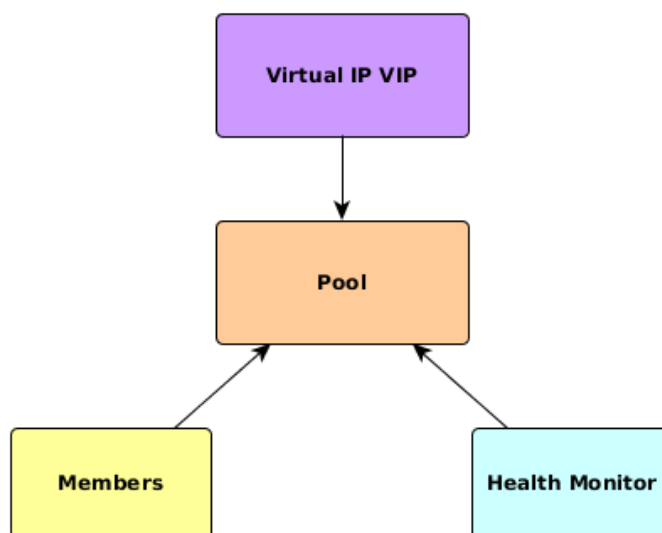
telé požadují a kterou většina load balancerů poskytuje.

OpenStack Neutron ve své implementaci obsahuje službu LBaaS. Je to jedna z jeho pokročilou služeb, která umožňuje použít jeden soubor API k ovládání load balanceru od poskytovatelů třetích stran. Jedinou podmínkou je, aby toto API implementovali. Toto velice zjednodušuje uživatelům OpenStacku ovládání load balancerů a odpadá díky tomu nutnost seznamování se s implementací a konfigurací těchto různých řešení, která mohou být velmi specifická a odlišná.

V této práci je ukázán příklad využití implementace load balanceru v OpenContralu, který může být přes toto api ovládán. Tento příklad je však univerzální a může být použit s jakoukoli implementací load balanceru, at už virtuálního (softwarového) či fyzického, pokud dokáže komunikovat s OpenStack Neutron LbaaS rozhraním. Dle dokumentace [29] je v OpenContrailu implementace load balanceru řešena pomocí HAProxy. HAProxy je zdarma dostupný open source software pro unix operační systémy [35].

Load balancer se v Neutron LbaaS skládá ze 4 objektů.

- Pool - Označuje síťový rozsah pro webové servery.
- Virtuální IP (VIP) - ip adresa, na kterou přichází komunikace
- Member - Označuje konkrétní instanci, která je členem poolu.
- Monitor - Monitoruje stav jednotlivých serverů a aplikací.



Obrázek 3.4: Neutron LbaaS

Obrázek č. 3.4 zachycuje jednotlivé závislosti mezi těmito objekty. Tato implementace má tyto hlavní funkce:

- Poskytuje Load balancing komunikace od klientů do poolu serverů. Load balancer zprostředkovává spojení prostřednictvím své VIP.
- Poskytuje load balancing pro HTTP, TCP a HTTPS komunikaci.
- Poskytuje možnosti pro monitorování aplikací. Prostřednictvím HTTP, TCP či PING. Zde celý proces tak, že se load balancer pokusí v určeném časovém intervalu navázat s daným serverem v pool spojení dle vybraného protokolu.
- Umožňuje asociaci floating ip (veřejné adresy) k VIP, čímž umožňuje přístup k serverů z veřejné sítě.

Celý proces probíhá tak, že každý virtuální server, který je asociovaný s daným poolem z něj obdrží IP adresu. Když přijde na VIP nějaký dotaz na danou webovou aplikaci, tak je tento dotaz předán dál na jednu z těchto přiřazeným IP adres. Pokud nastane s aplikací či serverem nějaký problém, který zachytí monitor, tak load balancer ip adresu tohoto serveru přestane posílat komunikaci, dokud není vše zase v pořádku. Výběr serveru může probíhat pomocí jedné z následujících metod:

- Round robin - zde se komunikace distribuuje rovnoměrně, resp. dle vah zadáných u jednotlivých memberů v poolu.
- Least connection - zde je vybrán member s nejméně spojeními.
- Source IP - u této metody je vybrán member na základě zdrojové ip adresy klienta.

3.3.3 LbaaS heat template

Aby nemusel uživatel ručně vytvářet load balancer ručně, tak byl celý proces vytváření load balanceru zautomatizován pomocí heat templatu. Navržený heat template pro LbaaS v sobě obsahuje několik prostředků, které se po jeho spuštění pokusí heat engine vytvořit. Celý template v sobě obsahuje i webové instance, které slouží pro testování. V produkci by však byly v odděleném templatu. Template je parametrizovaný a konkrétní hodnoty pro jednotlivé zdroje (ip adresy, ip) jsou v tzv. environment file, který se zadává při spuštění daného templatu. Dále jsou popsány pouze hlavní části heat templatu.

- privatni síť - k této síti jsou připojeny obě webové instance, load balancer a router. Součástí je definice toho zdroje jsou i subnet, který má dále parametry týkající se DHCP ip adres.

```

private_net:
  type: OS::Neutron::Net
  properties:
    admin_state_up: True
    name: { get_param: private_net_name }
    shared: False
private_subnet:
  type: OS::Neutron::Subnet
  properties:
    allocation_pools:
      - start: { get_param: private_net_pool_start }
        end: { get_param: private_net_pool_end }
    cidr: { get_param: private_net_cidr }
    enable_dhcp: True
    ip_version: 4
    name: { get_param: private_net_name }
    network_id: { get_resource: private_net }

```

Ukázka kódu 3.1: Privátní síť a subnet

- 2 x web instance - jedná se o virtuální instance s operačním systémem Ubuntu 14.04. Po spuštění heat templatu se na tyto instance nainstaluje Apache server a vytvoří se index.html. Díky tomu je možné následně otestovat zda load balancer distribuje komunikaci mezi těmito dvěma servery.

```

instance_01:
  type: OS::Nova::Server
  properties:
    image: { get_param: instance_image }
    flavor: { get_param: instance_flavor }
    key_name: { get_param: key_name }
    name: test-web01
    networks:
      - network: { get_resource: private_net }
    security_groups:
      - default
      - { get_resource: http_security_group }
    user_data_format: RAW
    user_data: |
      #!/bin/bash -v
      apt-get install apache2 -yy
      echo "Instance 01" > /var/www/html/index.html

```

Ukázka kódu 3.2: Web server 1

- router - toto je Neutron router implementující SNAT. V tomto příkladě je využíván webovými servery pro konektivitu k Internetu. Toto je nutné pro nainstalování programu Apache na webové servery.

```
router:
  type: OS::Neutron::Router
  properties:
    name: { get_param: router_name }
    external_gateway_info:
      network: { get_param: public_net_id }
```

Ukázka kódu 3.3: Web server 1

- public síť - toto je veřejná síť, ze které je získána VIP pro load balancer. Na tuto VIP bude dále asociována floating ip.

```
public_net:
  type: OS::Neutron::Net
  properties:
    admin_state_up: True
    name: { get_param: public_net_name }
    shared: False
public_subnet:
  type: OS::Neutron::Subnet
  properties:
    allocation_pools:
      - start: { get_param: public_net_pool_start }
        end: { get_param: public_net_pool_end }
    cidr: { get_param: public_net_cidr }
    enable_dhcp: True
    ip_version: 4
    name: { get_param: public_net_name }
    network_id: { get_resource: public_net }
lb_floating:
  type: OS::Neutron::FloatingIP
  properties:
    floating_network_id: {get_param: public_net_id}
    port_id: {get_attr: [lb_pool, vip, port_id]}
```

Ukázka kódu 3.4: Public síť a subnet

- pool - jedná se o definování poolu pro load balancer. Na ukázce je vidět, že byla zvolena metoda Round Robin. Tato metoda byla zvolena kvůli co nejjednoduššímu testování tohoto templatu.

```

lb_pool:
  type: OS::Neutron::Pool
  properties:
    admin_state_up: True
    lb_method: ROUND_ROBIN
    name: { get_param: lb_name }
    protocol: HTTP
    monitors:
      - { get_resource: lb_ping_health_monitor }
    subnet_id: { get_resource: private_subnet }
  vip:
    protocol_port: 80
    address: { get_param: public_net_ip }
    admin_state_up: True
    subnet: { get_resource: public_subnet }

```

Ukázka kódu 3.5: Load balancer pool

- **members** - po vytvoření instancí je nutné jejich přidání do poolu jako members. Pokud webová aplikace na serverch využívá jiný port než port 80, je možné ho zde změnit.

```

lb_pool_member_instance_01:
  type: OS::Neutron::PoolMember
  properties:
    address: { get_attr: [ instance_01 , first_address ] }
    admin_state_up: True
    pool_id: { get_resource: lb_pool }
    protocol_port: 80

```

Ukázka kódu 3.6: Members

- **health monitoring** - zdroj pro monitoring. Dle zvolených parametrů je vidět, že každých 5 sekund bude poslán ping na servery a bude se čekat 5 sekund na odpověď. Pokud nepřijde, tak load balancer usoudí, že je daný server není v pořádku a přestane na něj přeposílat komunikaci.

```

lb_ping_health_monitor:
  type: OS::Neutron::HealthMonitor
  properties:
    admin_state_up: True
    delay: 5
    max_retries: 1
    timeout: 5

```

```
type: PING
```

Ukázka kódu 3.7: Monitor

V celém heat templatu je ještě více zdrojů, které se vytváří. Ty zde však nebudou popsány. V případě zájmu lze nálezt kompletní heat template v příloze.

3.4 Firewall as a Service

Dalším častým příkladem VNF, kterou uživatelé cloudu mohou potřebovat je firewall. Oproti LbaaS je zde situace o něco komplikovanější. Je zde totiž více možností a jak daný firewall využívat. U tohoto příkladu služby bude uvedeno několik scénářů, které

3.4.1 Servisní instance v OpenContrailu

Servisní instance v OpenContrailu je jednoduše virtuální stroj, který poskytuje danou VNF. Úplně nejjednodušším příkladem může být virtuální stroj s operačním systémem GNU/Linux, který může sloužit jako router mezi dvěma sítěmi. Pro vytvoření takového virtuálního stroje jsou nutné 3 základní elementy.

- Service Template
- Service Instance
- Service Policy

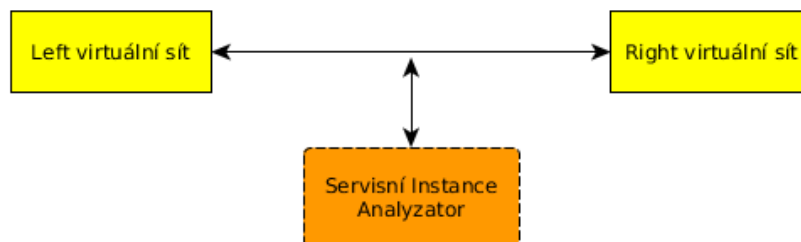
Servisní Template obsahuje obecný předpis pro danou VNF v OpenContrailu. Pro správné fungování je nutné zadat nastavit správné parametry patří:

- Název - Název je označení daného Servisního Templatu. Pomocí něho lze následně identifikovat daný template a spustit dle jeho parametrů Servisní instanci.
- Image - Je image, který má být použit pro vytvoření dané servisní instance. V našem případě se bude jednat o image, který obsahuje požadované síťové funkce. Tento image musí před tím než může být použit nahrán do OpenStacku Glance.
- Service Type - V OpenContrailu, prozatím existují dva typy. Jsou to Traffic Analyzer a Firewall.
- Service Mode - Zde se určuje v jakém modu daný template bude nastaven. Jsou zde 3 možnosti. , .

- Transparent - v tomto případě se jedná o neroutovaný firewall, neboli L2 firewall.
 - In-Network - poskytuje výchozí bránu a průchozí traffic je routovaný. Tento mode může být využit pro NAT, HTTP proxy, atd.
 - In-Network-NAT - zde je situace podobná jako u In-Network, ale navracající traffic nemusí být routovaný zpět do zdrojové sítě.
- Typy síťových portů - Zde se určuje kolik portů bude daná instance, vytvořená pomocí tohoto templatu mít a jaká bude jejich role. Jsou zde možnosti Left, Right a Management.

Po úspěšném vytvoření Servisního templatů je možné z něj vytvořit libovolný počet Servis Instancí. Ty běží jako klasické instance v OpenStacku. Jak tedy vyplývá z výše uvedených informací, tak existují dva druhy servisních instancí v OpenContrailu.

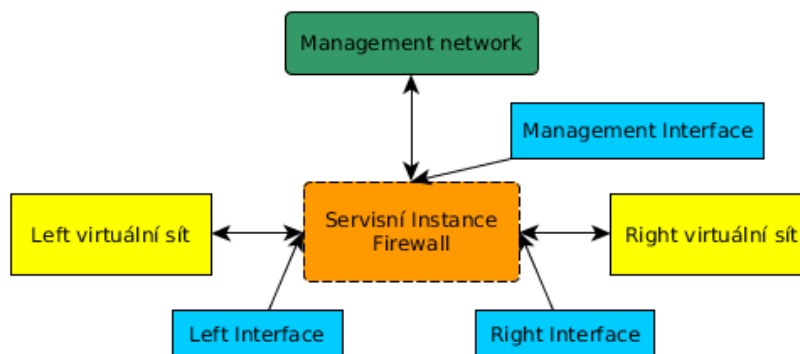
První z nich je Analyzer. Ten slouží k analýze a zachytávání síťového trafficu. Image pro tento typ servisní instance obvykle obsahuje protokolový analyzátor a paketový sniffer, jako je například oblíbený program Wireshark. Tato instance dostává traffic, který je poslán mezi dvěma sítěmi. Tento traffic vybírá OpenContrail podle nastaveného pravidla pro dané sítě. Podle těchto pravidel je vybrána jen část trafficu, která je následně dána k dispozici servisní instanci. Samotná servisní instance nijak nemanipuluje s trafficem a ani do něj žádný negeneruje. Jednoduše lze říci, že má nastavený síťový port v promiskuitním modu a pouze pozoruje traffic. Poté jen hlásí zachycené události uživateli či jiným entitám v síti. Obrázek č. 3.5 znázorňuje tento typ servisní instance.



Obrázek 3.5: Schéma zapojení servisní instance Analyzer

Druhým typem servisní instance je firewall. V tomto případě již servisní instance manipuluje s trafficem. Hlavní bodem při vytváření servisní instance jako firewall je přiřadit správné virtuální sítě k správným virtuálním síťovým portům. Servisní instance má obvykle dva síťové porty - left a right. Ty slouží pro propojení sítí do kterých jsou zapojeny. V některých případech je možné servisní instanci přidat třetí síťový port, který

slouží pro out-of-band management. Přestože některá řešení pro servisní instance sloužící jako firewall mohou mít již své požadované chování definované hned při jejich startu, tak tento port může být velice užitečný při konfiguraci dané servisní instance. A to ať už se jedná o konfiguraci manuální či pomocí nějaké vyšší management entity.



Obrázek 3.6: Schéma zapojení servisní instance

Service policy dovoluje síťový traffic mezi virtuálními sítěmi a říká systému, aby ho posílal skrze servisní instanci.

3.4.2 Scénář NAT

Nejčastěji servisní instance nasazována jako firewall, který bude sloužit pro NAT (Network address translation). V OpenContrailu je sice možnost využívat implementaci routeru s SNAT, která umožňuje instancím v privátních sítích konektivitu s externí sítí. Pokud však uživatel potřebuje využít pokročilejší funkce firewallu, tak je možné vytvořit servisní instanci, která bude sloužit jako VNF. V té může být použit libovolný požadovaný image firewallu uživatele.

V tomto případě má tedy každý uživatel cloudu možnost si vytvořit vlastní servisní instanci.

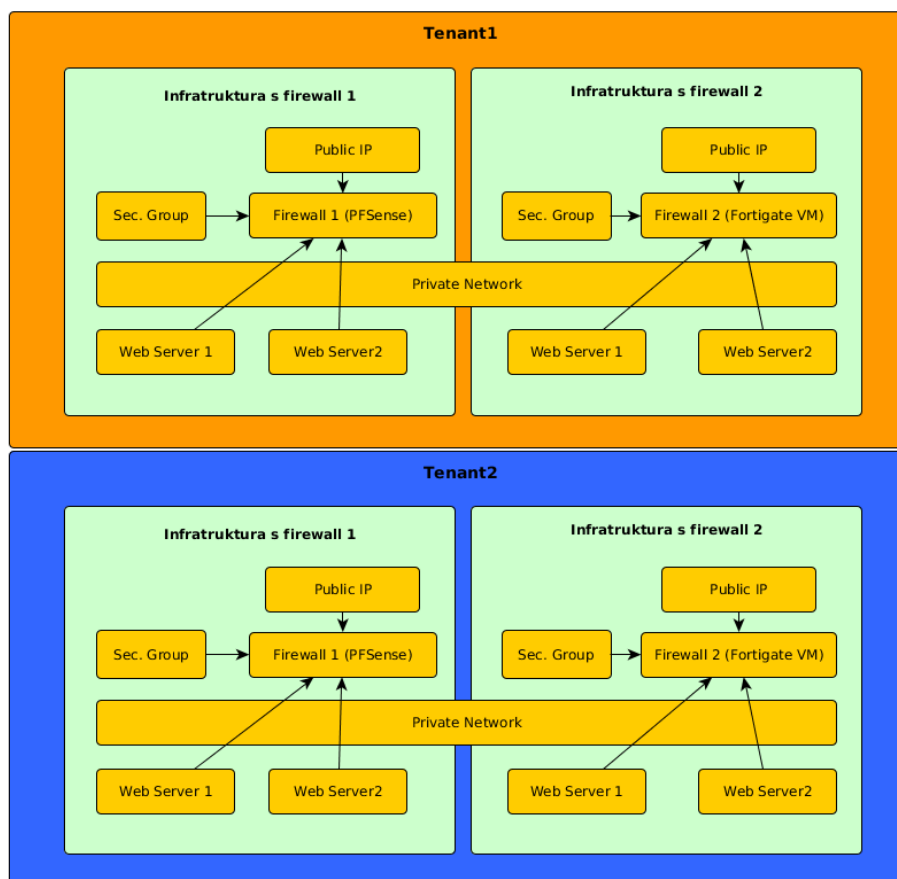
3.4.3 Scénář HA firewall

Největší výhodou použití firewall VNF v OpenContrailu je možnost nasazení tohoto firewallu ve vysoké dostupnosti.

3.4.4 FwaaS template

Pro FwaaS je naruhot heat template, který obsahuje:

- privátní síť

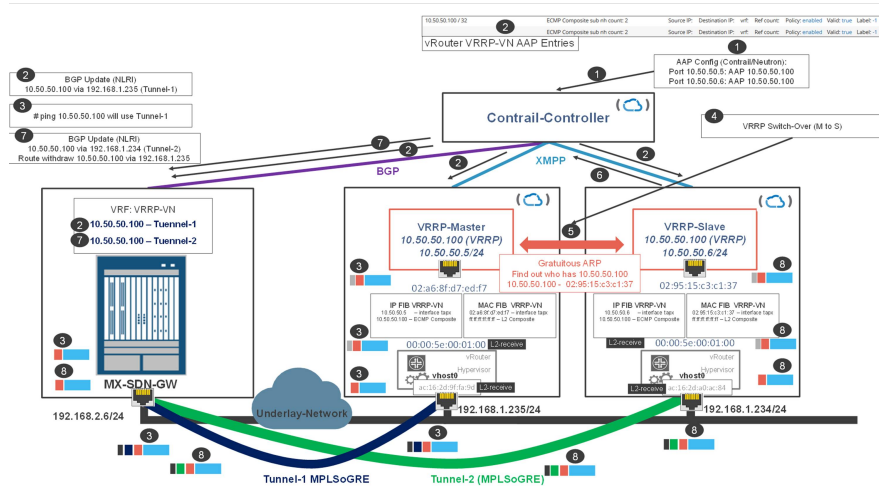


Obrázek 3.7: Firewall as a Service

```
private_net_1:
  type: OS::Neutron::Net
  properties:
    name: { get_param: private_net_1_name }

private_subnet_1:
  type: OS::Neutron::Subnet
  depends_on: private_net_1
  properties:
    network_id: { get_resource: private_net_1 }
    cidr: { get_param: private_net_1_cidr }
    gateway_ip: { get_param: private_net_1_gateway }
    allocation_pools:
      - start: { get_param: private_net_1_pool_start }
        end: { get_param: private_net_1_pool_end }
```

Ukázka kódu 3.8: Privátní síť



Obrázek 3.8: High Availability Firewall

- firewall template

```
service_template:
  type: OS::Contrail::ServiceTemplate
  properties:
    name: { get_param: template_name }
    service_mode: { get_param: template_mode }
    service_type: { get_param: template_type }
    image_name: { get_param: template_image }
    service_scaling: { get_param: scaling }
    availability_zone_enable: { get_param: availability_zone }
    ordered_interfaces: { get_param: ordered_interfaces }
    flavor: { get_param: template_flavor }
    service_interface_type_list: { "Fn::Split" : [ ",", Ref:
      service_interface_type_list ] }
    shared_ip_list: { "Fn::Split" : [ ",", Ref:
      shared_ip_list ] }
    static_routes_list: { "Fn::Split" : [ ",", Ref:
      static_routes_list ] }
```

Ukázka kódu 3.9: Firewall servisní instance

- firewall instance

```
service_instance:
  type: OS::Contrail::ServiceInstance
  depends_on: [private_subnet_1]
  properties:
```

```

name: { get_param: private_instance_name }
service_template: { get_resource: service_template}
availability_zone: { get_param: private_availability_zone}
scale_out:
  max_instances: { get_param: max_instances }
interface_list: [
  {
    virtual_network: "auto"
  },
  {
    virtual_network: {get_param: public_net}
  },
  {
    virtual_network: {get_resource: private_net_1}
  }
]

```

Ukázka kódu 3.10: Privátní síť

- virtuální instance

```

test_instance_01:
  type: OS::Nova::Server
  properties:
    image: { get_param: instance_image }
    flavor: { get_param: instance_flavor }
    key_name: { get_param: key_name }
    name: test-web01
    networks:
      - network: { get_resource: private_net_1 }
    security_groups:
      - default
    user_data_format: RAW
    user_data: |
      #!/bin/bash -v
      apt-get install apache2 -yy
      echo "Instance 01" > /var/www/html/index.html

```

Ukázka kódu 3.11: Virtuální instance pro testování

- contrail policy

```
private_policy:
  type: OS::Contrail::NetworkPolicy
  depends_on: [ private_net_1, service_instance ]
  properties:
    name: { get_param: policy_name }
    entries:
      policy_rule: [
        {
          "direction": { get_param: direction },
          "protocol": "any",
          "src_ports": [{ "start_port": {get_param:
            start_src_ports}, "end_port": {get_param:
            end_src_ports}}],
          "dst_ports": [{ "start_port": {get_param:
            start_dst_ports}, "end_port": {get_param:
            end_dst_ports}}],
          "dst_addresses": [{ "virtual_network":
            {get_param: public_net}}],
          "action_list": {"apply_service": [{get_resource:
            service_instance}]},
          "src_addresses": [{ "virtual_network":
            {get_resource: private_net_1}}]
        },
      ]
  ]
```

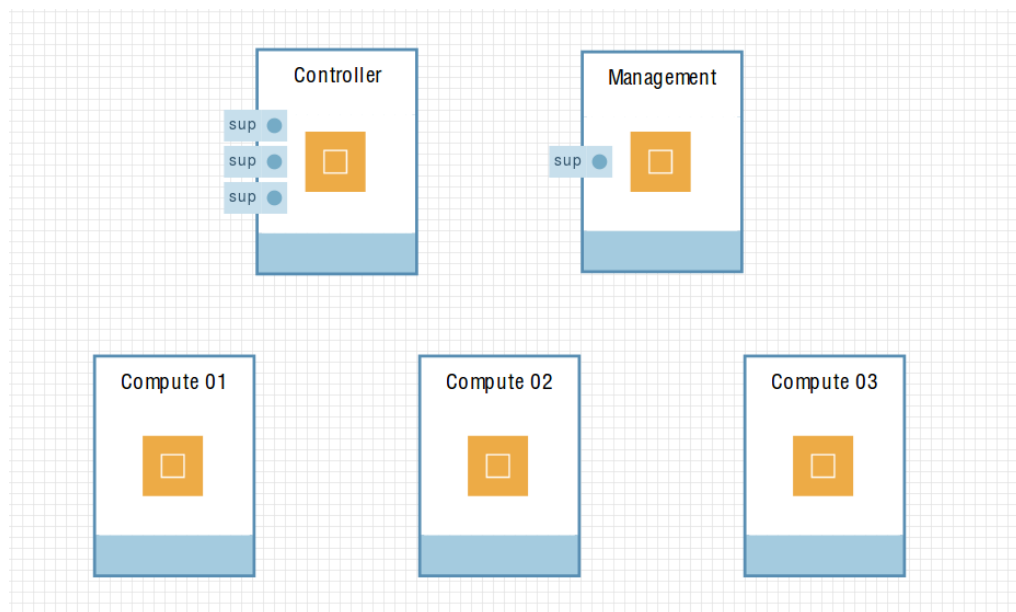
Ukázka kódu 3.12: Contrail network policy

4 Ukázka praktického spuštění jednotlivých VNF

V předchozí kapitole byly popsány návrh jednotlivých řešení pro příklady VNF, které byly v této práci použity. V této kapitole bude uvedeno několik příkladů, jak lze jednoduše vytvořit VNF v prostředí OpenStack a OpenContrail pomocí heat templatů. Všechna uvedená řešení byla testována v prostředí OpenStack s OpenContrailem, které bylo pro tyto účely poskytnuto společností tcp cloud a.s.

4.1 Testovací topologie

Celá testovací topologie se skládala z 4 serverů. Jeden server



Obrázek 4.1: Testovací topologie

4.2 Testované síťové funkce

Navrhnutá řešení v této práci představují virtuální síťové funkce pro firewall a load balancing. Jsou zde ukázány celkem 3 scénáře případu užití. Dva jsou zaměřeny na FwaaS (Firewall as a Service) a jeden na LbaaS (Load balancer as a Service). Všechna řešení jsou vytvořena pomocí Heat šablon, které se spouští v prostředí OpenStack.

Aby mohla být nějaká VNF vůbec vytvořena, tak musel být nejprve zvolen software či operační systém, který má požadovanou funkci implementovanou. Pro tyto účely byly použity následující řešení:

- PFSense – open-source firewall založený na operačním systému FreeBSD.
- FortiGate-VM – je plnohodnotně vybavený Fortigate firewall zabalený jako virtuální instance.
- Neutron Agent-HAproxy – je velmi rychlé a spolehlivé řešení nabízející vysokou dostupnost, load balancing a proxy pro aplikace založené na TCP a HTTP

Následující diagram znázorňuje logickou architekturu navrženého řešení dle referenční architektury zmíněné v kapitole 2.4. OpenStack spolu s OpenContrail poskytuje NFV infrastrukturu jednotlivé VNF jsou řízeny pomocí Heat.

4.3 Testování LbaaS

Pro vytvoření heat stacku s Load balancerem je nutné daný template vytvořit pomocí příkazu:

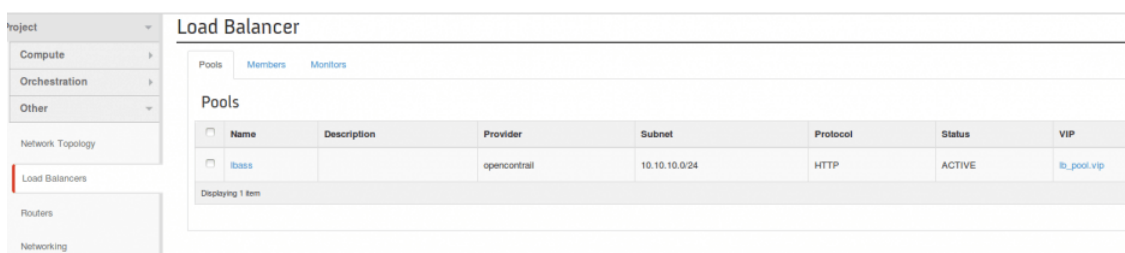
```
heat stack-create -f heat/templates/lbaas_template.hot -e heat/env/lbaas_env.env lbaas
```

Tento příkaz vytvoří všechny již uvedené prostředky pro load balancing. Konkrétní load balancer má nakonfigurovanou virtual ip adresu (VIP) a k ní přiřazenou floating adresu, která je přístupná z externích sítí. Zároveň má tento load balancer přiřazený pool, ke kterému je přiřazena privátní síť 10.10.10.0/24. Na obrázku č. X znázorňuje tento pool a obrázek č. X+1 jsou vidět členové (members) toho poolu.

Další zdrojem, který byl vytvořen je health monitor, který lze vidět na obrázku č. X+2. Díky němu má load balancer přehled o aktuálním stavu webových instancí. Pokud by náhodou některá z nich přestala odpovídat, v tomto případě na ping, tak by load balancer na tuto instanci přestal zasílat traffic.

Finální síťovou topologii znázorňuje obrázek č. X+3.

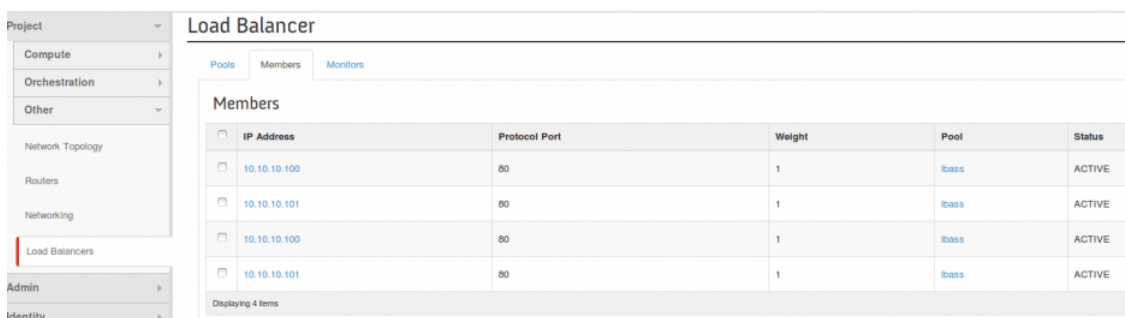
Otestování webových serverů lze provést příkazem curl, kterému dáme jako parametrem ip VIP nebo floating ip load balanceru. Po několika takovýchto zadání tohoto pří-



Load Balancer							
Pools							
Name	Description	Provider	Subnet	Protocol	Status	VIP	
lbass		opencontrail	10.10.10.0/24	HTTP	ACTIVE	lb_pool.vip	

Displaying 1 item

Obrázek 4.2: Vytvořený pool



Load Balancer					
Members					
IP Address	Protocol Port	Weight	Pool	Status	
10.10.10.100	80	1	lbass	ACTIVE	
10.10.10.101	80	1	lbass	ACTIVE	
10.10.10.100	80	1	lbass	ACTIVE	
10.10.10.101	80	1	lbass	ACTIVE	

Displaying 4 items

Obrázek 4.3: Vytvoření members

kazu je vidět, že oba web servery odpovídají a je probíhá mezi nimi load balancing metodou round robin. Celý tento test je vidět na obr. č. X+4

4.4 Testování Fwaas

Pro vytvoření heat stacku s PFSense z templaty lze použít příkaz:

```
heat stack-create -f heat/templates/fwaas_mnmg_template.hot -e heat/env/fwaas_pfsense_env.env pfsense
```

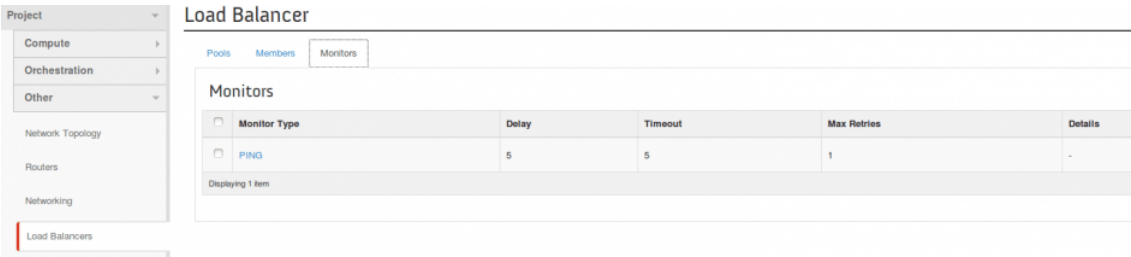
a pro vytvoření heat stacku s Fortigate VM jde vytvořit pomocí příkazu:

```
heat stack-create -f heat/templates/fwaas_mnmg_template.hot -e heat/env/fwaas_fortios_contrail.env fortios
```

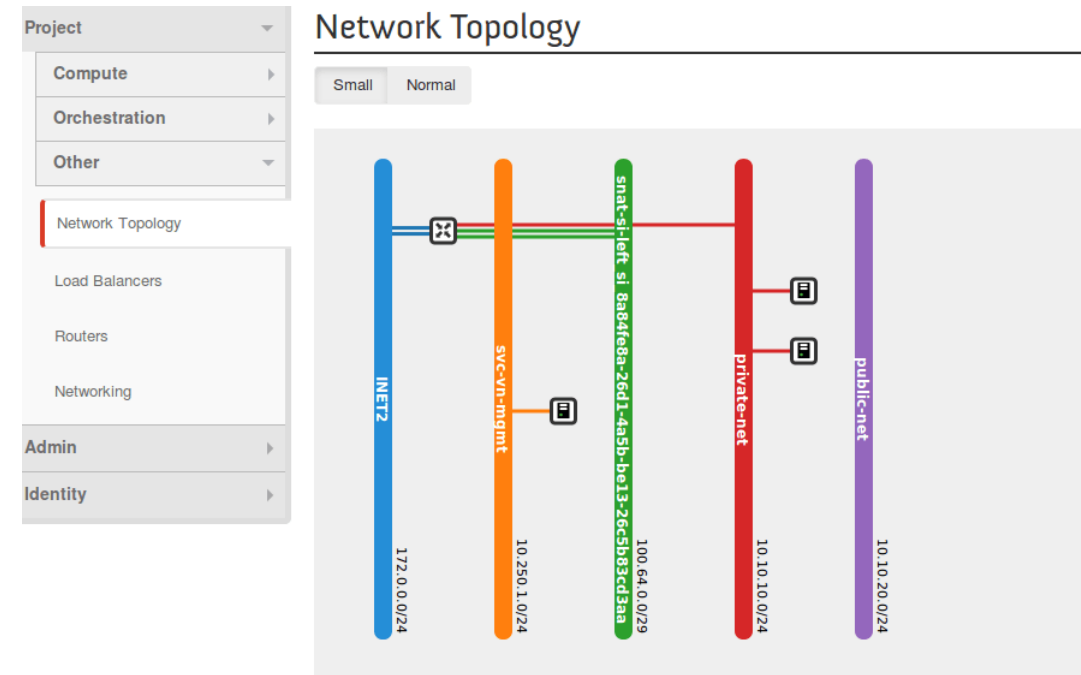
By default, pfsense firewall is configured to NAT after the heat stack is started. As a result, there is no need to make any configuration for this function. Pfsense image was preconfigured with DHCP services on every interface and there is outbound policy for NAT.

After we start the heat with pfsense there is already functional service chaining. Testing instance has default gateway to contrail and contrail redirects it to pfsense.

There is also NAT session in pfsense. In shell run command:



Obrázek 4.4: Vytvořený health monitor



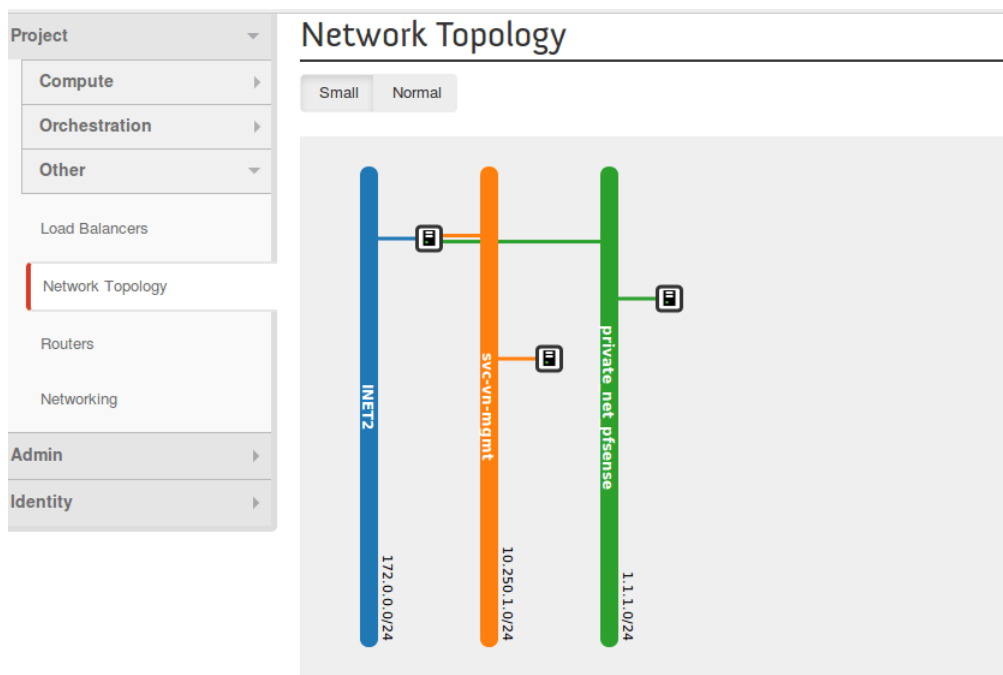
Obrázek 4.5: Vytvořená síťová topologie

4.4.1 Fortigate VM

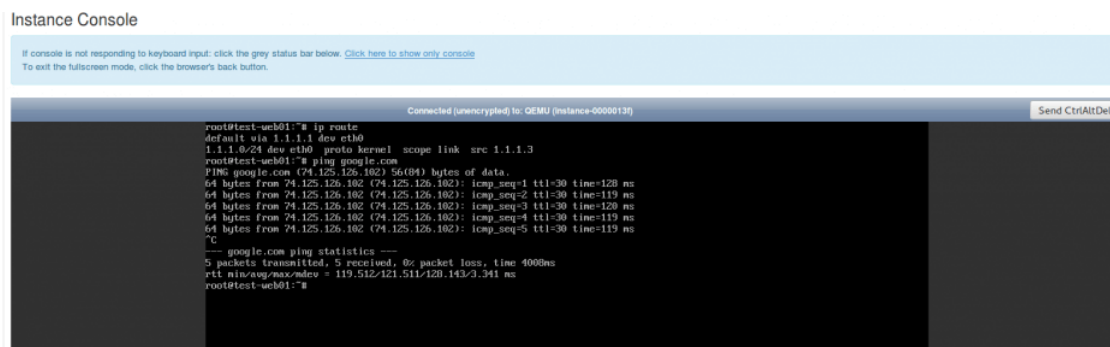
4.4.2 PFSense

```
File Edit View Search Terminal Help
root@Management:~# curl 172.0.0.6
Instance 01
root@Management:~# curl 172.0.0.6
Instance 01
root@Management:~# curl 172.0.0.6
Instance 02
root@Management:~# curl 172.0.0.6
Instance 02
root@Management:~# curl 172.0.0.6
Instance 01
root@Management:~# curl 172.0.0.6
Instance 01
root@Management:~# curl 172.0.0.6
Instance 02
root@Management:~# curl 172.0.0.6
Instance 02
root@Management:~# curl 172.0.0.6
Instance 01
root@Management:~# curl 172.0.0.6
Instance 01
root@Management:~# curl 172.0.0.6
Instance 02
root@Management:~# curl 172.0.0.6
Instance 02
root@Management:~# curl 172.0.0.6
Instance 01
root@Management:~# curl 172.0.0.6
Instance 01
root@Management:~# curl 172.0.0.6
Instance 02
root@Management:~# curl 172.0.0.6
Instance 02
root@Management:~# curl 172.0.0.6
Instance 01
root@Management:~#
```

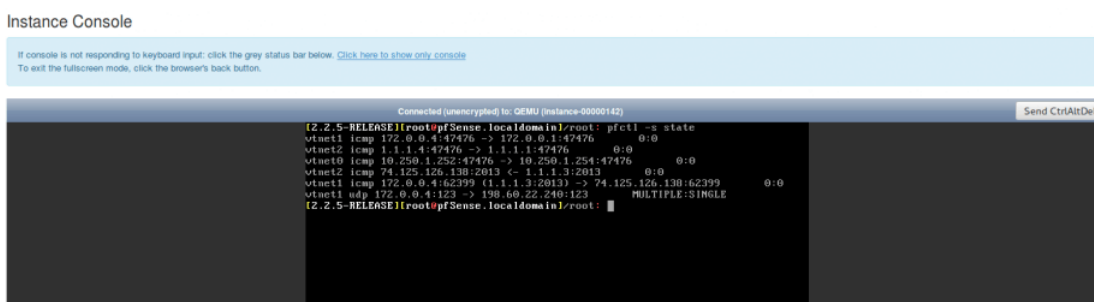
Obrázek 4.6: Test konektivity a load balancingu



Obrázek 4.7: Síťová topologie



Obrázek 4.8: Test konektivity PFSense



Obrázek 4.9: Ukázka NAT session

```

root@mnm01:~# python fortios_intf.py
This is the diff of the configs:

This is how to reach the desired state:
config system interface
    edit port1
        set allowaccess ssh ping http https
    next
    edit port2
        set defaultgw enable
    next
    edit port4
        set mode static
    next
    edit port5
        set mode static
    next
    edit port6
        set mode static
    next
    edit port7
        set mode static
    next
    edit ssl.root
        set mode static
    next
end
root@mnm01:~#

```

Obrázek 4.10: Fortigate VM intergace konfigurace

```

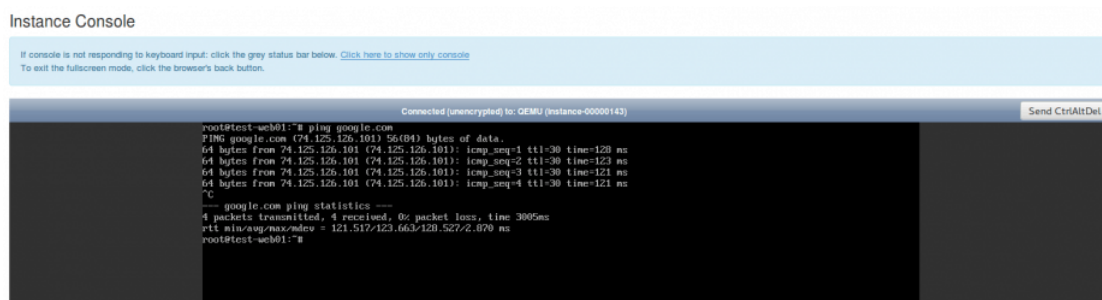
ubuntu@Management:~$ ssh root@172.0.0.5
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.19.0-26-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Tue Jan 12 10:03:49 2016 from mgmtserver14041vag
root@mnm01:~# ls
fabfile.py  fortigate-formula  fortios_intf.txt  fortios_nat.py  param.py  update.sh
fabfile.pyc  fortios_intf.py  fortios_nat.conf  fortios_nat.txt  text.py
root@mnm01:~# python fortios_nat.py
This is the diff of the conigs:

This is how to reach the desired state:
  config firewall policy
    edit 1
      set nat enable
      set service ALL
      set schedule always
      set srcaddr all
      set dstintf port2
      set srcintf port3
      set action accept
      set dstaddr all
      set logtraffic all
    next
  end
root@mnm01:~# █

```

Obrázek 4.11: Fortigate VM NAT konfigurace



Obrázek 4.12: Test konektivity

5 Shrnutí poznatků a možná budoucí rozšíření

K čemu to je dobrý, na co jsem narazil, atd.

6 Závěr

Je v paráda.

Literatura

- [1] R. Guerzoni, "Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges and Call for Action. Introductory white paper," in SDN and OpenFlow World Congress, June 2012. [online]. [cit. 2016-04-07]. Dostupné také z: https://portal.etsi.org/NFV/NFV_White_Paper.pdf
- [2]
- [3] Open platform for nfv. <https://www.opnfv.org/>. Accessed September 28, 2014.
- [4] MIJUMBI, Rashid, Joan SERRAT, Juan-Luis GORRICHIO, Niels BOUTEN, Filip DE TURCK a Raouf BOUTABA. *Network Function Virtualization: State-of-the-Art and Research Challenges*. IEEE Communications Surveys. 2016, 18(1), 236-262. DOI: 10.1109/COMST.2015.2477041. ISSN 1553-877x. Dostupné také z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7243304>
- [5] HAN, Bo, Vijay GOPALAKRISHNAN, Lusheng JI a Seungjoon LEE. *Network function virtualization: Challenges and opportunities for innovations*. IEEE Communications Magazine. 2015, 53(2), 90-97. DOI: 10.1109/MCOM.2015.7045396. ISSN 0163-6804. Dostupné také z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7045396>
- [6] KUSNETZKY, Dan. *Virtualization: a manager's guide*. Sebastopol, CA: O'Reilly, c2011. ISBN 1449306454.
- [7] J. Smith and R. Nair, "The architecture of virtual machines," Computer, vol. 38, no. 5, pp. 32–38, May 2005. doi: 10.1109/MC.2005.173
- [8] <http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2015-state-cloud-survey>
- [9] K. Chandrasekaran. *Essentials of CLOUD COMPUTING*. Boca Raton: CRC Press, 2015. ISBN 978-1-4822-0544-2.

- [10] JENNINGS, Brendan a Rolf STADLER. *Resource Management in Clouds: Survey and Research Challenges*. Journal of Network and Systems Management. 2015, 23(3), 567-619. DOI: 10.1007/s10922-014-9307-7. ISSN 1064-7570. Dostupné také z: <http://link.springer.com/10.1007/s10922-014-9307-7>
- [11] ETSI, “Network Function Virtualization: Use Cases”, http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf, 2013
- [12] KREUTZ, Diego, Fernando M. V. RAMOS, Paulo ESTEVES VERISSIMO, Christian ESTEVE ROTHENBERG, Siamak AZODOLMOLKY a Steve UHLIG. *Software-Defined Networking: A Comprehensive Survey*. Proceedings of the IEEE [online]. 2015, 103(1), 14-76 [cit. 2016-04-09]. DOI: 10.1109/JPROC.2014.2371999. ISSN 0018-9219. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6994333>
- [13] DOHERTY, Jimmy. *SDN and NFV simplified: a visual guide to understanding software defined networks and network function virtualization*. 1st edition. Indianapolis, IN: Addison-Wesley Professional, 2016. ISBN 9780134306407.
- [14] ETSI Industry Specification Group (ISG) NFV, “ETSI GS NFV 002 V1.2.1: Network Functions Virtualisation (NFV); Architectural Framework,” December 2014. [online]. [cit. 2016-04-07]. Dostupné také z: <http://www.etsi.org/deliver/etsisgs/NFV/001099/002/01.02.0160/gsNFV002v010201p.pdf>
- [15] ETSI Industry Specification Group (ISG) NFV, “ETSI GS NFV 003 V1.2.1: Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV,” December 2014. [online]. [cit. 2016-04-07]. <http://www.etsi.org/deliver/etsisgs/NFV/001099/003/01.02.0160/gsNFV003v010201p.pdf>
- [16] http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/001/01.01.01_60/gs_nfv-inf001v010101p.pdf
- [17] http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/003/01.01.01_60/gs_NFV-INF003v010101p.pdf
- [18] http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/004/01.01.01_60/gs_nfv-inf004v010101p.pdf
- [19] http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/005/01.01.01_60/gs_NFV-INF005v010101p.pdf

- [20] http://www.etsi.org/deliver/etsi_gs/NFV-SWA/001_099/001/01.01.01_60/gs_nfv-swa001v010101p.pdf
- [21] http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf
- [22] MIJUMBI, Rashid, Joan SERRAT, Juan-luis GORRICHIO, Steven LATRE, Mari-nos CHARALAMBIDES a Diego LOPEZ. *Management and orchestration challenges in network functions virtualization*. IEEE Communications Magazine. 2016, 54(1), 98-105. DOI: 10.1109/MCOM.2016.7378433. ISSN 0163-6804. Dostupné také z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7378433>
- [23] KHEDHER, Omar. *Mastering OpenStack: design, deploy, and manage a scalable Open-Stack infrastructure*. First published. Birmingham: Packt Publishing, 2015. Community experience distilled (Packt). ISBN 978-1-78439-564-3.
- [24] JACKSON, Kevin. *OpenStack cloud computing cookbook*. Third Edition. Birmingham: Packt Publishing, 2015. ISBN 978-1-78217-478-3.
- [25] GALLAGHER, Simon a Aidan DALGLEISH. *VMware private cloud computing with vCloud Director*. Indianapolis, Ind.: Sybex, c2013.
- [26] RIJSMAN, Bruno a Ankur SINGLA. *Day One: Understanding OpenContrail Architecture*. Juniper Networks Books, 2013
- [27] Architecture Dokumentation. SINGLA, Ankur a Bruno RIJSMAN. *OpenContrail*. [online]. Juniper Networks Books, 2013 [cit.2016-04-04]. Dostupné z: <http://www.opencontrail.org/opencontrail-architecture-documentation/>
- [28] VMware *NSX Network Virtualization Design Guide*. VMware [online]. 2013 [cit. 2016-04-07]. Dostupné z: <https://www.vmware.com/files/pdf/products/nsx/vmw-nsx-network-virtualization-design-guide.pdf>
- [29]
- [30]
- [31]
- [32]

[33]

[34]

[35]

Přílohy

Seznam obrázků

2.1	Koncept virtualizace síťových funkcí (NFV)	5
2.2	Schéma hypervisorů	6
2.3	Schéma SDN, převzato z [12]	9
2.4	Ukázka klasického service chainu pomocí fyzických síťových prvků . .	11
2.5	Ukázka VNF service chainu	11
2.6	NFV architektura, převzato z [14]	12
2.7	Schéma NFV infrastruktury	14
2.8	Schéma virtuální síťové funkce	15
2.9	Schéma NFV MANO	16
2.10	Schéma VMware vCloud, převzato z [25]	20
2.11	Schéma OpenContrail, převzato z [26]	22
3.1	Architektura NFV řešení	25
3.2	Popis heat orchestrace	26
3.3	Load Balancer as a Service	27
3.4	Neutron LbaaS	28
3.5	Schéma zapojení servisní instance Analyzer	34
3.6	Schéma zapojení servisní instance	35
3.7	Firewall as a Service	36
3.8	High Availability Firewall	37
4.1	Testovací topologie	40
4.2	Vytvořený pool	42
4.3	Vytvoření members	42
4.4	Vytvořený health monitor	43
4.5	Vytvořená síťová topologie	43
4.6	Test konektivity a load balancingu	44
4.7	Síťová topologie	44
4.8	Test konektivity PFSense	45
4.9	Ukázka NAT session	45
4.10	Fortigate VM intergace konfigurace	45

4.11 Fortigate VM NAT konfigurace	46
4.12 Test konektivity	46

Seznam tabulek

Seznam ukázek kódu

3.1	Privátní síť a subnet	30
3.2	Web server 1	30
3.3	Web server 1	31
3.4	Public síť a subnet	31
3.5	Load balancer pool	32
3.6	Members	32
3.7	Monitor	32
3.8	Privátní síť	36
3.9	Firewall servisní instance	37
3.10	Privátní síť	37
3.11	Virtuální instance pro testování	38
3.12	Contrail network policy	38

Podklad pro zadání DIPLOMOVÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Smola Ondřej	Polizy 16, Osice - Polizy	11475

TÉMA ČESKY:

Orchestrace a management virtuálních síťových funkcí

TÉMA ANGLICKY:

Orchestration and management of virtual network functions

VEDOUcí PRÁCE:

Ing. Vladimír Soběslav, Ph.D. - KIT

ZÁSADY PRO VYPRACOVÁNÍ:

Cílem této práce je analyzovat možnosti vytváření a nasazení virtuálních sítí v cloud computingu s důrazem na technologie VNF nad NFV a jejich srovnání. V rámci závěrečné práce budou analyzovány metody a nástroje pro vývoj a automatizaci služeb virtuálních sítí. V závěrečné části provede autor implementaci VNF řešení v prostředí cloud computingové platformy OpenStack.

Osnova:

1. Úvod
2. Problematika virtualizace síťových funkcí
3. Testovací prostředí
4. Příklad virtualizace síťových funkcí
5. Shrnutí
6. Závěr

SEZNAM DOPORUČENÉ LITERATURY:

DOSTÁLEK, Libor.; KABELOVÁ, Alena. Velký průvodce protokoly TCP/IP a systémem DNS. 5. aktualizované vydání, Brno: Computer Press, a.s., 2008. 488 s. ISBN 978-80-251-2236-5.

HICKS, Michael. Optimizing Applications on Cisco Networks. 1. vydání. Indianapolis: Cisco Press, 2004. 384 s. ISBN: 978-1-58705-153-1.

HUCABY, David. CCNP SWITCH 642-813 Official Certification Guide. 1. vydání. Indianapolis: Cisco Press, 2011, 533 s. ISBN 978-1-58720-243-8.

Podpis studenta:

Datum:

Podpis vedoucího práce:

Datum: