Proclamation

I declare that this thesis was made by myself with assistance of my supervisor. All parts taken over word by word from literature or other publications are referenced. I approve publishing this thesis or any part of it with referencing author of original text.

In Prague at 2014-11-11 .....

Abstract

Contents

Part I

Theoretical

Virtualization

Virtualization is, in my opinion, the most important technology in data centers, because it caused significant progress in this field. It is not technology itself, so it should rather be called model than technology.

Definition of virtualization as stated in [2] says that "virtualization is a technique for hiding the physical characteristics of computing resources from the way in which other systems, applications or end users interact with those resources. The concept of virtualization is very broad and can be applied to devices, servers, operating systems, applications and even networks." This definition gives description of the virtualization and can be applied to any type of virtualization.

The most common approach is virtualization of computers, because it is the oldest one and most widely used there days. It started in 1960s with mainframes as an attempt to employ resource sharing and this idea is still alive in current time. Virtual computer is logical representation of computer in software. [2] Virtual computers are usually called virtual machines (VM) and physical machine hosting VMs is called hypervisor. Rigorous term for physical hosting machine is host and hypervisor is software performing the virtualization, but word hypervisor is widely used in technical text for machine as well. It is possible and very advantageous to host many virtual machines on single physical computer, because it brings technical and economical benefits. Decoupling computer and it's software from hardware is important advantage, because it brings additional level of abstraction and gives ability to shift virtual machines between hypervisors. Economical benefit is quite obvious, since it is not necessary to buy single physical server for every service and electricity saving are also appreciable.

Another important type of virtualization is virtualization of networks. It is usually used together with computer virtualization, since it gives an occasion to separate network devices from network itself. Physical machines are not as flexible as VMs are, so plugging them into virtual network is not as beneficial as VMs, because there are still physical network cables, that can be hardly virtualized. There is a hot topic called Software Defined Networking (SDN) having potential to provide virtualization info physical network infrastructure, thus it may be good idea to integrate physical machines into virtual network as well.

Storage virtualization should also be taken into account, because it provides abstraction of the storage. Typical unvirtualized storage uses some physical device for storing data and metadata, but this approach is not flexible enough since it is usually limited to just one physical machine or group of machines connected to shared storage. It is necessary to find any method of storage virtualization, which would be able to provide any storage to any physical of virtual computer.

Service virtualization, memory virtualization, I/O virtualization or database virtualization are another types of virtualization. It is not necessary to mention all the types of virtualization because it is possible to virtualize almost everything and emerging of new types is quite probable.

Term virtualization is going to be used in further text as computer virtualization, another types of virtualization will always be denoted.

1.1 Types of virtualization

There are three different virtualization types and they vary by method used t add virtualization layer between guests and physical hardware. It it not possible to easily choose better or worse virtualization types, because it depends on intended usage, character of computing tasks and required operating system.

Architectures of computers, especially x86, are designed to run on physical devices, thus is in not easy to virtualize them. Access to hardware is controlled by priority levels called rings. Lowest priority is used by userspace applications and highest priority (ring 0) is reserved for operating system. It is necessary to insert virtualization layer between operating system and hardware, but there is not any ring with higher priority than operating system uses. This problem needs to be solved and it is not only one challenge. There are sensitive instructions incompatible with virtualization, because they use different semantics when they are not run in ring 0, as mentioned in [9].

1.1.1 Paravirtualization

Paravirtualization is type of virtualization with necessity of modifications in guest kernel. Modifications of kernel are necessary, because operating system uses non-virtualizable instructions that are trying to gain direct access to the hardware. These instruction need to be replaced with hypercalls that communicate directly with virtualization layer of hypervisor. [9] It is obvious, that guest operating system knows it is running virtualized.

Biggest advantage of paravirtualization is lower overhead compared to other types, because it is not necessary to translate instructions before running. However this advantages becomes less significant during time since there are already available processors optimized to run hardware assisted virtualization with less overhead. Main drawback of this type of virtualization is need for modifications done at an operating system, which is not always possible or allowed. Running modified OS also brings additional administration and thus additional cost.

It is possible to take a different look at paravirtualization and do not try to create entire virtual machine, but use operating system-level virtualization, where kernel allows to run multiple userspaces. These userspaces are called containers and therefore this approach is sometimes called container virtualization. It does not provide entire isolated virtual machine, but allows to run software packed in container. It is advantageous because there is almost none overhead in running software from container while maintaining sufficient level of container isolation. Container virtualization is applicable for situation, where whole virtual machine is not needed and then brings huge performance improvements since operating system layer is shared. Some says, that containers are going to bring next revolution info virtualization. For example Dustin Kirkland, Cloud Solutions Product Manager at Canonical wrote: "Linux containers, repositories of popular base images, snapshots using modern copy-on-write filesystem features. Brilliant, yet so simple. Docker.io for the win!" [10]. I think, that container virtualization may brings compelling advantages and I also

like using it, but it is not suitable for every situation. It is still technically kind of paravirtualization and thus it is limited to provide only additional layer on host's operating system.

1.1.2 Full virtualization

Virtualization type capable or running unmodified operating system is called full virtualization. It utilizes runtime translation, which captures non-virtualizable commands and emulates them using hypervisor virtualization layer. Virtualizable instructions are executed directly on the hardware. Modification of "problematic" calls is carried by the hypervisor and it is the main difference compared with paravirtualization.

Most important benefit of full virtualization is it's ability to run guest operating system without any changes, so guest OS is not aware of being virtualized. This makes guest operating system fully abstracted from underlaying hardware, it is possible to multiple different operating system on single host and provides simple migration from physical to virtual machine. Drawback of this type is overhead caused by catching and translating non-virtualizable calls.

1.1.3 Hardware assisted virtualization

Full virtualization has significant overhead caused by binary translation, so CPU vendors introduced technologies capable of inserting virtualization layer between ring 0 and physical hardware. It speeds-up trap of privileged and sensitive calls to the hypervisor and it is not necessary to perform binary translation of to modificate kernel of guest operating system.

Benefit of this type is quite obvious, because it lowers virtualization overhead and thus provides better performance compared with full virtualization together with elimination of need for guest kernel modifications compared with paravirtualization. It is necessary to have a support in host's CPU is primary drawback of this type, but there is support in almost every processor in current marker.

Running unmodified guest operating system leaves all necessary translations of instructions on hypervisor layer, so I would be good to to introduce small changes to guest's operating system, which will reduce work left for the hypervisor but also do not need any significant changes in guest's kernel. This approach is called hybrid virtualization and it is subset of hardware assisted virtualization. Installation of additional drivers is required, but it is not necessary to apply any changes on whole kernel. These drivers are aware of virtualization and use virtualization layer directly without any translations made by the hypervisor. This method increases driver's IOPS and therefore it is usually used for virtualized network cards and storages. Driver able to deliver hybrid virtualization is *virtio* for KVM, Xen call it *paravirtualized device drivers* and VMWare *Guest Tools*.

1.1.4 Summary on types of virtualization

There were presented some virtualization general virtualization types and their pros and cons. There is not any universal virtualization type suitable for all use cases, thus is is always possible to decide on planned usage. It also depends whether

it is required to run different kernel on single physical host or it is sufficient to share one kernel for all containers. Differences are compared in table 1.1.1.

We can divide types into two groups:

- One group provides guests with full virtual machine, every VM uses it's own isolated kernel and VMs are fulll or almost fully decoupled from hardware. Full, hardware assisted and hybrid virtualization belongs to this group.
- Members of second group are containers and paravirtualization. This group is specific by lightweight containers and host kernel shared by all running containers.

Virtualization is massively used even by czech IT companies. First group is used for example by *Wedos* for their virtual server hosting and related services. Second group is uses by *Seznam.cz* and they use LXC for web serves as well as for Hadoop cluster.

Table 1.1.1: Comparison of virtualization types

Type	**method**	**guest modif.**	**usage**
Paravirtualization	hypercalls by guest kernel	yes	same workloads and same OS
Full	translation of instructions	no	when full abstraction is needed
Hardware assisted	translation with help of hardware	no	same as full, but with compatible CPU
Hybrid	translations and driver changes	driver only	when possible to install additional drives

1.2 Advantages of virtualization

Most important advantages is decoupling software from physical hardware, at least in my opinion. It is possible to migrate virtual machines with running services between physical hosts without significant impact on service behavior. This brings amazing opportunity to adapt service environment on demand and scale the service.

It is possible to perform any hardware and software upgrades, because all running services may be temporarily migrated to other physical host. Virtual machines are much more easier to deploy than physical ones. It takes only a few seconds to create and run VM compared to at least hours to deploy physical machine. Deploy of virtual machine do not have to be performed by persons, because it is possible to employ an orchestration and scale up the service (add virtual machines) automatically. Reset of virtual machine is actually just software instruction in hypervisor, so it may by done remotely with ease.

Geographical backups or failover is much more easier to accomplish with virtualization approach. You can rent virtual machine from provider in foreign country

and start your services in a few moments. It is huge simplification compared with running physical machine at foreign data center.

Virtualization brings also some economical and environmental advantages. Economical advantages are quite obvious, because it is no longer necessary to buy physical servers. Non-virtualizational approach requires one physical machine for every running server, but it is not longer necessary with virtualization. It is possible to run many virtual servers or containers on single physical machine. It is also possible to move even to the higher level of CAPEX cutting and rent virtual machine from provider and absolutely eliminate need for running any server machine. Renting virtual server increases OPEX, but they are more flexible and easier to control. Electrical consumption should also be taken into account, because single physical machine, even under higher load, will definitely consume less power compared with two or more similar machines.

I asked Petr Hodač, technical manager at SiliconHill and he stated, that they managed to reduce electricity consumption of whole server room by 19% iter alia due to deployment of virtualization. It produced also additional saving, because they need less UPS batteries and less cooling capacity, but savings on cooling are not included in mentioned savings.

However there are some drawback too. Failure of physical host causes failure of all virtual machines or containers running on this host. It is kind of single point of failure, but we can fight it with duplication of service nodes between different hosts, datacenters, providers or continents. Another disadvantage is hidden in additional virtualization level, since it is necessary to take care of hypervisors. I is not a real disadvantage, since traditional non-virtualized approach needs to take a care of many virtual machines.

Deployment of virtualization should always be well planned, because it can bring many amazing advantages, but it is also able to cause a disaster in case of poor system design or lame administration.

Cloud computing

It is possible find many services called "cloud based" and it is important to agree on accurate definition of these services. It is quite clear, that cloud based service will use principle of cloud computing. Definition of cloud computing by NIST says, that "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) than can be rapidly provisioned and released with minimal management effort or service provide interaction." [11]. This definition clarifies what cloud computing is, but says nothing about parameters and used technologies.

I think, that it would be more convenient to start definition from lower levels, which provides elementary parts, and get to the cloud service afterwards. This definition gives different look at cloud computing than NISTs, but it uses same conditions and therefore results are basically same. It focuses on currently used principles, which may change during time, so it may not be valid after some time, but it provides more technical overview on operation of cloud services.

Cloud computing services are nowadays heavily dependent on virtualization, because it allows to replace physical machines with virtual machines (VMs) or containers and brings a lot more flexibility than physical machine can ever provide.

Basic part of cloud computing system is virtual machine. Physical machine can also be part of the cloud system, but it is not able to deliver required rapid provisioning and it is not possible to deploy physical machine without service provider interaction. Virtual machine is elemental resource and also use some additional resources. These resources can be for example networking, which is used for interconnection between VMs as well as for reaching customers, storage used for system internal or customer data. It is important do employ some configuration management and orchestration, because it is able to deliver rapid provisioning of virtual machines and minimizes effort required for administration.

Virtual machines together provides the service, which is exposed to users via any kind of network. It doesn't matter whether customers access the service directly at virtual machines or via a proxy, but hiding worker VMs brings additional flexibility for migration and scalability.

Difference between cloud computing and bare virtualization is intelligence included in cloud, because it may be controlled automatically according to events or monitoring observed at cloud system. It is common to supply customers with configuration interface, which allows them to tune service parameters and provides user-friendly interface for administration. Bare virtualization does not offer any intelligence, even if it is equipped with shiny user interfaces with opportunity to scale virtual machines up or down, because all change performed manually.

Cloud computing is kind of hype these days, so it is often used just for marketing purposes and thus it is recommended to perform service analysis and do no absolutely trust every buzzword used in specification.

2.1 Deployment models

There are three scenarios possible for deploying cloud solutions. Models differs by ownership and subject responsible by administration of the system. Right solution depends on expected load, available budget as well as on expected classification of data. Public model and private model are mutually contradictory and last model called hybrid is combination of first two mentioned. Model are compared in table 2.1.1.

2.1.1 Private

Private model defines cloud environment build exclusively for single subject. Typical scenario is to build private cloud in datacenter owner by the subject, but it is not strictly required. There is common misunderstanding of term private, because it means private usage of cloud resources and not private ownership of cloud infrastructure. Private cloud may be leased from third-party provider and it also can be running on third-party hardware.

Running private cloud gives an advantage in elimination of any inter-tenant isolation problems and it is possible to adapt configuration to fit owner needs. There is a law, which forces sensitive data to be stored internally and with limited access, so it is necessary to build private or hybrid cloud for this kind of usage. It is not clear how to handle clouds and especially storages with law, because it this topic is wide and it is not possible to define rigid rules. There is currently running case with US judge ordering Microsort to provide data stored in Ireland. [12]

Drawback of private cloud is higher initial cost and probably also higher operational costs, but it depends on expected usage.

2.1.2 Public

Public cloud deployment model is based on resource sharing between the tenants. There is usually one subject called cloud provider and many customers (tenants) and these tenants buy resources and rights to use them. Resources are usually charged according to it's usage.

Billing per resource usage is called pay-per-use and it is interesting method of shifting costs between initial and operational. There are usually plans with various CPU, memory and storage options and final cost depends on real usage of resource. Pricing plan based on pay-per-use is favourable to services with low load with occasional peaks. Service under constant high load is not well suited for this payment model, because it does not bring any benefits. It is also to run scalable application, since unscalable will not be able to

Infrastructure is not dedicated and it is shared between tenants. Resources are shared, but must be strictly isolated, because it is unacceptable to allow any interference between tenants, unless they make an explicit request to allow it.

It is common to provide services with flexible parameters, for example Amazon calls it EC2 - Elastic Compute Cloud. Elasticity of provided services allows tenant to use more resources when needed and fall back to usual amount.

2.1.3 Hybrid

Hybrid cloud is model utilizing both, public and private, previous mentioned models. The goal is to combine advantages of both model and eliminate drawbacks. Public cloud is usually more cost effective, but may not be able to meet the security requirements and on the other hand private cloud can be designed to comply with users requests, but it is expensive. Hybrid designed solution can use private cloud part for confidential data and public cloud for less sensitive ones.

It is also possible to utilize cloud bursting in which system runs in private cloud and delegates part of load into public cloud. Lets describe it with application for collecting votes - sensitive part responsible for counting votes and generating results report will run in private cloud and public report will be saved to and served from infrastructure of public cloud. High level of security of counting votes is guaranteed and application is also able to deliver results to many subscribers as it can scale up into public cloud.

Table 2.1.1: Comparison of deployment models

model	**private**	**public**	**hybrid**
initial cost	higher	lower	medium
operational cost[1]	higher	lower	medium
security	higher	lower	medium
elasticity	lower	high	medium

2.2 Service models

Purpose of cloud computing is to deliver the service and provide customers with tools to manage this service. Service models differs by level of control provided to customers and thus with areas of responsibility. I am going to call border between responsibility of customer and responsibility of provider as responsibility border. Responsibility borders according to service models are depicted at figure 2.2.1.

Some of service models leave almost all control of service and responsibility at provider side and other supplies customer with more control. It is necessary to select right service model according to expected service usage and required control level.

2.2.1 Infrastructure as a Service

IaaS is model with the most of configuration tasks left at customer's side. Customer is responsible for virtual machines and it's services, so it gives much more flexibility than other models and it is well-suited for services with extraordinary requirements.

Customer manages virtual machines as well as running services, so provider is responsible only for virtualization and underneath layers. It is even possible to run custom operating system, but provider usually offers prepared images with different operating systems. Prepared OS images are tested and modified to run well in cloud environment. There can be installed hybrid virtualization drivers, kernel tweaked to run virtualized and it is also good idea to remove useless drivers and software.

This model is good choice if special configuration is needed, but service deployment is more difficult because some expertise is required. IaaS can be used if customer require additional level of security, because virtual machine can use crypted volume and make data unreadable for provider. Unfortunately it is still possible for provider to acquire confidential from other sources, for example from memory, but it is much complicated to perform this.

Typical example of IaaS is Amazon Web Services and Active24's "Virtuální Privátní servery".

2.2.2 Platform as a Service

Border of responsibility of PaaS is located two layers higher compared to IaaS. Service provider is responsible for platform and all underlaying layers, thus provider takes care of same layers as in IaaS plus operating system and platform. Leaving operating system maintenance on provider's side may be beneficial, because provider can adjust operating system for virtualization and takes care about software updates.

Provider usually manages a lot of operating systems for many customers, so this updating and maintenance tasks may be automatized or executed in batch. Sharing operating system layer between customers with preserving adequate level of isolation can save many resource and make operating system administration even easier.

Customer using service according to this model runs his own software and does not take care of any lower layers. It is not necessary to do any administration tasks and more effort can be given to application development.

This model is well-suited for running applications without any special requirements. It makes service deployment faster and easier, but is more limited by used platform. Typical example is project Evia and Microsoft Azure.

2.2.3 Software as a Service

SaaS is model with none of administration tasks left at customer's responsibility. Software is hosted and maintained by service provider and customer is using the service. Service is accessed remotely via network (usually Internet). It is common mistake to say, that service is accessed as Web service, because any remote access can be used.

TODO: finish this part

This model is right solution for customers looking for service without hassling with it's administration.

2.3 Networking

Networking is essential part of cloud computing because it is not be possible to access any services without networking. Every service in cloud computing system is accessed via network. Network is usually also used for communication between virtual machines, migrations, storage access and for many other tasks.

First part of networking to think about is physical layer. Various Ethernet versions are used for physical layer in cloud data centers. There are many versions with different link bandwidth and wiring, but 1G and 10G with twisted pairs or optical fibers is used most widely. There were 100M called Fast Ethernet but it does

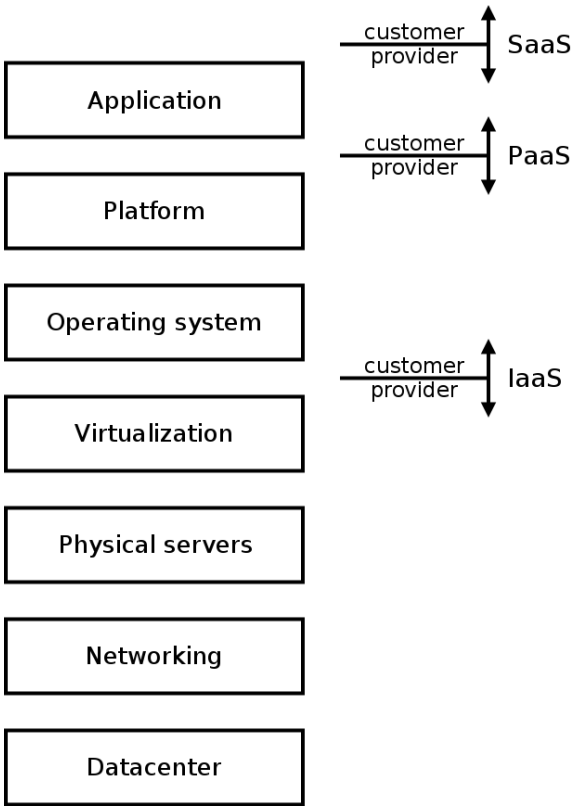

Figure 2.2.1: Service model responsibility

not make sense to use this for server link today, because of it's limited bandwidth and almost similar price compared to 1G.

It is common to insert two independent NICs into every server and connect them into independent ToR switches, because it improves fault-tolerance. There is usually one more NIC for remote management and some additional cards if SAN is used. Remote management can be connected using detached cable or shared with any of network cards. Separate cable brings more flexibility and fault-tolerance and shared cable reduces cabling effort and thus simplifies maintenance and improves cooling effectiveness. Both solutions are used.

About 40 servers fits into traditional rack and these serverl need to be connected to network infrastructure. There are different topologies, but most common is variant of two or three tier hierarchical structure. [8] There is also approach called "fabric" which implies non-blocking every-to-every mesh connection between switches. However fabric technologies are proprietary and limited to vendor. TODO: doplnit a více ocitovat surver-architectures

Every rack contains about 40 servers and these servers are connected to switch called ToR. This switch is located in the rack and acts as access layer for servers. Servers are connected to at least two ToRs if additional fault-tolerance is needed. Upper network topology layers depends on data center size and scaling requirements. There can be distribution and core layer, collapsed core or some kind of fabric.

Physical topology must be adjusted to spread network layers between two or more datacenter networks. It is obviously not possible to spread physical layer, but data link layer and upper layers are possible to spread.

Another view on network topology is at network layer. Internet is based on TCP/IP so it is necessary to use this protocol family and assign IP addresses to servers, virtual machines and other network elements. There are two different versions of IP protocol:

version 4 is the older one, with 32 bit address space. This version is still used more than version 6 even though it's address space is depleted and new version exists for more than 15 years.

version 6 is the "new" one, uses 128 bit address space and different headers, thus it is incompatible with version 4.

Moder data center must provide both versions of IP protocol, because supporting only one versions is a huge limitation and can not be accepted for new services deployment. However there is a problem with obtaining IPv4 addresses, because available pool had already been depleted and all available addreses had been divided between RIRs. It is beneficial to make efforts to employ IPv6 protocol as primary one and try to limit the amount of required IPv4 addresses.

There are different ways how to use both versions concurently:

- Dual-stack is the simpliest and probably the most used solution. Each interface gets at least one IPv4 and one IPv6 address. Use of both versions causes additional maintenance effort because it is necessary to take care of two separate L3 networks.

- Tunelling IPv6 via existing IPv4 infrastructure with technologies like 6to4, 6rd or ISATAP is another way. This solution can be used for IPv6 deployment in networks with working IPv4, because it is used for transmission of all packets. Tunelling is usually focused on deployment in access networks, but deployment in data center network is also applicable, as described in [14].

- Translating IPv4 addresses into part of IPv6 addressing space is different approach than previous mentinoned, because it operates on IPv6-only networks. This technique does not require every box to have assigned an IPv4 address and thus is good for saving address space. Hovewer address translations may not be suitable for data center usage, because it does not preserve original IP addresses and makes customer tracking almost impossible. Further information can be found in [1].

It is beneficial to deploy protocol IPv6 as primary one in my opinion, because it will become more and more needed during time. Hardware on current market usually support protocol IPv6 at least partialy, but there are still some hidden pitfalls. There may be problem for example with server's remote management, because it may not support IPv6 and thus is totally unusable on IPv6-only network. None of servers I have used for practical part of this thesis have support for IPv6 on IPMI. However there is currently only small demand and IPv6 deployment does not bring any direct profit. Even thought there are many problems and advantages are quite hidden, it is not possible to ignore this protocol and stay with old IPv4.

Virtual machine migration must be taken into account durring addressing schema design, because it plays crucial role in data center operation. Migrations are performed between hypervisors, i.e. physical servers, and these servers may be located

in different racks, halls or even in different data centers. It is usually required to preserve IP address of virtual machine during migration process and thus adressing schema must be prepared to move single IP address around almost whole data center nwithout any significant configuration changes.

First solution for unlimited migrations while preserving IP address is L2 sharing between hypervisors. Data link layer is shared between all hypervisors and then virtual machine is located in same L2 network before and after migration. This solution does not scale well since it is not recomended to place more than a few hundreds of hosts into single L2 domain, so it may be necessary to divide single big L2 into many smaller networks. It is quite easy to employ this solution for hypervisors in same rack, but it is more difficult with more distant servers and even more when servers are located at different data centers.

Another way how to accomplish unlimited VM migrations is to use routinl for machine connectivity. This method uses temporary and fixed IP addresses. Hypervisor do not have to be in same L3 network and there is higher variability in addresses assigned to virtual machines. Temporary address is assigned according to VM's location and it changes during migration. Fixed address is routed to VM and this address does not change, because changes are made only in routing tables. Any routing protocol can be usel, e.g. OSPF, to provide correct routing of fixed address destination virtual machine. Is is neccesary to insert one record in routing table for every virtual machine, because \32 routes are being advertised. Huge routing table may be problem for data centers with many virtual machines. Higher layer is used to get more flexibility than lower level can offer. Main drawback of this solution is additinal complexity caused by routing and longer address swap, because it takes some time to propagate routing to new temporary address. It is not easy to perform live migration, because it is neccesary to change IP address of VM's interface and thus open sessions will terminate.

2.3.1 Overlays

Virtualization is used heavily these days, therefore it is necessary provide networking solution with at least same flexibility as virtualization offers. Multitenancy, VM migrations, fast reconfiguration and rapid deployment are most missing features of physical networks these days. It is currently possible to migrate virtual machines without service interruption, but there is still not clear solution how to perform migration across whole data center with maintaining IP address. Overlay networking is one of proposed solutions and it is supposed to bring additional abstraction layer capable of decoupling network from physical hardware. Technologies capable to build overlay network are VXLAN, STT and NVGRE.

Data center network need to be robust enough so parallel paths are used to provide redundancy and avoid outage caused by single link failure. It is necessary to avoid loops on L2 network because there are loops caused by redundant paths and L2 network does use anything like TTL field. Spanning Tree Protocol (STP) can be used for avoiding loops in L2 networks, but there are two mayor problems. First is need to adjust STP if VLANs are used because it is necessary to build special tree for each VLAN. Second problem with using STP is utilization of parallel links. STP keep only one of parallel link running and the others are disables to avoid topology loops. It is not optimal solution because links utilization is low and it is impossible

to increase connection bandwidth by adding parallel links. Upgrading to higher link speed is limited and does not make economical sense. 10G cards are becoming affordable, but 40G cards are still expensive.

Servers housed in rack are usually connected to switch called ToR and this switch is learning their MAC addresses. Common ToR switch have 24 or 48 ports so it should be able to learn addresses of these serves. However virtualization techniques let us to run many virtual machines on single physical server so number of MAC addresses can increase significantly. Amount of address may grow even more because each virtual machine can have more than one interface and ToR switch must be capable of learning all addresses.

Public cloud solutions tend to serve many tenants and it is necessary to avoid unwanted interaction between them. It is quite easy to guarantee this on virtualization layer but much harder to accomplish on network layer. Tenant isolation can be performed on Layer 3 or Layer 2. VLANs are often used for isolation on Layer 2, but this solution suffer from insufficient scaling issues because only 12 bit VLAN identifier is used and it provides only 4096 different tags. This might be enough for smaller solution but it is not sufficient for huge cloud system. Each physical server can host up to 100 virtual machines/containers[2] owned by different tenants so 1 server can consume about 100 VLAN tags. All available tags can be depleted by 40 high performance servers which can fit in just one rack. Isolation at Layer 3 is does not provide sufficient scaling as well. It is necessary to provide unlimited migration facility between hypervisors in different racks and sometimes is required to spread Layer 2 between all tenant's virtual machines. Layer 3 isolation is not capable of this.

VXLAN

Virtual eXtensible Local Area Network is an overlay scheme with multitenancy and domain isolation. It is defined on Layer 3 and uses encapsulation as tunneling mechanism.

Most important think is encapsulation since it provides VXLAN domain isolation and defines overlay network. Block called VTEP is responsible for encapsulation and tunnel organization. It analyzes every packet received from VM and prepend outer header with label. This label is called VXLAN Network Identifier (VNI) and it is used to isolate domains so virtual machines in different domains are not allowed to communicate directly with each other. Encapsulated packet is send to destination VTEP as UDP packet. Destination VTEP unpacks packet, check whether there is any virtual machine in VXLAN domain and deliver this frame. NVE is another term for VTEP and it was introduced in [**?**] as part of general network virtualization framework.

It is necessary for VTEP to be able to find destination VTEP for every encapsulated packet. This can be solved by data plane learning during forwarding as specified in [6] or by acquiring this information from orchestrator. Getting information from orchestrator in my opinion much better because it avoids additional actions. Some kind of orchestrator or at least information system must be present in every data center and this system already knows location of all virtual machines

[2]Server node2.brg/vpsfree.cz is currently running 117 VPSs with hardware: Supermicro X9DR3-F, 2x Xeon E5 2630Lv2, 256GB DDR3, 8x 2TB WD2002FAEX, 2x Intel DC S3700 200GB

as well as addresses of VTEPs. I think that it does not make sense to perform learning during forwarding because required informations are already saved in orchestrator. Orchestrator can directly distribute forwarding rules to all VTEPs or VTEP can ask orchestrator on-demand via any kind of API. Overlay unicast traffic can be forwarded directly to destination VTEP without any additional learning or even flooding.

There is traffic called BUM - Broadcast, Unknown unicast and Multicast which not easy to handle by VXLAN. This kind of traffic needs to be delivered to more than one host in a single VXLAN domain and thus it is necessary to send encapsulated packet to many VTEPs at the same time. Multicast should be used ad described in [6], but it needs mapping between VXLAN VNI and multicast address. This mapping should be managed by orchestrated by management layer. It would be beneficial to have technology for delivering BUM traffic without need of multicast because it brings additional complexity and it is not allowed in global Internet. Unknown unicast can be mitigated with getting information about addresses from orchestrator as described in previous paragraph because the orchestrator knows all addresses and there will no longer exists any unknown traffic. Sending encapsulated broadcast and multicast to many VTEPs can be achieved by multicast in underlay network or any advanced delivery methods can be uses. It is possible to send this traffic as unicast between source VTEP and every other VTEP. However this solution is suboptimal thanks to packet duplication and higher network bandwidth usage. Only one advantages is that multicast in underlay network is not required. It is also possible to select on node as "router" for encapsulated VXLAN traffic between VTEPs but it is technically similar to building multicast tree and thus it might be easier to deploy multicast in underlay. There is proprietary technology called IBM DOVE with is very similar to VXLAN but does not require multicast.

I think thatVXLAN is quite promising technology for network virtualization in data center. It brings much more flexibility that traditional VLAN approach and it can be called as en evolution of VLAN. Principle of overlay network is building virtual network on top of physical infrastructure. Benefits were described in previous articles and main drawback is lack of cooperation between overlay and underlay network. Multicast might be also quite problematical to arrange.

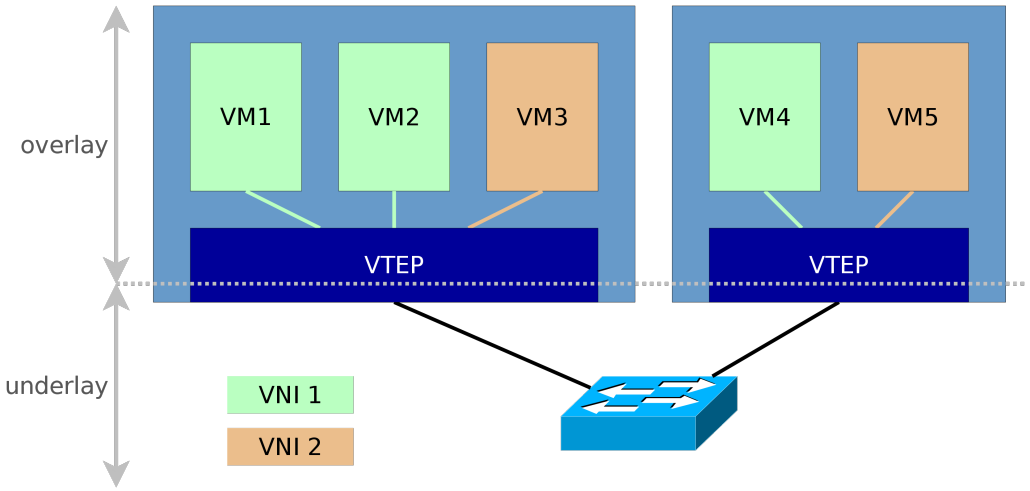

Figure 2.3.2: Model of VXLAN topology

NVGRE

Network virtualization using GRE is another overlay technology for multi-tenant data centers. It is very similar to previously mentioned VXLAN because it uses same topology scheme with different encapsulation mechanism. I am not going to provide detailed description of protocol as with VXLAN but main differences will be mentioned. Further information and current draft version can be found in [7] [5] [4].

The biggest difference is encapsulation mechanism. VXLAN uses own approach but NVGRE uses GRE. Using quite old encapsulation standard may be beneficial because some boxes already support it and there is no need to make significant changes to physical infrastructure. Details about encapsulation mechanism can be found in [5] and [4] describes header extensions used to carry VSID. VSID is identifier for virtual subnet isolation, it is analogue of VNI used by VXLAN with same length of 24 b.

Outer header of packet with encapsulated payload is sent to destination NVE thus usage of ECMP may be suboptimal. There may be lack of entropy in outer header because destination address is same for all virtual machines residing at same NVE. This problem should be solved in ECMP hashing procedure by integrating VSID into sources for hash generation.

Multicast and broadcast traffic within overlay network is handled using multicast in underlay network. There is also defined N-Way unicast which do not depend on multicast: "In N-Way unicast, the sender NVE would send one encapsulated packet to every NVE in the virtual subnet. The sender NVE can encapsulate and send the packet as described in the Unicast Traffic Section 4.3. This alleviates the need for multicast support in the physical network." [7] However this solution is suboptimal because there is unwanted packet duplication and thus it is better to deploy multicast and use it as carrier mechanism.

Definition of NVGRE [7] is still labeled as draft, last version was published on 2014-11-05 and new updates are expected. Proposed draft is simple and there are still mayor problems waiting to be solved. For example there is not any method how to distribute locations of addresses within overlay network. Document [7] says: "This information can be provisioned via a management plane, or obtained via a combination of control plane distribution or data plane learning approaches. This document assumes that the location information, including VSID, is available to the NVGRE endpoint." It is obvious that this need to be solved before deployment in production use.

STT

Last but not least technology used for building overlay network is Stateless Transport Tunneling (STT). It is designed to meet common requirements as allow overlapping of tenant's address space, decouple virtual network from physical infrastructure and allow unlimited virtual machine migration.

Basic principle is still same - some box (usually called NVE) encapsulates packets from overlay network and send it through underlay network to other NVEs. However STT introduces completely new encapsulation method. TCP-like header is used as an encapsulation header but there is no three-way handshake or sessions because packets are processed different way. Header is used only as a storage for metadata about encapsulation. Field called Context Identifier is assigned to every flow and

it is used as a generalized form of virtual network identifier. [3] It is beneficial to use this generalization because there is space for future services. Space reserved for Context Identifier is 64 bits long so there is really enormouns amount of combinations available.

It is important for every overlay technology to support ECMP because efficient flow distribution between multiple paths can be used for underlay network in data center. First important requirement is to route each packet belonging to single flow same way and it is accomplished by using same ports and addresses for these packets. Second requirement is to provide enough of entropy for uniform flow distribution. Packet's source port is function of inner header and thus it provides entropy data for ECMP mechanism.

Using almost standard TCP segmentation for encapsulation is advantageous because it may bring significant performance improvement. Segmentation offloading is heavily used these days so it can be used to speed-up encapsulation process. The most important advantage of STT is providing new functions and using of existent hardware techniques.

However I can see some problems with deploying STT. First and the most important is changing meaning of TCP header field since this will probably cause problems in middle boxes. It will be necessary to adjust configuration of state firewalls to allow STT because TCP headers are expected to behave different than is is used. Defining document [3] is still in draft version and it is already expired. Last version is #06 at time of writing this paragraph (2014-13-11) and this version expired on 2014-10-17. It is possible that this technology will be used in future but I do not think it is going to be used as overlay technology very much these days.

2.3.2 Hop-by-hop network virtualization

It is possible to use new technologies, e.g. SDN, and build different kind of virtual network called hop-by-hop. Hop-by-hop virtual network is totally different than previously described overlay networking since it does not use any encapsulation and data path is established by joining independent links between hops.

Hop is responsible only for forwarding data unit to next hop and whole flow is directed by a controller. Controller is software appliance responsible for communication with physical boxes, distributing routes and analyzing packets received from forwarding plane. It is usually tightly collaborating with orchestrator.

There is different perspective on network since control plane is separated from forwarding plane so physical devices are used only for fast packet transfers and data plane is responsible for network control. Every decision is performed in controller or orchestrator and propagated to forwarding plane through data plane. It is obvious that the orchestrator should not be physically centralized because it would create single point of failure so it is better to use any distributed solution.

Hop-by-hop networks are tightly connected with topic called Software defined networking described in 2.3.5.

2.3.3 Load balancing and high availability

Load balancing is an essential part of service operation because there it is required to improve scalability and availability better than single machine approach

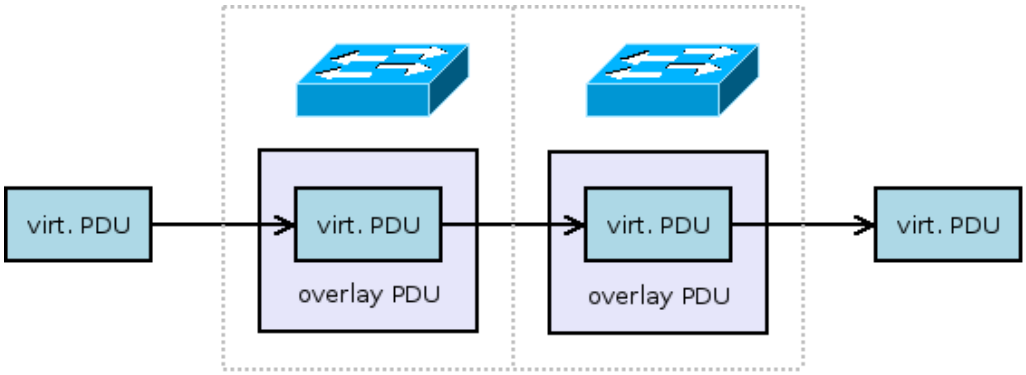

Figure 2.3.3: Overlay virtual network

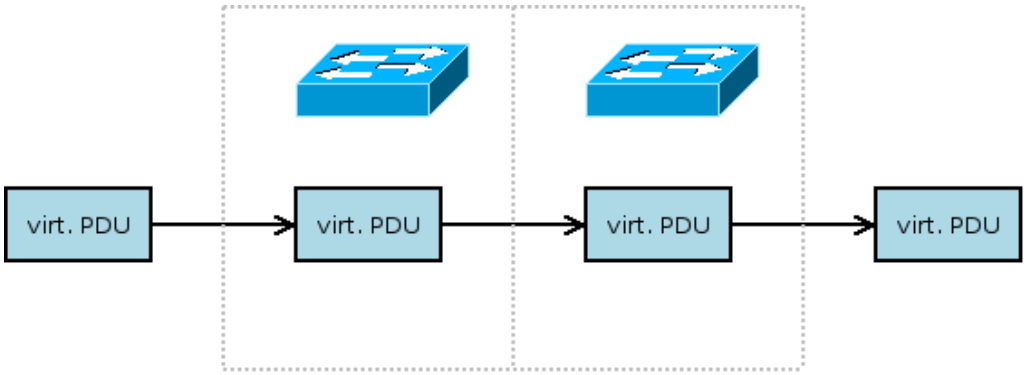

Figure 2.3.4: Hop-by-hop virtual network

can ever achieve. It quite common in the past that one service had been served just by one machine. However this solution is suboptimal since it is totally unscalable and it is impossible to provide high-availability solution.

It is necessary to employ service load balancing because load and service demand is still increasing and only properly designed load balancing solution can meet all of the requirements. Common requirements are

- low latency - request should be processed without any significant delay
- high availability - service should stay up during partial infrastructure failure
- scalability - infrastructure should be ready to increase resource when the load is enormous

There are many different ways how to deploy load balancing and they differ by flexibility and functions available. We can be say that load balancing performed at higher levels is more flexible, but the best solution are combination between two or more technologies on different layers. Appropriate solution depends also on access method because there are different balancing possibilities for example for HTTP API, remote terminal service and video streaming service. Method presented in following text will be general as well as access method specific, but right use case will be always mentioned.

Load balancing is closely related with scaling. There are two types scaling - scaling-up and scaling-out. Scaling-up is accomplished by using more powerful resources, e.g. using interfaces with higher line rate or upgrading server. It is easier, achievable faster and does not require load balancing, but it is quite easy to reach limit of scaling-up. 1G Ethernet NICs are very common, 10G are a bit more expensive but still possible to buy and 100G are very rare and extra expensive. It is also necessary to take economical aspect into account since performance improvement and price function are not equally steep. Scaling-out is other possible scaling schema and it is accomplished by adding many parallel workers with common capacity. This approach is more favourable from economical view because performance growth is almost linear and technical benefit lies in redundancy. However it is necessary to use load balancing to distribute workload across nodes. I think that best scaling solution lies somewhere between so I would recommend to slowly scale-up and use scale-out for massive increase of performance.I think that best scaling solution lies somewhere between so I would recommend to slowly scale-up and use scale-out for massive increase of performance.

Load balancing methods can be divided into two group whether they provide session persistence or not. Session persistence mean that one client is always routed to same computing node. It is required if there is a client's information, called session, available only on this computing node and session would be lost in case of redirecting to another node. Application can be designed with taking load balancing into account and thus it does not require session persistence. However session persistence is usually needed for load balancing of services designed without load balancing capabilities.

DNS based approach

DNS load balancing is first possible solution because it is take place before establishing session. It is relative easy to deploy and application redesign may not be necessary. Basic implementation can be round-robin DNS which is carried out by assigning many AAAA or A records for service host name. Client selects one record after resolving host name and use it thus basically performs load balancing already at user's device. This method is really simple but lacks any advanced management options. First problem is with high availability because it is not possible to quickly remove host from zone in case case of failure. There is field called TTL assigned to every record in zone and this field defines how long can be this record cached, maximum time between change in zone and propagation to all client should be TTL a SOA. However there are Internet Service Providers ignoring this standard so it is possible that some client will still get wrong records even after TTL expiration. Sample zone file with AAAA and A records and TTL 6 minutes is in figure 2.3.5.

More advance variant of DNS based load balancing is modification of zone performed by authoritative DNS server. There is usually just one AAAA/A record for service hostname, but returned IP address can be different for every query. This method may use geolocation and return IP address of the nearest server according to user's position, however user can use different recursive servers and geolocation can be very inaccurate. Technically this method is only variation of previously mentioned with better control of distribution and there is still same problem with TTL. This method is used by web portal Seznam.cz for load balancing between primary

and secondary data center. They use TTL 5 minutes and also experienced problem with incorrect caching however I am not allowed to publish any detailed information.

Another problem with DNS load balancing, especially failover, is DNS pinning. It is mechanism implemented in web browser to make DNS rebinding attacks more difficult. This attack is based on pushing faked DNS record to client and then forward all traffic to attacker's IP address. Browser with pinning implemented "pins" first resolved IP address and use it even after TTL expiration so it basically prevent load balancing mechanism to switch client to another computing node. Further information can be found in [13].

Figure 2.3.5: Example zone file for DNS load balancing

```
app.example.com    360    IN    AAAA    2001:db8::1
app.example.com    360    IN    AAAA    2001:db8::2
app.example.com    360    IN    AAAA    2001:db8::3
app.example.com    360    IN    A       192.0.2.1
app.example.com    360    IN    A       192.0.2.2
app.example.com    360    IN    A       192.0.2.3
```

Application level load balancing

One of the most flexible method is application load balancing. Is is performed on Layer 7 so it is possible to differentiate in all lower layers. This solution is beneficial because application is able to decide on exact mapping between customer connection and working node. Customer is connected to balancing part of application at first. This part (group of nodes) is responsible for redirecting or forwarding request to computing node. It is possible require login before redirecting and then forward request according to information required. Every information about customer is already available, like IP address and login name, so computing node can be selected and it is also very simple to achieve session persistence. Balancing procedure is depicted in figure 2.3.6.

Advantage of this method is direct connection between client and computing node, so balancing part is not overloaded with translating requests between users and computing nodes. Direct connection eliminates bottlenecks because there is not any central authority responsible for load balancing. Technically there is central authority in load balancing part, but it can be redundant and balanced using other method, e.g. DNS load balancing. However it is necessary to expose computing nodes to users network and thus some may say that is insecure. I think that exposing computing nodes to outside word is not security hazard, because security should be provided by proper application design and network security. Obscurity is not good security approach in my opinion.

Anycast load balancing

It is possible to use anycast routing for load balancing and distribute workload between nodes. Model situation is depicted at figure 2.3.7 Only anycasted IP address is propagated to outside world, so every incoming packet have this destination

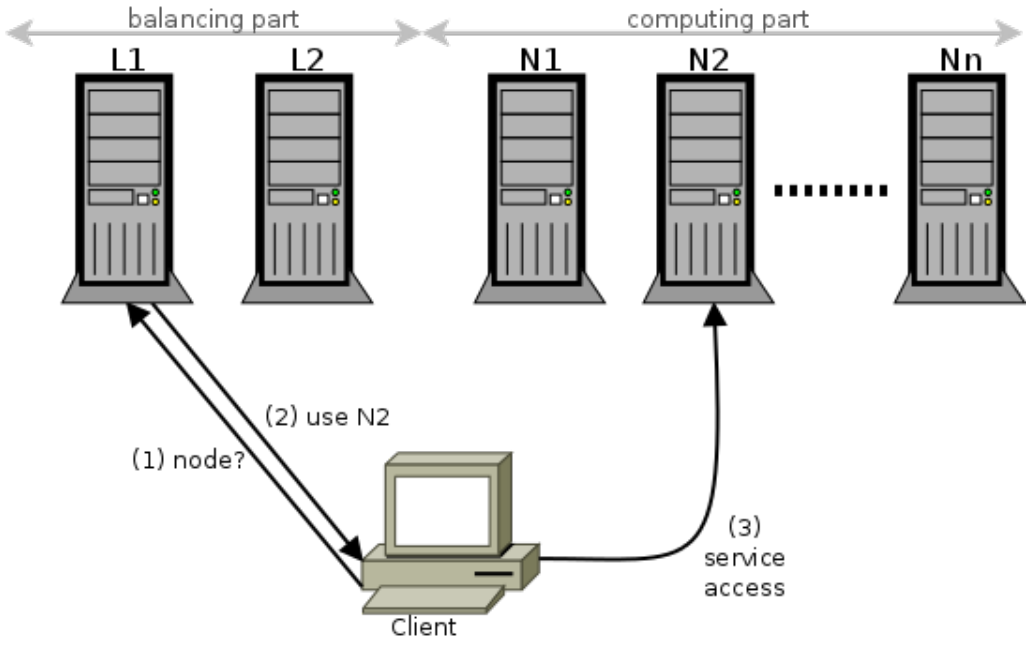

Figure 2.3.6: Load balancing at application level

address. This address is also assigned to local interfaces of computing nodes and advertised to local router using any routing protocol, e.g. OSPF.

An incoming packet is delivered to local router and this router performs lookup and selects destination address according to it's actual routing table. This is advantageous because it is possible to assign priority to routes propagated by computing nodes and node is almost immediately remove from routing table in case of node failure.

However this solution is not capable to provide session persistence because packet can be routed to different computing node every time. There would be a bottleneck in topology described in figure 2.3.7 but this method can be adjusted to eliminate this problem and propagate different anycast addresses from different autonomous systems.

Global pool of root DNS servers use exactly this load balancing principle so request should always be delivered to the nearest server and thus almost perfectly distributed around word servers. According to data published in [15] up to 80% of DNS queries are routed to the nearest anycast instance.

2.3.4 Firewall

2.3.5 Software Defined Networks

2.4 Storage

2.5 Orchestration software

2.5.1 OpenNebula

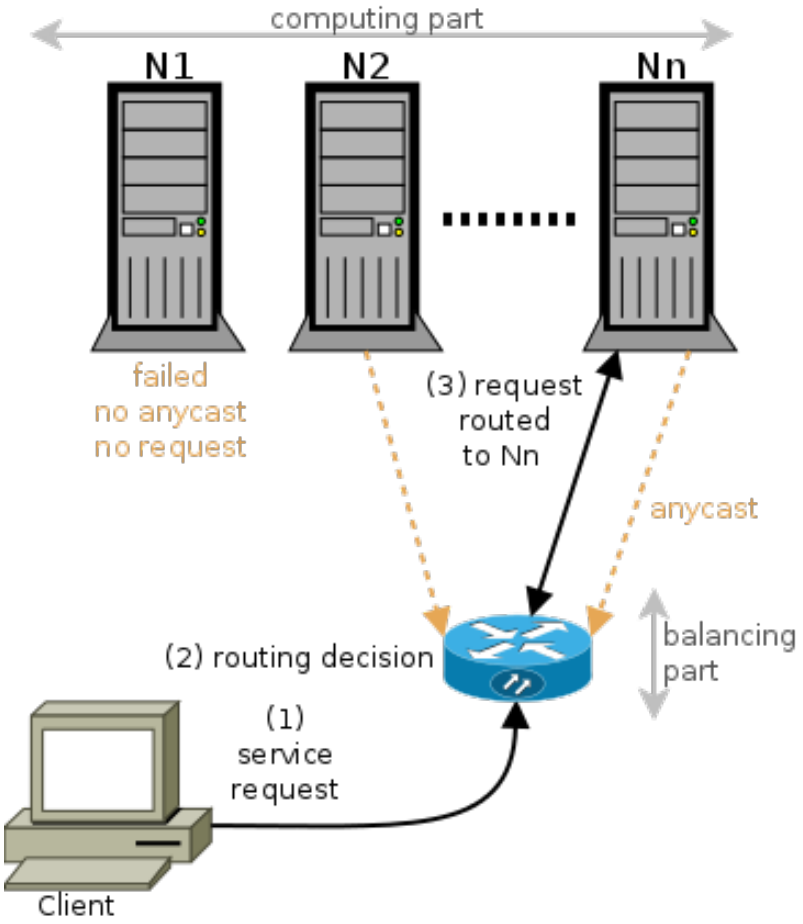

Figure 2.3.7: Anycast load balancing

Migration of virtual machines

Distributed data center

Part II

Practical

Methodology overview

Framework

Results

List of Abbreviations

API	Application Programming Interface.
BUM	Broadcast, Unknown unicast and Multicast.
CAPEX	Capital Expenditures.
CPU	Central Processing Unit.
DNS	Domain Name Service.
ECMP	Equal-cost multi-path routing.
GRE	Generic Routing Encapsulation.
HTTP	HyperText Transfer Protocol.
IaaS	Infrastructure as a Service.
IOPS	Input/Output Operations Per Second.
IP	Internet Protocol.
IPMI	Intelligent Platform Management Interface.
IPv4	Internet Protocol version 4.
IPv6	Internet Protocol version 6.
ISATAP	Intra-Site Automatic Tunnel Addressing Protocol.
IT	Information Technology.
KVM	Kernel-based Virtual Machine.
LXC	LinuX Containers.
MAC	Media Access Control.
NIC	Network Interface Card.
NIST	National Institure of Standards and Technology.
NVE	Network Virtualization Edge.
NVGRE	Network Virtualization using Generic Routing Encapsulation.
OPEX	Operating Expenditures.
OS	Operating System.
OSPF	Open Shortest Path First.
PaaS	Platform as a Service.
RIR	Regional Internet Registry.
SaaS	Software as a Service.
SAN	Storage Area Network.
SDN	Software Defined Networking.
SOA	Start Of Authority.
STP	Spanning Tree Protocol.
STT	Stateless Transport Tunelling.
TCP	Transmission Control Protocol.
ToR	Top of Rack.
TTL	Time To Live.
UDP	User Datagram Protocol.

UPS	Uninterruptible Power Supply.
US	United States.
VLAN	Virtual Local Area Network.
VM	Virtual Machine.
VNI	VXLAN Network Identifier.
VPS	Virtual Private Server.
VSID	Virtual Subnet Identifier.
VTEP	VXLAN Tunnel Endpoint.
VXLAN	Virtual Extensible Local Area Network.

List of Figures

List of Tables

Bibliography

[1] Ondřej Celetka. IPv4 jako služba aneb jak síť zbavit dual-stacku. http://www.root.cz/clanky/ipv4-jako-sluzba-aneb-jak-sit-zbavit-dual-stacku/. [Online; retrieved 2014-09-30].

[2] IBM Corporation. Virtualization in education. http://www-07.ibm.com/solutions/in/education/download/Virtualization%20in%20Education.pdf, 2007. [Online; retrieved 2014-09-17].

[3] Davie and Gross. A stateless transport tunneling protocol for network virtualization (stt). http://tools.ietf.org/html/draft-davie-stt-06. [Online; retrieved 2014-11-12].

[4] G. Dommety. Key and sequence number extensions to gre. http://tools.ietf.org/html/rfc2890. [Online; retrieved 2014-08-10].

[5] Farinacci et al. Generic routing encapsulation (gre). http://tools.ietf.org/html/rfc2748. [Online; retrieved 2014-08-10].

[6] Mahalingam et al. Virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks. http://tools.ietf.org/html/rfc7348. [Online; retrieved 2014-11-10].

[7] Sridharan et al. Nvgre: Network virtualization using generic routing encapsulation. http://tools.ietf.org/html/draft-sridharan-virtualization-nvgre-06. [Online; retrieved 2014-11-11].

[8] A. Hammadi and L Mhamdi. A survey on architectures and energy efficiency in data center networks. *Computer Communications*, 40, 2014.

[9] Chris Horne. Understanding full virtualization, paravirtualization, and hardware assist. http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf. [Online; retrieved 2014-08-20].

[10] Dustin Kirkland. Docker in ubuntu, ubuntu in docker. http://blog.docker.com/2014/04/docker-in-ubuntu-ubuntu-in-docker/. [Online; retrieved 2014-09-20].

[11] T. Mell, P. Grance. The NIST definition of cloud computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf. [Online; retrieved 2014-08-17].

[12] Ellen Nakashima. Judge orders microsoft to turn over data held overseas. http://www.washingtonpost.com/world/national-security/judge-orders-microsoft-to-turn-over-data-held-overseas/2014/07/31/b07c4952-18d4-11e4-9e3b-7f2f110c6265_story.html. [Online; retrieved 2014-09-12].

[13] B. Radha and S. Selvakumar. Deepav2: A dns monitor tool for prevention of public ip dns rebinding attack. In *Advances in Recent Technologies in Communication and Computing (ARTCom 2011), 3rd International Conference on*, pages 72–77, Nov 2011.

[14] M. Townsley S. Tsuchiya, Ed. and S. Ohkubo. IPv6 rapid deployment (6rd) in a large data center. http://tools.ietf.org/html/draft-sakura-6rd-datacenter-04. [Online; retrieved 2014-05-30].

[15] S. Sarat, Vasileios Pappas, and A. Terzis. On the use of anycast in dns. In *Computer Communications and Networks, 2006. ICCCN 2006. Proceedings.15th International Conference on*, pages 71–78, Oct 2006.