VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

VZDÁLENÉ PŘIPOJENÍ NA VIRTUALIZOVANÝ OPERAČNÍ SYSTÉM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MAREK VESELÝ
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

VZDÁLENÉ PŘIPOJENÍ NA VIRTUALIZOVANÝ OPERAČNÍ SYSTÉM

REMOTE CONNECTION TO VIRTUALIZED OPERATING SYSTEM

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

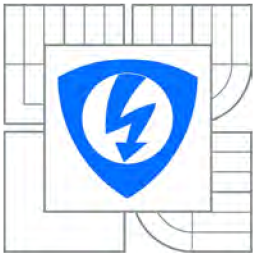
AUTOR PRÁCE — Bc. MAREK VESELÝ

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. DAN KOMOSNÝ, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Marek Veselý ***ID:*** 134658
Ročník: 2 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Vzdálené připojení na virtualizovaný operační systém

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s principem virtualizace operačních systémů. Seznamte se s nástrojem SSH (Secure Shell) pro textové připojení na vzdálené stanice. Proveďte virtualizaci operačního systému Linux, distribuce CentOS. Konfiguraci virtualizace upravte tak, aby bylo možno se vzdáleně připojit na virtualizovaný operační systém. Porovnejte parametry vzdáleného připojení na operační systém bez virtualizace a s virtualizací.

DOPORUČENÁ LITERATURA:

[1] Linux Dokumentační projekt. Computer Press, 2008. ISBN: 978-80-251-1525-1.
[2] DOSTÁLEK L. et al.: Velký průvodce protokoly TCP/IP: Bezpečnost. 2. aktualizované vydání. Computer Press, 2003. ISBN 80-7226-849-X.
[3] COOPER, M. Advanced Bash-Scripting Guide. Lulu.com, 2010. ISBN: 978-1435752191.

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 26.5.2015

Vedoucí práce: doc. Ing. Dan Komosný, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práce je zaměřená na nový trend, kterým je virtualizace. V této práci je rozebrána virtualizace operačního systému z pohledu poskytování síťových služeb. Dále je zde popsán protokol TCP, protokol ICMP a služba SSH. V poslední části se práce věnuje samotnému měření, kde se zjistí výhody a nevýhody virtualizace.

KLÍČOVÁ SLOVA

virtualizace, protokol, SSH, TCP, ICMP, odezva SSH, VMware vSphere, Virtualbox, KVM, Hyper-V, PlanetLab

ABSTRACT

The thesis is focused on the new IT trend – virtualization. Below are described the types of virtualization, as well as different implementations of it. In the following chapters there will be described the TCP and ICMP protocols, as well as the SSH services. The last part of this work is dedicated to measurements where there will be highlighted the advantages and disadvantages of virtualization.

KEYWORDS

virtualization, protocol, SSH, TCP, ICMP, delay of SSH, VMware vSphere, Virtualbox, KVM, Hyper-V, PlanetLab

VESELÝ, Marek *Vzdálené připojení na virtualizovaný operační systém*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 60 s. Vedoucí práce byl doc. Ing. Dan Komosný, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Vzdálené připojení na virtualizovaný operační systém“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Danu Komosnému, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM KÓDŮ

ÚVOD

V posledních několik letech se stále více datových center snaží ušetřit náklady na vynaložené energie. Což je z většiny způsobeno velkým počtem hardwarových serverů, který se snaží omezit dnes populární technologie virtualizace. Tato práce je z velké části zaměřená přímo této technologii a jejím řešením v reálném provozu.

Při vzdáleném připojení pomocí SSH služby na virtualizovaný operační systém vzniká veliké zpoždění. V práci se tedy snažím zjistit, jak velký vliv má počet virtuálních stanic na komunikaci pomocí SSH protokolu. To vedlo k sepsání dalších tří kapitol, kde jedna se věnuje funkci služby SSH a obecně protokolu samotnému. Další kapitola se věnuje TCP protokolu, jehož znalost pomohla k zachytávání komunikace, která poté určovala velikost zpoždění. Třetí kapitola se věnuje ICMP protokolu, který používá nástroj ping.

V poslední části už zde jsou prezentovány výsledky měření pro dva typy různých měření, kde první se věnuje měření latence z portu 22 a druhé, které se věnuje změření času od zahájení SSH komunikace až po její skončení. Tyto typy měření jsou provedeny pro tři virtualizační metody VMware vSphere, Virtualbox a KVM. Moje práce je nejvíce zaměřená na metodu KVM, jelikož to je virtualizace, kterou používá síť PlanetLab a hlavní popud pro sepsání této práce bylo právě zpoždění vznikající při vzdáleném připojení, v síti PlanetLab, na virtualizovaný operační systém.

1 VIRTUALIZACE

V posledních několika letech vzhlíží datová centra na celém světě k virtualizačním technologiím. Hlavními stimuly jsou snížení uhlíkové stopy a provozování klasických serverových systémů, které jsou využity jen na deset či méně procent, tudíž spotřebovávají zbytečnou energii, potřebují prostor a dostatečné chlazení. Virtualizace nabízí možnost využít servery na 80procentní nebo vyšší využití hardwaru se zachováním stejné zátěže na slabším hardwaru.

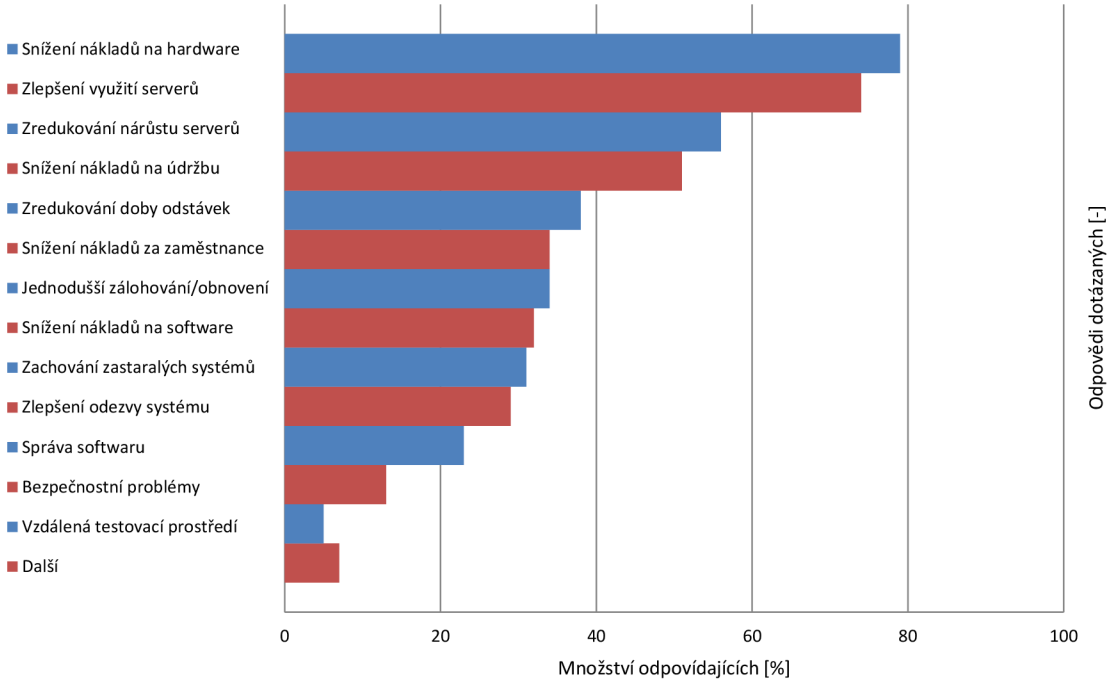
Za používání serverů jen na desetinu procent jejich využití je obviňována společnost Microsoft. V devadesátých letech minulého století se stal populárním systém Windows NT Server. Postupným testem tohoto serverového operačního systému správci zjistili, že Windows NT byl monolitickým operačním systémem. Mnoho operací, zvláště aplikačních, bylo prováděno na úrovni jádra operačního systému, tudíž když aplikace zamrzla, tak často zamrzl celý operační systém a způsobil chybu BSOD[1]. Díky tomuto začali lidé vytvářet pouze jednoúčelové servery, které spouští pouze jednu aplikaci. Tento přístup se rychle roznesl i mezi koncové uživatele a začala jej používat většina firem. Postupem času společnost Microsoft vyřešila problém svého monolitického systému, ale návyky uživatelů se s tímto operačním systémem nezměnily. Což zapříčinilo masivní nárůst počtu serverů, který v posledních letech vzniká [1].

Od šedesátých let minulého století společnost IBM začala vytvářet oddíly na svých počítačích, aby hostily více instancí jejich operačního systému. Tuto myšlenku zpopularizovala společnost VMware během devadesátých let. Společnost představila produkt VMware Workstation, což byl produkt zaměřený přímo pro koncové uživatele, který umožnil uživatelům spouštět libovolný počet instancí operačního systému architektury x86. Klíčem tohoto úspěchu bylo, na rozdíl od firmy IBM, že umožňuje spouštět celou řadu operačních systémů různých výrobců. Nyní je společnost VMware lídrem na virtualizačním trhu [1].

Podle průzkumu Ziff-Davis z roku 2008 existuje několik častých virtualizačních stimulů, viz obr. 1.1. K nejčastějším se řadí snížení nákladů na hardware a zlepšení využití serverů, což naznačuje, že serverová virtualizace je nejvýznamnějším aspektem pro přechod k virtualizaci. Nicméně toto není jediná vrstva datového centra, kterou je možno virtualizovat [1].

V poslední době náklady na energie a chlazení serverů v datovém centru exponenciálně rostou a je potřeba je kontrolovat. Při správné konfiguraci stejného fyzického serveru pro serverovou virtualizaci je možné zde spustit více než deset virtuálních počítačů, serverů nebo desktopů. Přitom žádný z těchto virtuálních počítačů nepotřebuje své vlastní napájení, nevytváří vlastní teplo ani nevyžaduje žádné vlastní

[1] Blue Screen of Death, čili modrá obrazovka smrti



Obr. 1.1: Graf závislosti typů odpovědí dotázaných na jejich počtu [2].

místo. Přitom všechny tyto virtualizované počítače mohou nabídnout stejné služby jako klasické fyzické servery. Všichni manažeři datových center souhlasí s tím, že [1]:

- fyzické servery zabírají příliš mnoho místa
- fyzické servery jsou málo využity, někdy jen na 5 až 15 procent
- fyzické servery vytváří příliš mnoho tepla
- fyzické servery potřebují stále více energie
- fyzická datová centra potřebují složitější řešení odolnosti proti výpadku
- doba výpadku je trvalý problém, který je potřeba odstranit
- fyzické servery mají složitou správu a je potřeba ji zjednodušit.

1.1 Výhody virtualizace

Mezi výhody serverové virtualizace patří [1]:

- První z mnoha výhod spočívá v úrovni nasazení. Virtuální počítač je možné nasadit a přizpůsobit za krátkou dobu.
- Mobilita virtuálního počítače je další výhodou, díky které můžeme virtuální počítač kdykoliv přesunout z jednoho hostitele na jiného. Někdy je možné je přesunout i za běhu, což je ohromná výhoda, která pomůže snížit dobu výpadku v síti.

- Virtuální počítač se snadno používá, když je jednou vytvořen a nakonfigurován, tak jej stačí jen spustit a ihned je připraven poskytovat služby.
- Podporují běžné konfigurace. Stačí vytvořit běžný virtuální počítač a poté z něj zkopírovat zdrojové soubory tohoto virtuálního počítače pokaždé, když potřebujeme nový počítač. Tento postup nám zajistí vždy standardní konfiguraci všech virtuálních počítačů.
- Podporují koncept nestálých služeb. Když je potřeba spustit určitou sérií testů, tak je možné spustit nový virtuální počítač, využít jej na provedení testů a po jejich dokončení jej lze jednoduše smazat. Podobné testy s fyzickými počítači se jen těžko aplikují.
- Mohou být certifikovány výrobcem virtualizačního produktu, což zaručuje, že virtuální počítače využívají nejlepší funkce jejich technologií.
- Virtuální počítače jsou bezpečnější než klasické fyzické servery. Lze je kdykoliv zcela izolovat. Stačí pouze přerušit jejich komunikaci pomocí virtualizační technologie daného hostitele.
- Výkon virtuálních počítačů je škálovatelný. Pro zmenšení výkonů stačí jednoduše vytvořit více virtuálních počítačů se stejnými službami. Naopak pro zvýšení výkonů stačí ukončit několik virtuálních počítačů a přiřadit mu více zdrojů, jako například více operační paměti RAM, disků, síťových karet či více jader procesorů.
- Jsou ideální pro zotavení po havárii, protože vše co se musí udělat, je zkopírovat veškeré soubory do jiného umístění, ať už v rámci zcela jiného místa nebo v rámci datového centra.

Kvůli zjednodušení správy desktopů se stále více podniků pouští do virtualizace desktopů. Tato virtualizace má často větší smysl a výhody než správa fyzických desktopů [1]:

- Centrálně spravované desktopy lze poskytnout uživatelům s libovolným koncovým zařízením, jako jsou tenké klienty, stolní počítače, webové klienty a další.
- Virtuální desktopy jsou centrálně řízeny, takže stačí vytvořit bitovou kopii desktopu a tu můžeme kopírovat tak často, jak potřebujeme.
- Není potřeba jakékoliv správy fyzických počítačů. Tyto počítače slouží pouze k zajištění připojení virtuálního desktopu ke vzdálené ploše.
- Můžete zde nastavit různé úrovně oprávnění. Na koncovém bodu mohou mít uživatelé oprávnění správce, ale virtuální desktop pro ně může být uzamčen. Což jim umožní dělat na koncových bodech, co chtějí, ale na podnikovém virtuálním desktopu je jejich chování pod kontrolou.
- Lze tak snížit náklady a zvýšit stabilitu, jelikož každý uživatel pracuje vzdáleně na virtuálním desktopu.
- Lze vytvářet časově řízené bitové kopie virtuálních desktopů, tak aby splnili

aktuální požadavky. Například v důsledku zvýšených sezónních potřeb lze vygenerovat bitové kopie osobních počítačů časově omezené trváním pracovního poměru.
- Lepší zabezpečení informací díky uchování virtuálního desktopu v datovém centru. Vytvořenou bitovou kopii stačí uzamknout, aby neměla přístup k externím zařízením.
- Na specifické virtuální počítače mohou být aplikovány složité nebo citlivé aplikace, čímž je zajištěna jejich správná funkčnost. Tyto aplikace se následně nemusí mezi sebou ovlivňovat, ani nemusí spolu koexistovat.

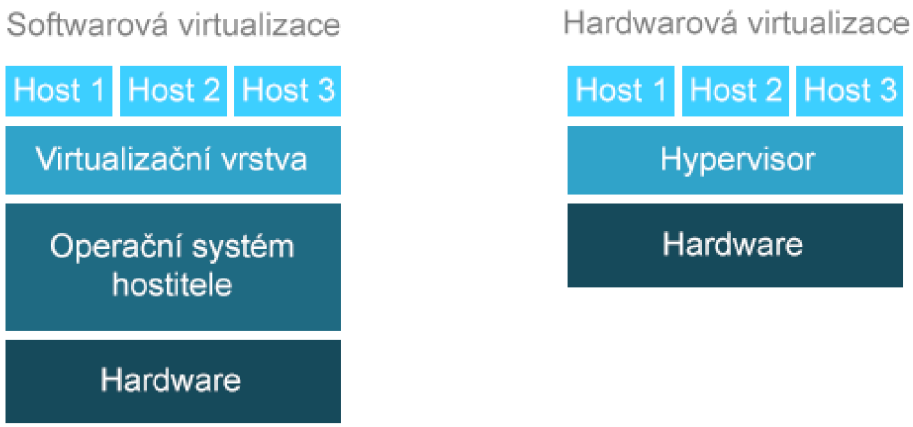
1.2 Typy virtualizace

V dnešní době je již virtualizace rozvinutá technologie, kterou lze využít na více vrstvách [1]:

- **Serverová virtualizace** – rozdělení fyzické instance operačního systému na virtuální instanci. Jsou zde dva aspekty serverové virtualizace:
  - **Softwarová virtualizace** – virtualizovaný operační systém je spuštěn nad softwarovou virtualizační platformou spuštěnou na existujícím operačním systému.
  - **Hardwarová virtualizace** – virtualizovaný operační systém je spuštěn nad softwarovou platformou přímo nad hardwarem bez existujícího operačního systému. Ke spouštění hardwarové virtualizace se používá engine označovaný jako *hypervisor*.
- **Virtualizace úložišť** – sloučení fyzického úložiště z několika úložišť takovým způsobem, aby se úložiště jevilo jako jeden fond úložišť.
- **Virtualizace sítí** – fyzická síť je rozdělena na několik logických sítí.
- **Správa virtualizace** – zaměřuje se na technologie spravující celá datová centra.
- **Virtualizace desktopů** – naskytuje možnost centralizovat nasazení desktopů a snížit tak náklady na správu. Uživatelé přistupují k desktopům pomocí různých tenkých či nespravovaných zařízení.
- **Virtualizace aplikací** – odděluje provozní aplikace od samotného operačního systému.

1.3 Serverová virtualizace

Doposud nejpoužívanější virtualizační technologií je serverová virtualizace. Když pomyslíme na ty tisíce dnes provozovaných serverů, které jsou ve většině případů



Obr. 1.2: Dva modely serverové virtualizace [1].

využity jen na méně než 5 procent. Každý z tohoto počtu serverů spotřebovává velké množství energie, což vyžaduje velké množství energie na chlazení a prostor v datových centrech všude na světě. Poté když si představíme, že při správné konfiguraci stejného fyzického serveru pro serverovou virtualizaci můžeme spustit více než deset virtuálních serverů. Tyto virtuální servery nám mohou nabídnout stejné služby, jako kdyby byly provozovány na fyzickém počítači, ovšem s tím rozdílem, že ušetříme energii a potřebné místo [1].

Každý virtuální počítač je jen množinou souborů někde na disku. Lze tedy vzít fyzickou instanci serveru a převést ji na virtuální instanci, tedy na množinu souborů ve složce. V tomto stavu již můžete tuto instanci přesunout z jednoho serveru na jiný, vypnout ji, restartovat, uspat a v podstatě dělat vše, co dříve bez jakéhokoli významného snížení výkonu.

Jak již bylo uvedeno v předchozí kapitole, existují dva virtualizační modely serverové virtualizace, viz obr. 1.2.

1.3.1 Softwarová virtualizace

K vytvoření prvního virtualizačního projektu se často využívá tento první model softwarové virtualizace, protože se spoléhá na jednodušší a v častém případě bezplatné technologie. Tato metoda je ovšem méně účinná, neboť vyžaduje alespoň základní operační systém hostitele. Tento systém také potřebuje zdroje, tudíž ovlivní provoz virtuálních počítačů běžících nad ním.

Softwarová virtualizace by se nikdy neměla použít v reálném provozu, pokud

nejsou sjednány smlouvy o poskytnutí úrovně služeb (SLA[2]) a můžeme si dovolit odstavit síť na virtuálních počítačích na delší dobu bez negativního dopadu na uživatele. Operační hostitelský systém podléhá procesu aktualizací stejně jako všechny virtuální počítače, jsou-li potřeba restarty, tak budou restartovány i všechny virtuální počítače [1].

1.3.2 Hardwarová virtualizace

V provozním prostředí se doporučuje před softwarovou virtualizací. U této virtualizace je kód hypervisoru přímo integrován do hardwaru a vystaví virtuálním počítačům hardware hostitelského serveru. Stane se tak pouze na úkor malého množství zdrojů hostitele, což umožní spuštění nejvyššího možného počtu virtuálních počítačů. Jelikož hostitel nezahrnuje běžný operační systém, tak nevyžaduje „záplatování“ ani aktualizace vůbec, nebo alespoň ne tak často jako běžný operační systém. Toto minimalizuje vliv hypervisoru na virtuální počítače, které hostí, a vytváří z tohoto typu virtualizace zdaleka nejlepší model, který lze použít pro serverovou virtualizaci [1].

Například VMware ESXi je hypervisor o velikosti 32MB, který nevyžaduje operační systém. Tento hypervisor dokáže běžet přímo z firmwaru nebo interního USB klíče. Tato úroveň integrace dává přímo možnost vytvořit nový model bezdiskového hostitelského serveru pouze s využitím operační paměti RAM, procesoru a síťových prostředků pro hostované virtuální počítače.

1.3.3 Přehled dostupných řešení

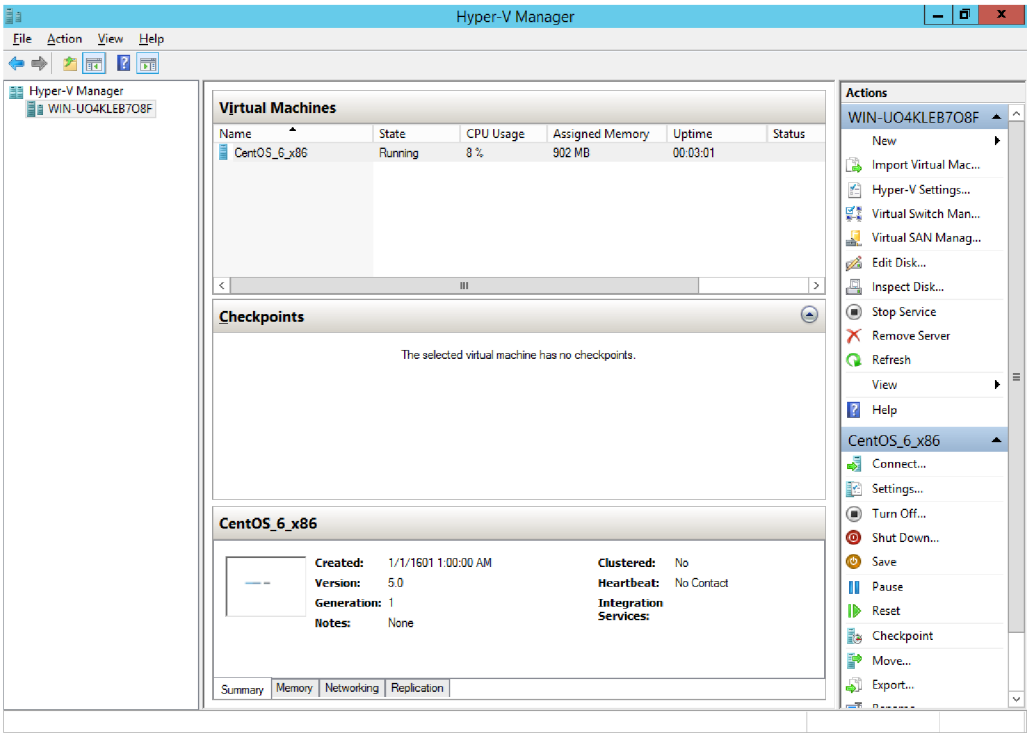
Výrobci produktů pro serverovou virtualizaci se to jen hemží, ale trh dobyli hlavní 3 výrobci:

- Společnost Citrix nabízí velké množství virtualizačních technologií, jednou z nich je produkt XenServer. Tento produkt je dodáván ve čtyřech verzích. Bezplatná z nich je Express Edition, což je základní verze produktu. Další verzí je Standard Edition, která podporuje dvě nabídky virtuálních služeb současně. Verze Enterprise umožňuje spouštět neomezený počet nabídek virtuálních služeb a přidává také možnost vytvářet fond hardwarových zdrojů. Dynamické přidělování hostitelů a nabídky virtuálních služeb přidává verze Platinum Edition. Dále společnost Citrix nabízí OEM[3] verzi svého hypervisoru, která je integrovaná přímo do hardwaru serveru [1].

[2] service-level agreement

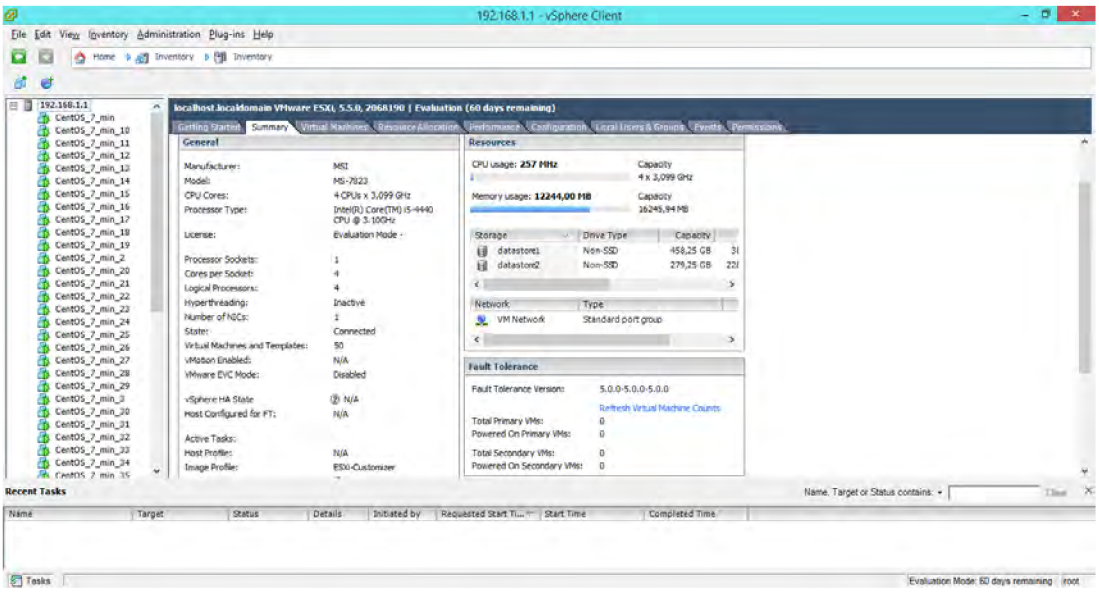
[3] Original Equipment Manufacturer

- Další společností je společnost Microsoft, která nabízí celou řadu virtualizačních technologií ve všech oblastech virtualizace. Ze softwarové virtualizace nabízí produkty Virtual PC a Virtual Server, které jsou bezplatné. Dále nabízí hypervisor obsažený v operačním systému Window Server 2008, Hyper-V, který běží pouze na hardwaru architektury x64. Od této doby je již vydávaný jako samostatný bezplatný produkt Hyper-V Server 2012 R2, viz obr. 1.3, což je v současné době nejnovější verze [3].



Obr. 1.3: Grafické uživatelské prostředí Hyper-V Manageru.

- Společnost VMware nabízí široké spektrum nástrojů pro virtualizaci serverů i desktopů. VMware je společnost, která nabídla jako první hypervisor přímo integrovaný do hardwaru serveru se systémem ESXi a jako první z něj udělala bezplatný doplněk k hostitelskému serveru. K ovládání tohoto hypervisoru se používá program s názvem vSphere Client, viz obr. 1.4 [1].
- Své vlastní hypervisory a virtualizační softwary rovněž nabízí společnosti Red Hat(Xen), Oracle (Oracle VM), Novell (Xen), Sun (xVM), IBM, Virtual Iron a další [1].



Obr. 1.4: Grafické uživatelské prostředí vSphere Client.

1.4 Virtualizace na koncových stanicích

Stejná technologie, která pohání serverovou virtualizaci, může pohánět i virtualizaci desktopů. Tato virtualizace centralizuje nasazení desktopů, díky čemuž nad nimi můžeme získat plnou kontrolu. Uživatelé pro přístup k vnitřní síti používají připojení ke vzdálené ploše RDC[4] a mohou na tomto stroji pracovat na tenkém výpočetním zařízení, nespravovaném osobním počítači, domácím osobním počítači nebo veřejném osobním počítači. U virtualizace desktopů každý uživatel získá přístup ke svému vlastnímu desktopu a tudíž i své vlastní sadě aplikací. Což omezí možný dopad na aplikace, které jsou potřeba u jiných desktopových relacích. Pokud v jednom desktopu dojde k něčemu neobvyklému, tak to neovlivní žádné další virtuální desktopy běžící na stejném serveru.

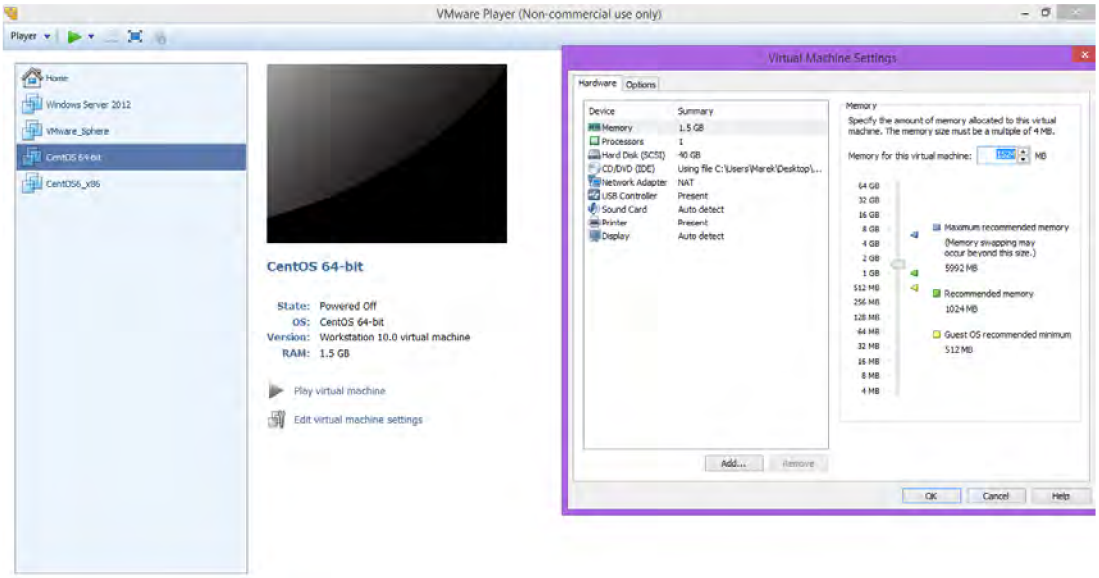
Při přechodu na virtualizaci desktopů jsou potřeba, od fyzických pracovních stanic, pouze tři věci [1]:

- Základní operační systém, může jím být jakýkoliv systém od Windows XP, nebo nějaká alternativa od Linuxu. Tento systém je potřeba aktualizovat a záplatovat, což musíte dělat tak či tak.
- Dobrá antivirová ochrana, kterou musíte tak či jinak spravovat, pokud máte fyzické desktopy.
- Klienta pro připojení ke vzdálené ploše.

Je zde několik různých typů enginů virtualizace desktopů a obvykle spadají do dvou různých modelů: místní a centralizovaný. Model místní virtualizace desktopů

[4]Remote Desktop Connection

se zaměřuje na používání vlastních zdrojů koncového bodu pro spouštění virtuálního desktopu. Na fyzickém desktopu je nainstalován virtualizační software a bitová kopie systému je přímo poskytována koncovému uživateli, který pak spouští virtuální desktop nad svým fyzickým desktopem.



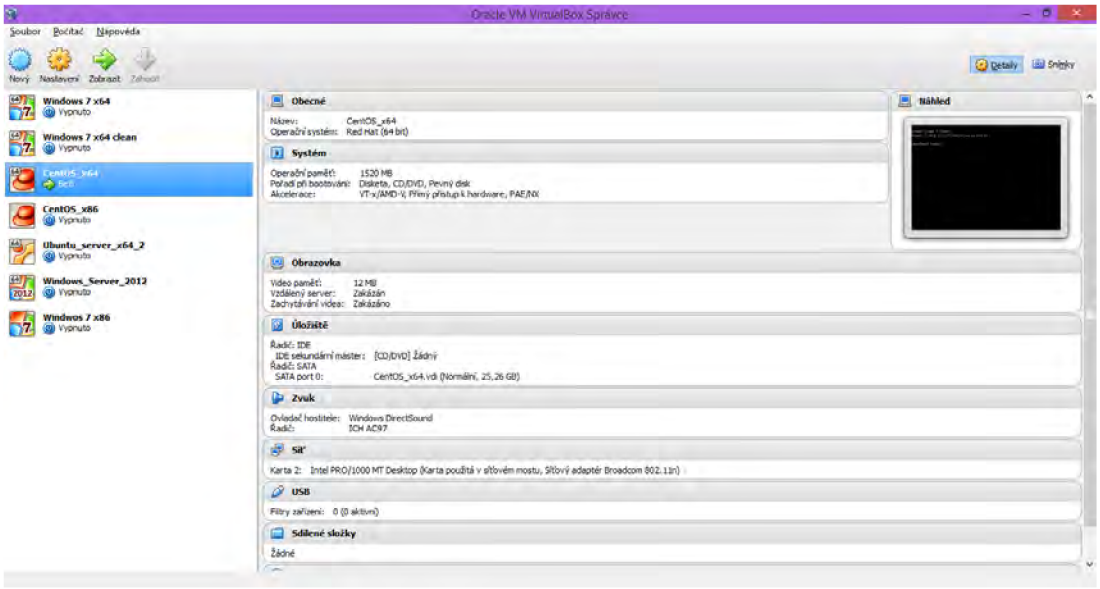
Obr. 1.5: Grafické uživatelské prostředí VMware Player.

Druhý centralizovaný model bývá označený jako virtualizace desktopů hostovaná na serveru. Zde virtuální desktopy běží nad provozním hypervisorem a uživatel k nim přistupuje vzdáleně.

Stejně jako u serverové virtualizace, tak i u desktopové je zde řada produktů, ale mezi ty ověřené a kvalitní řešení patří:

- Společnost VMware nabízí bezplatný produkt VMware Player, viz obr. 1.5, což je engine pro provoz virtuálních strojů na desktopech. Další produkt VMware Workstation nabízí nejpokročilejší sadu funkcí pro virtuální počítače od této společnosti. Pro platformu MAC je zde i další řešení VMware Fusion. Pro centrálně řízenou správu a provoz služeb je určený Virtual Desktop Infrastructure (VDI), nebo VMware Ace, který nabízí bezpečné, uzamčené a zašifrované virtuální počítače [1].
- Společnost Cirtix nabízí produkt XenDesktop, který stejně jako VDI od VMware dokáže centrálně řídit správu desktopů [1].
- Společnost Microsoft nabízí bezplatný produkt Virtual PC pro provoz virtuálních počítačů na desktopech. Dále bezpečné, uzamčené a zašifrované virtuální počítače určené ke spouštění na desktopu uživatele nabízí produkt Kidaro. Nově také produkt Hyper-V jde nainstalovat na nový desktopový systém a vytvářet zde místní i centralizovaný model virtualizace desktopů [1].

- Další společností, která nabízí pokročilou správu je Oracle se svým produktem Virtualbox, viz obr. 1.6. Tato společnost poskytuje otevřený zdrojový kód pod licencí a její produkt podporuje veškeré verze operačních systémů, od Windows, přes Linux a Macintosh až k Solaris. Produkt není primárně určen pouze do firemního prostředí, ale i pro domácí použití [4].



Obr. 1.6: Grafické uživatelské prostředí Virtualboxu.

1.5 Virtualizační metody použité při měření

Pro měření jsem zvolil tři různé metody virtualizací. Kde první bude metoda od firmy VMware a to produkt vSphere, která je nejvíce rozšířená a zastupuje model Hardwarové virtualizace. Druhou metodou bude Virtualbox od firmy Oracle, která patří do modelu softwarové virtualizace a instaluje se na již přítomný operační systém. Poslední metoda má simulovat podobné podmínky, jako používá síť Planetlab, čili postavená na rozšíření Linuxového jádra, kterou bude KVM.

1.5.1 VMware vSphere

Pro hardwarovou virtualizaci jsem zvolil jednu z nejpoužívanějších virtualizačních metod VMware vSphere a tu ve verzi 5.5.0. Jádrem tohoto produktu je hypervisor tvořící virtualizační vrstvu, která slouží jako základ zbytku produktové řady. Existuje ve dvou podobách: VMware ESX a VMware ESXi. Oba tyto produkty podporují stejné virtualizační funkce, mají shodné virtualizační nástroje a instalují se pouze na „holý“ hardware. Tyto verze se liší pouze v balení.

VMware ESX je složený ze dvou komponent, kde se oba doplňují tak, aby tvořily dynamické a robustní virtualizační prostředí: jsou to Service Console a VMkernel.

Service Console je ve všech ohledech operační systém, je odvozená z Linuxu a obsahuje služby, které jsou v klasických operačních systémech, např. firewall nebo webový server. Současně ale i postrádá mnohé funkce a vlastnosti, které má klasický operační systém. Zde byla tato komponenta „očesaná“ úmyslně, aby poskytovala pouze ty služby, které jsou nezbytně nutné pro podporu virtualizace. Service Console zajišťuje přístup k druhé komponentě VMkernel [5].

Druhá komponenta je VMkernel, která je skutečným základem virtualizačního procesu. Zajišťuje přístup virtuálních počítačů k základnímu fyzickému hardwaru prostřednictvím plánování CPU, správy paměti a virtuálním přepínáním zpracování dat.

VMware ESXi je hypervisor nové generace. Tento hypervisor se instaluje a provozuje bez komponenty Service Console, díky tomu si vystačí pouze s 32 MB. Podporuje stejné virtualizační funkce jako ESX a sdílí stejný základní VMkernel [5].

Pro správu hostitelů ESX/ESXi slouží aplikace pro systém Windows VMware vSphere Client. Aplikace se instaluje tak, že navštívíme URL hostitele ESXi a vybereme příslušný odkaz na instalaci. Tento produkt je grafickým uživatelským rozhraním, které se používá ke správním úkonům a pro pokročilé konfigurace virtuální infrastruktury. Pro připojení klienta k hostiteli potřebujeme uživatelský účet na tomto hostiteli a adresu hostitele [5].

1.5.2 Virtualbox

Virtualbox je virtualizační aplikace, která jde distribuovat na několik operačních systémů. Může být nainstalovaná na počítače postavené na architektuře od Intelu nebo AMD a na systémech Windows, Mac, Linux nebo Solaris. V této aplikace lze spustit tolik virtuálních strojů, kolik potřebujeme, jediné limity jsou velikost disku a operační paměť [4].

Virtualbox potřebuje pro spuštění existující operační systém, takže patří do softwarové virtualizace. Na všechny podporované operační systémy je použita stejná platforma a stejné soubory Virtualboxu. Tudíž lze virtuální stroje vytvořené například v systému Windows přenést na operační systém Mac.

1.5.3 KVM

KVM je modul, který se vkládá do Linuxového jádra a umožňuje přístup k virtualizaci pro hardwarové procesory od Intelu a AMD. Tato virtualizace využívá QEMU pro emulaci hardwaru. QEMU je emulátor uživatelského prostoru, v tomto prostoru

se většinou nachází spustitelné soubory daných procesů, knihovny a další soubory vytvořené operačním systémem. Tento emulátor dokáže emulovat několik hostujících procesorů na hostitelské procesory s různým výkonem. Použitím jádrového modulu KVM tak povolíme virtuálnímu stroji přiblížit se k původní rychlosti před virtualizací [6].

KVM je zpravováno pomocí libvirt API a jejich nástrojů. Jako například nástroje: virsh, virt-install a virt-clone.

2 PROTOKOL TCP

Ke změření časové odezvy mezi navázáním a ukončením SSH komunikace je potřeba rozlišit tyto události. Tomu pomůže právě znalost protokolu TCP, kde je hlavní navázání a ukončení TCP spojení.

Protokol TCP přepravuje data mezi dvěma konkrétními aplikacemi běžícími na počítačích. Základní přenosovou jednotkou tohoto protokolu je „TCP segment“. Pro přenos těchto dat v síti se používá protokol IP.

Protokol je spojovanou službou, což znamená, že mezi dvěma aplikacemi v první řadě naváže spojení a následně se vytvoří virtuální okruh. Jednotlivé bajty jsou číslovány a pokud se některý z nich ztratí, či poškodí, tak jsou data znovu vyžádána. Kontrolním součtem je zabezpečena integrita přenášených dat.

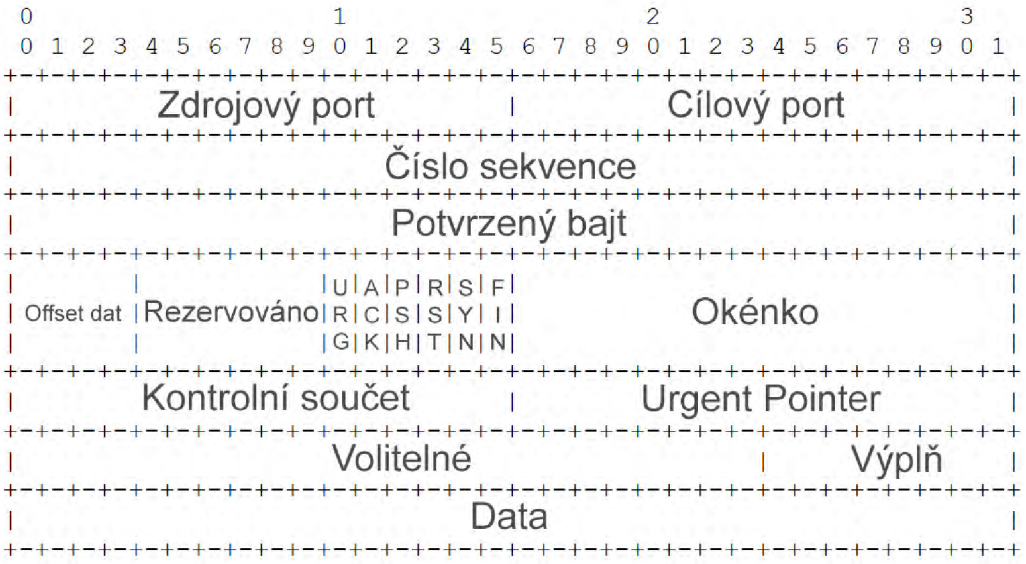
Číslem portu jsou určeny konce spojení, odesílatele i adresáta. Toto číslo má velikost dvou bajtů, tudíž může nabývat hodnot od 0 až po 65535. Za číslo portu se zpravidla napíše lomítko a název protokolu (80/tcp), čímž se určí o jaký protokol jde. V internetu je cílová aplikace adresována IP-adresou, číslem portu a použitým protokolem (TCP nebo UDP). Protokol IP dopraví datagram na konkrétní počítač, kde běží jednotlivé aplikace. Operační systém pozná podle čísla portu, jaké aplikace už má TCP segment doručit [7].

Vlastnosti TCP protokolu [7]:

- TCP protokol navazuje a ukončuje spojení, toto spojení je oboustranné. Pro navázání a ukončení spojení používá sadu příznaků v záhlaví segmentu.
- Pro oba směry spojení čísluje přenášené bajty a sleduje, jestli nebyl tok přenášených dat přerušen. Přenesená data jsou potvrzená příznakem ACK.
- Od navázání spojení začíná číslování přenášených dat od náhodně určeného počátečního čísla.
- Z přenesených dat počítá kontrolní součet, který je součástí TCP segmentu. Na základě tohoto pak příjemce zjistí, jestli nebyla data cestou poškozena. Pokud byla poškozena, či ztracena, tak si vyžádá zopakování přenosu dat.
- Dokáže řídit tok přenášených dat, aby byla přenosová cesta maximálně využita.

TCP segmenty jsou posílány jako internetové datagramy. Hlavička internetového protokolu obsahuje několik informačních polí, včetně zdrojové a cílové adresy hosta. TCP hlavička poskytuje informace specifické pro TCP protokol a následuje na ni internetová hlavička. Toto rozdělení dovoluje existenci jiných hostitelských protokolů než TCP [8].

Formát hlavičky TCP protokolu je na obr. 2.1. První pole je zdrojový port, které má velikost 16 bitů a obsahuje číslo zdrojového portu. Další je cílový port, které obsahuje číslo cílového portu a má také velikost 16 bitů. Pole číslo sekvence je veliké



Obr. 2.1: Formát hlavičky TCP protokolu [8].

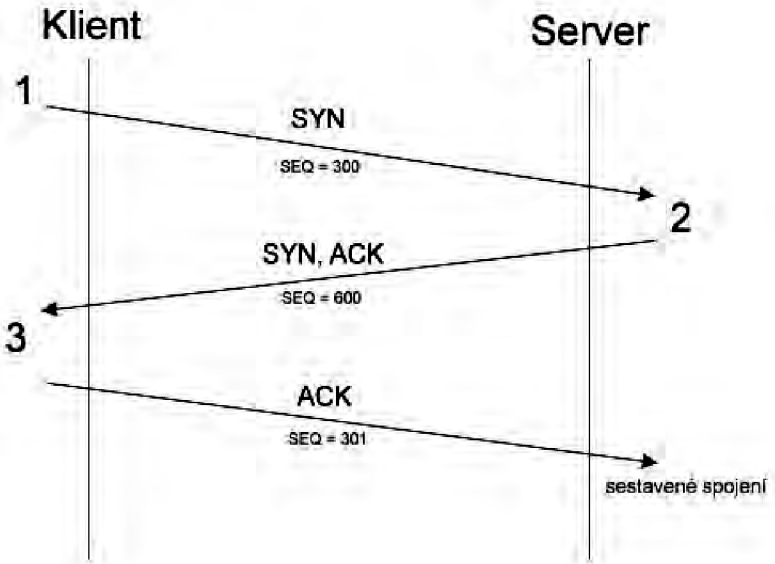
32 bitů a obsahuje číslo sekvence prvního oktetu v segmentu, s výjimkou přítomnosti příznaku SYN. Pokud je tento příznak přítomen, tak je číslo sekvence označené jako počáteční číslo sekvence (ISN[1]) a první oktet s daty je ISN+1. Pole s potvrzeným bajtem má 32 bitů a pokud je ACK nastaveno, tak toto pole obsahuje hodnotu dalšího čísla sekvence odesílatele segmentu, který má dojít. Jakmile je připojení sestaveno, tak toto pole je vždy nastaveno. Pole offset dat má 4 bity a udává, kde data začínají. Pole s rezervou má 6 bitů a je určeno do budoucna. Musí být vždy nastaveno na nulu. Pole s kontrolními bity, neboli příznaky, má 6 bitů a obsahuje příznaky URG, ACK, PSH, RST, SYN a FIN. Příznak RST znamená resetování komunikace, SYN slouží k synchronizaci čísel sekvence a FIN k ukončení spojení. Pole okénko udává počet oktetů začínajících jedničkou indikovanou v potvrzujícím poli, které posílající segment je ochotný přijmout. Toto pole má velikost 16 bitů [8].

Pole kontrolního součtu má 16 bitů a pokud segment obsahuje lichý počet záhlaví a textový oktet je zkontrolován, tak poslední oktet je ořezán vpravo s nulami, aby se vytvořilo 16-ti bitové slovo pro účely kontrolního součtu.

Pro sestavení TCP spojení se používá proces nazvaný Three-Way Handshake (nebo také TCP handshake). Tento tří cestný proces se používá i pro ukončení spojení, ovšem místo příznaků SYN se používají příznaky FIN.

Jak lze vidět na obr. 2.2, tak v první řadě začíná komunikaci klient. Pošle protější straně inicializační paket SYN s vygenerovaným číslem sekvence 300 a nastaví si časovač pro opětovné poslání. Pokud odpověď nedorazí do definované doby, odešle klient inicializační paket znovu.

[1]Initial Sequence number



Obr. 2.2: Sestavení TCP spojení.

Po přijetí inicializačního paketu server ověří nastavení příznaku SYN a kontrolní součet. Pokud je vše v pořádku, tak klientovi odešle paket s příznaky SYN a ACK. Server v paketu odešle vlastní vygenerované číslo sekvence, např. 600, a zároveň upozorňuje protější stranu, že další paket se očekává s číslem sekvence 301. Stejně jako klient, tak i server si nastaví vlastní časovač pro opětovné poslání [7].

Klient obdrží od serveru paket potvrzení synchronizace SYN a ACK. Po tomto přijetí je serveru odeslaný paket s příznakem ACK a s dohodnutým číslem sekvence 301. Dále klient informuje server, že další paket od něj očekává pod číslem sekvence 601. Po přijetí posledního třetího paketu serverem je TCP spojení sestaveno a může být zahájena komunikace mezi oběma zařízeními [9].

3 PROTOKOL ICMP

U jednoho z měření měřím latenci z portu 22. Což je port, kde naslouchá SSH server. Pro toto měření využívám aplikaci ping, která pro svou funkci využívá právě protokol ICMP.

Příležitostně cílová zařízení komunikují s zdrojovými hosty, například kvůli reportování chyb při zpracování datagramů. Pro tyto účely je určen protokol ICMP[1]. Tento protokol využívá základní podpory protokolu IP, jako kdyby to byl protokol vyšší úrovně, ale ve skutečnosti je nedílnou součástí IP protokolu a musí být implementován každým IP modulem [10].

ICMP zprávy jsou posílány v několika situacích. Například, když datagram nemůže dosáhnout svého cíle, když má brána plnou kapacitu zásobníku k přeposlání datagramu a když brána může přesměrovat hosta k posílání dat kratší cestou.

Internetový protokol není navrhnut, aby byl absolutně spolehlivý. Účel těchto kontrolních zpráv je zprostředkování zpětné vazby o problémech v komunikačním prostředí. Není zde garantováno, že datagram bude doručen nebo kontrolní zprávy budou vráceny. Některé datagramy stále mohou být nedoručeny bez jakýchkoliv reportů o jejich ztrátě. Pokud je potřeba, tak na vyšší úrovni protokolů, které využívají IP, mohou být implementovány jejich vlastní postupy ke spolehlivosti protokolu [10].

ICMP zprávy typicky reportují chyby ve zpracování datagramů. Aby se zabránilo nekonečnému zacyklování zpráv o zprávách jako, nejsou posílány ICMP zprávy o ICMP zprávách.

ICMP zprávy jsou posílány se základní IP hlavičkou. První oktet obsahuje pole ICMP typ. Toto pole určuje formát zbývajících dat. Jakékoliv pole označené jako „nepoužité“ je rezervováno pro pozdější použití a musí být označeno nulou při posílání, ale přijímací strana nepoužívá toto pole. Pokud není uvedeno jinak, tak podle popisu jednotlivých formátů jsou hodnoty polí internetových hlaviček popsány způsobem, že verze je 4, typ servisu je 0. Dále se zde nachází pole určující délku internetové hlavičky ve 32-bitovém slově, délka internetové hlavičky a dat v oktetech. Pole používané ve fragmentaci identifikace, značky a fragmentovací ofset. Pole „Time to Live“ je snižováno každým strojem, ve kterém je zpracováváno. Hodnota v tomto poli může být velká alespoň, jako počet bran, přes které datagram prochází. Pole protokol jenom zobrazuje ICMP = 1. Dále je zde pole kontrolní součet hlavičky, které je z 16-bitových jedniček sloučených z jedničkového součtu všech 16-bitových slov v hlavičce. Pro počítání kontrolního součtu může být kontrolní součet roven nule. Tento součet může být v budoucnu nahrazen. Poslední dvě pole jsou zdrojová a cílová adresa, kde zdrojová určuje adresu brány nebo hosta, který napsal ICMP zprávu a cílová určuje bránu nebo hosta, kterému zpráva byla poslána [10].

[1] Internet Control Message Protocol

Obr. 3.1: ICMP zpráva Echo či Echo Reply [10].

Příklad zprávy Echo či Echo Reply je na obr. 3.1. Adresa zdroje v této zprávě bude cílová z Echo odpovědi. V poli typ je hodnota buď nula pro zprávu typu „echo reply“ a osm pro zprávu typu „echo“. Kód je zde nastaven na nulu. Pokud je kód roven nule, tak identifikátor zde pomůže v porovnání zprávy, pokud jde o zprávu „echo“ či její odpověď, a může být roven nule. Číslo sekvence zde slouží ke stejnému účelu jako identifikátor a může být také nastaven na nulu. Například identifikátor může být použit jako port v TCP či UDP k identifikaci spojení a číslo sekvence může být zvýšeno za každou poslanou žádost echo. Odpovídající vrátí tyto hodnoty stejné ve zprávě echo reply [10].

Data doručená ve zprávě „echo“ musí mít odpověď od zprávy „echo reply“.

4 SLUŽBA SSH

Pro vzdálené připojení na operační systém se využívá služba SSH. Hlavním úkolem této služby je zaručení bezpečné komunikace po síti. Na nezabezpečeném kanále poskytuje silnou autentizaci a bezpečnou komunikaci. Je náhradou protokolu Telnet, který je nezabezpečený. Přes zabezpečený kanál SSH mohou být přenášena i spojení X11 (X Windows System), či libovolný protokol, kde je TCP server běžící na jednom pevném portu.

Původní kdo zveřejnil tento protokol, byla skupina finských programátorů, kteří ho dali k dispozici s poměrně volnou licencí, ale i tak nešlo používat protokol pro výdělečné účely zcela volně. Druhá verze protokolu byla uvedena na trh s přísnější licencí a s novým vylepšeným protokolem. Nakonec se protokolu ujalo IETF, které protokol v roce 2006 standardizovalo. Objevili se i další implementace protokolu a některé ve formě Open Source software, např. OpenSSH [7].

Protokol obsahuje 3 hlavní komponenty [11]:

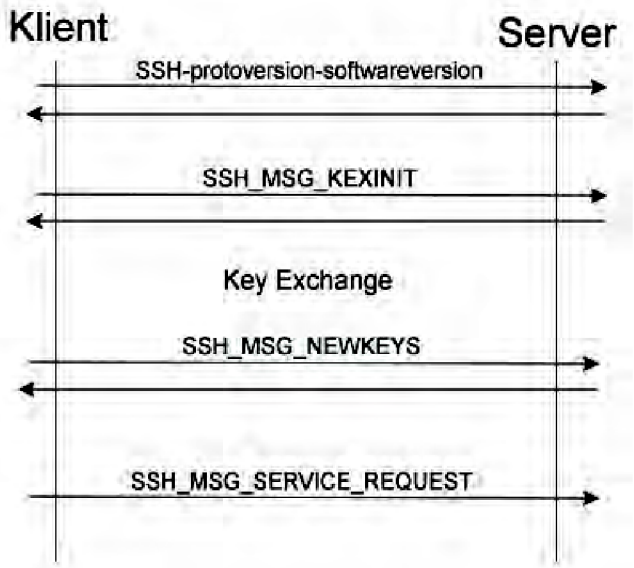
- **The Transport Layer Protocol** – zprostředkovává serverovou autentizaci, důvěrnost a integritu. Může i zprostředkovávat kompresi. Tato vrstva bývá typicky spuštěna pod TCP/IP připojením, nebo také může být použita na vrcholu některého spolehlivého datového toku.
- **The User Authentication Protocol** – autentizuje klientskou stranu k serveru. Tento protokol se pouští nad transportním protokol.
- **The Connection Protocol** – spojuje zabezpečené tunely do několika logických kanálů. Tento protokol se pouští nad předchozím protokolem.

Klient pošle servisní žádost, jakmile je zabezpečené připojení na transportní vrstvě sestaveno. Druhá servisní žádost je odeslaná, jakmile je autentizace uživatele kompletní. Toto umožní novým protokolům společně existovat s protokoly uvedenými výše.

Připojovací protokol zprostředkovává kanály, které mohou být použity pro velké množství účelů. Standardní metody zprostředkovávají terminálové služby a pro posílání libovolné TCP/IP porty a X11 spojení [11].

Rozdíly mezi verzí protokolu SSH1 a SSH2 [7]:

- Velká část protokolu SSH2 byla přebrána z protokolu SSH1.
- Jestliže se klient pokusí na server přihlásit s novější verzí protokolu, tak se musí ihned odhlásit a znovu se přihlásit se starší verzí protokolu.
- Pokud chce být server s verzí SSH2 kompatibilní se staršími verzemi, tak se ohlásí jako „SSH 1.99“. Klient s verzí SSH2 to pochopí a ví, že může používat novější verzi protokolu. Pokud klienti používají starší verzi protokolu, tak se jim server přizpůsobí.



Obr. 4.1: Inicializace spojení protokolu SSH [12].

- Podporuje nové metody výměny klíče s dvojitým šifrováním. Podporuje metodu Diffie-Hellman.
- Kromě RSA přibyla i podpora dalších algoritmů pro veřejné klíče DSA.
- Protokol je bezpečnější.
- Je zde zabudovaná podpora SOCKS.
- Bezpečný protokol pro přenos souborů (sftp).

Při navázání spojení SSH protokolu se v první řadě vymění informace o verzi používaného protokolu. Dále se začne s inicializací klíčů a jejich následnou výměnou. Po této výměně již může začít zabezpečená komunikace pomocí protokolu SSH [12].

Protokol OpenSSH je zdarma šířitelná verze SSH komunikačního nástroje. Uživatelé služby telnet, rlogin a ftp nemusí realizovat jejich hesla přes nezabezpečené internetové spojení. OpenSSH zabezpečí veškeré spojení (včetně hesel) k efektivní eliminaci odposlechu komunikace a dalším útokům. Dále OpenSSH zprostředkovává zabezpečené tunelové spojení, několik autentizačních metod a podporu všech verzí SSH protokolu [13].

Sada OpenSSH nahrazuje rlogin a telnet s SSH programem, rcp s scp a ftp s sftp. Dále je zde implementováno sshd a další nástroje jako ssh-add, ssh-agent, ssh-keysign, ssh-keyscan, ssh-keygen a sftp-server [13].

Protokol OpenSSH vyvíjí projekt OpenBSD. Software je vyvíjen v zemích, které

umožňují rozšíření kryptografie a je volně použitelný pro všechny pod licencí BSD [1]. Vývoj tohoto protokolu mají na starosti dva týmy. První z nich dělá striktně vývoj založený na OpenBSD, který cílí na produkci kódu, který je jednoduchý, čistý a bezpečný, jak může být. Tato jednoduchost vede k lepší kvalitě kódu a jednoduché kontrole kódu. Druhý tým vytváří čisté a přenosné verze, aby mohli být spuštěny na mnoha operačních systémech [13].

Protokol OpenSSH se nejčastěji používá na operačním systému Linux. K puštění OpenSSH serveru je zapotřebí nejdříve na tomto systému nainstalovat balíček openssh-server. Konfigurační soubor tohoto serveru u systému CentOS se nachází v souboru `/etc/ssh/sshd_config` [14]. Zde lze nastavit veškeré parametry SSH serveru. Při každé změně v tomto souboru se musí posléze OpenSSH služba restartovat příkazem `/sbin/service␣sshd␣restart`. Takto jde službu i zastavit změnou v příkazu za stop a zapnout změnou za start.

[1] Berkeley Software Distribution

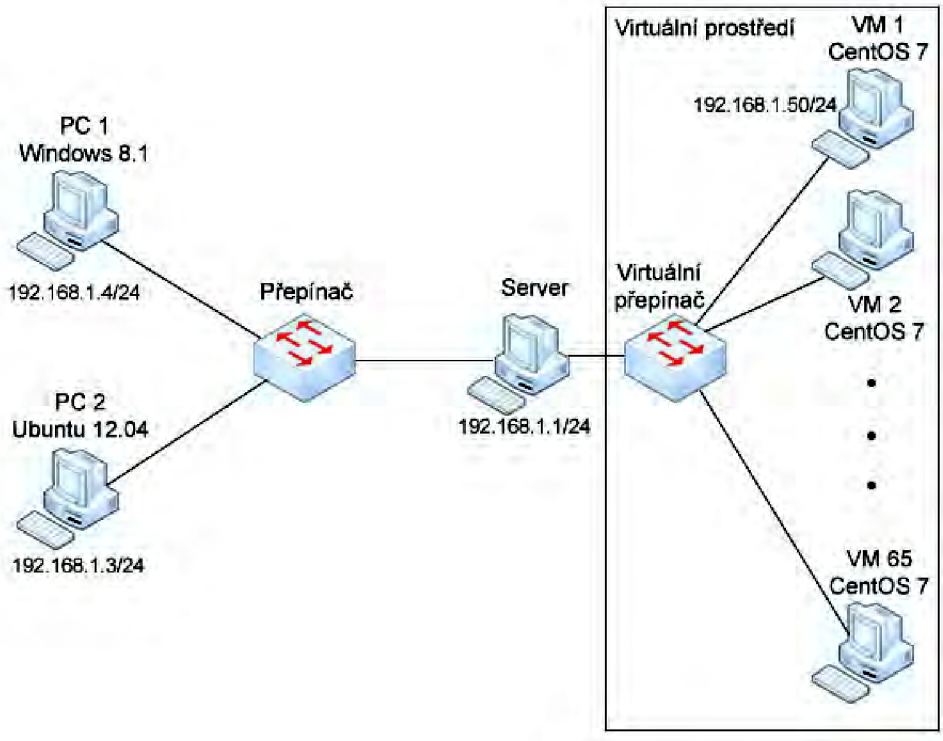
5 VÝSLEDKY PRÁCE

Pro vlastní měření jsem použil tři metody virtualizace. První je hardwarová virtualizace, pro kterou jsem zvolil produkt VMware vSphere, který je v této oblasti nejvíce rozšířený. Druhou je softwarová virtualizace, kde jsem zvolil produkt od společnosti Oracle, kterým je Virtualbox. Poslední jsem, na doporučení vedoucího práce, zvolil metodu, která je podobná metodě používané v celosvětové výzkumné síti PlanetLab. Tou je metoda, která se přímo implementuje do jádra Linuxového systému, KVM[1].

Jelikož hlavním popudem pro sepsání této práce bylo zpoždění vznikající při vzdáleném připojení v síti PlanetLab, tak se tato práce zaměřuje hlavně na virtualizační metodu KVM.

5.1 Realizované pracoviště

Jak lze vidět na obr. 5.1, tak jsem pro měření použil několik koncových bodů. Jsou



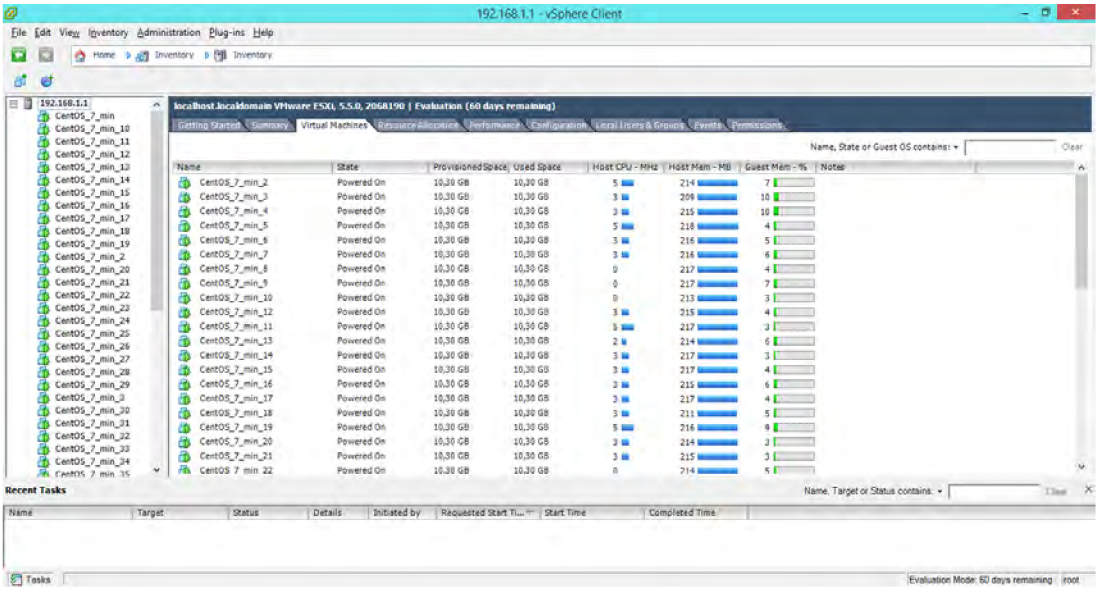
Obr. 5.1: Schéma realizovaného pracoviště.

zde dva klienti a jeden server. Veškeré koncové body jsem měl spojené přes linku s maximální rychlostí 1 Gb/s. Ústředním bodem byl přepínač Tenda G1008D s osmi gigabitovými porty. Jako klienty jsem použil dva notebooky, kde v jednom jsem

[1]Kernel-based Virtual Machine

pracoval v klasickém operačním systému Windows 8.1 x64 a druhý jsem měl vždy jiný, ale pracoval jsem zde z bootovatelného USB flash disku, kde jsem měl vytvořený a nastavený systém Ubuntu 12.04 LTS. Na druhé straně byl desktopový počítač s procesorem Intel Core i5-4440, operační pamětí 16 GiB, integrovanou grafickou kartou Intel HD Graphics 4600 a síťovou kartou, kterou podporují všechny použité virtualizační systémy.

Desktopový počítač, kde byla vždy implementovaná virtualizační metoda, obsahoval v systému i virtuální přepínač, kam byly připojeny veškeré virtuální počítače (VM[2]). Všechny virtuální počítače přistupovaly ke stejnému virtuálnímu úložišti v desktopovém počítači, které obsahovalo dva disky o velikosti 500 GB. Každý virtuální počítač, viz obr. 5.2, měl stejný 64-bitový operační systém, kterým byl sys-



Obr. 5.2: Vložené virtuální počítače ve VMware vSphere.

tém CentOS 7 ve verzi Minimal, která nemá grafické uživatelské rozhraní a obsahuje pouze základní služby pro běh systému. Virtuální počítač měl přidělené pouze jedno jádro procesoru a 200 MB operační paměti RAM.

5.2 Popis způsobu měření

Měření jsem měl rozdělené na dvě části, kde v první jsem měřil odezvu z portu 22, což je port, kde naslouchá SSH server, na počtu virtuálních stanic. Druhé měření bylo použito pro měření závislosti počtu virtuálních stanic na rychlosti komunikace SSH protokolu.

[2]VM – virtual machine

Pro první měření jsem zvolil jako výchozí bod klienta s operačním systémem Windows 8.1, kde jsem si implementoval aplikaci PsPing v2.01 od Microsoftu. Tato aplikace dokáže jednoduše změřit latenci spojení. Použil jsem následující příkaz:

Kód 5.1: Příkaz na spuštění aplikace PsPing.

```
psping -4 -n 30 -h 192.168.1.50:22
```

kde parametr `-4` si vynucuje používání IPv4, `-n␣30` znamená počet odeslaných požadavků, čili 30, dále parametr `-h` vytiskne přehledný histogram do konzole a nakonec je zde zadaná IP adresa i s názvem portu prvního virtuálního stroje, kde naslouchal SSH server [15].

Druhý způsob měření byl značně komplikovanější, jelikož zde šlo o změření času od zahájení SSH komunikace po její skončení. Zde jsem si tedy vytvořil několik automatizovaných skriptů, které mi ušetřili dosti času. Rozdíl času od navázání po skončení komunikace jsem nakonec zvolil metodou, kdy jsem zachytával čas odeslání příznaku SYN u TCP handshaku a příchodu příznaku FIN při ukončování spojení. Jelikož jsem používal skripty, které jdou lépe implementovat do systému s Linuxem, tak jsem pro toto měření zvolil druhého klienta, kdy jsem měl systém Ubuntu 12.04 LTS.

První skript slouží pro připojení ke vzdálenému SSH serveru a následném odhlášení od tohoto serveru.

Kód 5.2: Skript pro vzdálené připojení k SSH serveru a následné ukončení spojení.

```
#!/usr/bin/expect -f
set timeout -1
spawn ssh root@192.168.1.50
expect "s password:"
send "12345678\r"
expect "#"
send "ifconfig\r"
expect "collision"
send "exit\r"
interact
```

Zde se v první řadě vynuluje časovač, který by nám zrušil měření při delším neodpovídání serveru. Dále se připojíme k ssh serveru přes uživatele *root* na IP adresu prvního virtuálního počítače. Po tomto příkazu se čeká až se do konzole zapíše „s password:“, kde se napíše heslo 12345678 a odešle se. Jelikož dopředu víme, že je heslo správné a připojení se zdaří, tak se zde znovu čeká na znak, který se objeví po připojení klienta k serveru. Až je nalezen znak, tak se znovu napíše do konzole příkaz

`ifconfig`, který zobrazuje seznam jednotlivých síťových karet a jejich podrobnosti. Tento příkaz zde je jen pro prodloužení prodlevy a zdůraznění časové zpoždění, pokud nastane. Po zapsání příkazu `ifconfig` znovu očekáváme určitý text, kde se nachází slovo `collision`. Po tomto se jen napíše příkaz `exit`, který celé spojení ukončí.

Pro zkvalitnění měření se první skript pouští tři krát, což je znovu zautomatizované další skriptem, který zároveň pouští ten předchozí.

Kód 5.3: Skript pro spuštění stejného skriptu třikrát.

```
#!/bin/bash
expect /home/ubuntu/Desktop/login/login_ssh.sh
expect /home/ubuntu/Desktop/login/login_ssh.sh
expect /home/ubuntu/Desktop/login/login_ssh.sh
```

Tento skript tedy jen pouští původní, který spustí tři krát po sobě, takže se k SSH serveru připojíme tři krát.

Další skripty už slouží k zaznamenání komunikace, kde první z nich:

Kód 5.4: Skript pro zachytávání příznaku SYN.

```
#!/bin/bash
sudo tcpdump >mereni_syn.txt -i eth0 "tcp[tcpflags] &
(tcp-syn) != 0" -vv and src 192.168.1.3
```

kde použijeme program tcpdump, který veškeré co zachytne zaznamená do souboru `mereni_syn.txt`. Tento program analyzuje vše na rozhraní `eth0`, kde kontroluje TCP příznaky SYN, nerovnající se nule, a vyfiltruje pakety se zdrojovou IP adresou klienta PC 2.

Druhý skript pro zachytávání komunikace:

Kód 5.5: Skript pro zachytávání příznaku FIN.

```
#!/bin/bash
sudo tcpdump >mereni_fin.txt -i eth0 "tcp[tcpflags] &
(tcp-fin) != 0" -vv and src 192.168.1.50
```

zde skript zachytává komunikaci do dalšího souboru a zachytává jen vyfiltrovanou komunikaci, kde pakety obsahují TCP příznaky FIN, nerovnající se nule, a jejich zdrojová adresa je IP adresou virtuálního stroje s SSH serverem, kam se připojujeme.

Poslední skript pro zachytávání komunikace:

Kód 5.6: Skript pro zachytávání síťové komunikace.

```
#!/bin/bash
sudo tcpdump -w mereni -i eth0 -vv
```

kde skript zachytává veškerou komunikaci na rozhraní `eth0` u klienta na PC 2. Toto slouží pro případnou analýzu spojení.

Díky těmto skriptům, jsem dostal v textových souborech výsledky měření, které byly s přesností na tisícinu milisekundy. Zde je příklad textu zkopírovaného z textového souboru:

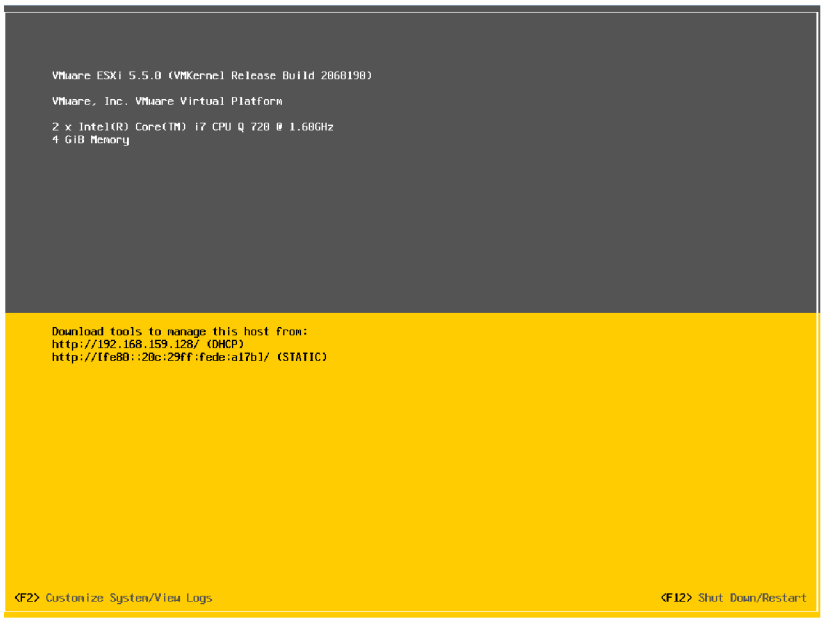
```
16:17:11.291928 IP (tos 0x0, ttl 64, id 14579, offset 0, flags [DF],
proto TCP (6), length 60) ubuntu.local.42639 > 192.168.1.50.ssh:
Flags [S], cksum 0x27b5 (correct), seq 3238133052, win 29200, options
[mss 1460,sackOK,TS val 116175 ecr 0,nop,wscale 7], length 0

16:17:31.541010 IP (tos 0x0, ttl 64, id 56740, offset 0, flags [DF],
proto TCP (6), length 60) ubuntu.local.42640 > 192.168.1.50.ssh:
Flags [S], cksum 0x07ed (correct), seq 3322807345, win 29200, options
[mss 1460,sackOK,TS val 121237 ecr 0,nop,wscale 7], length 0

16:17:51.791203 IP (tos 0x0, ttl 64, id 10387, offset 0, flags [DF],
proto TCP (6), length 60) ubuntu.local.42641 > 192.168.1.50.ssh:
Flags [S], cksum 0x75f9 (correct), seq 2191246799, win 29200, options
[mss 1460,sackOK,TS val 126300 ecr 0,nop,wscale 7], length 0
```

5.3 VMware vSphere

VMware vSphere je virtualizační produkt, jehož jádrem je hypervisor ESX/ESXi, viz obr. 5.3. Tento produkt patří do modelu hardwarové virtualizace.

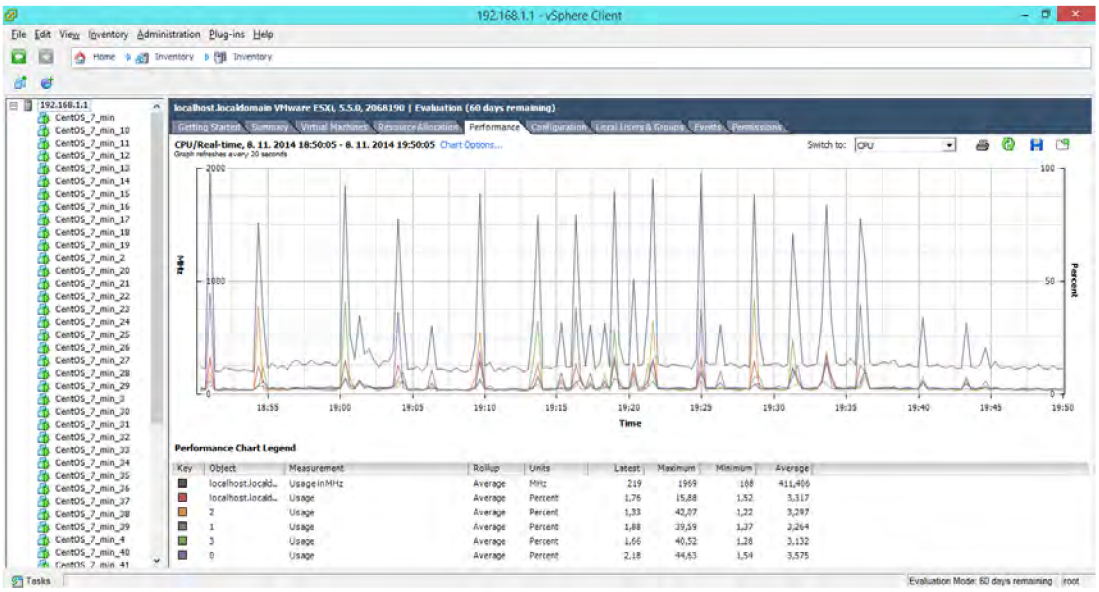


Obr. 5.3: Ukázka úvodní obrazovky VMware ESXi.

5.3.1 Vlastní měření

Před vlastním měřením jsem si propojil celou síť, nastavil veškeré potřebné adresy a ověřil konektivitu jednotlivých spojení. Po tomhle ověření jsem mohl začít s měřením. Nejdříve jsem na serveru přes klienta VMware vSphere zapnul první virtuální stroj, na který jsem se zkusil vzdáleně připojit pomocí protokolu SSH, jestli vše korektně funguje. Tudíž jsem mohl spustit jednotlivé skripty, které pracují s programem tcpdump a zachytávají komunikaci. Po tomto kroku už začalo samotné měření, kde jsem spustil skript na vzdálené připojení. Po skončení skriptu jsem mohl veškeré skripty pozavírat a zkontrolovat naměřené hodnoty. Tyto hodnoty jsem posléze vložil do předem připraveného dokumentu vytvořeného v programu Microsoft Excel.

Dále jsem se přesunul k prvnímu PC, kde jsem spustil program psping a provedl tak druhé měření. Po zpracování veškerých hodnot jsem zapnul druhý virtuální počítač a znovu jsem opakoval obě měření, takto jsem postupoval až pro 65 spuštěných virtuálních strojů, což byla velikost, která u této virtualizační metody ještě nedokázala nějak výrazně zatížit hardware serveru, viz obr. 5.4

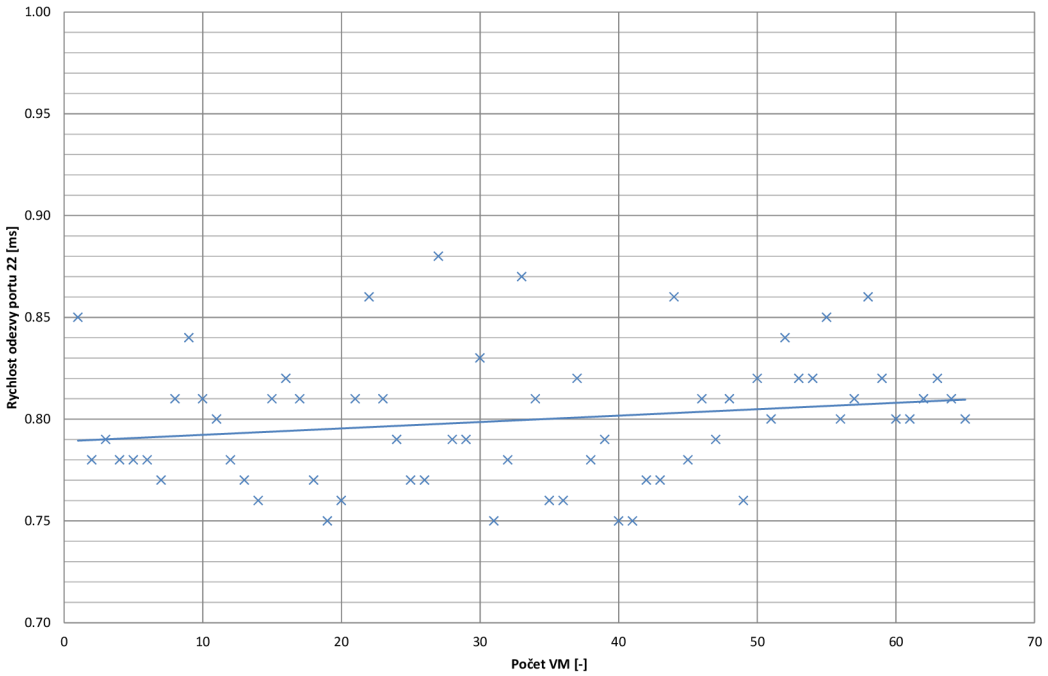


Obr. 5.4: Zatížení procesoru při spuštěných 50-ti VM.

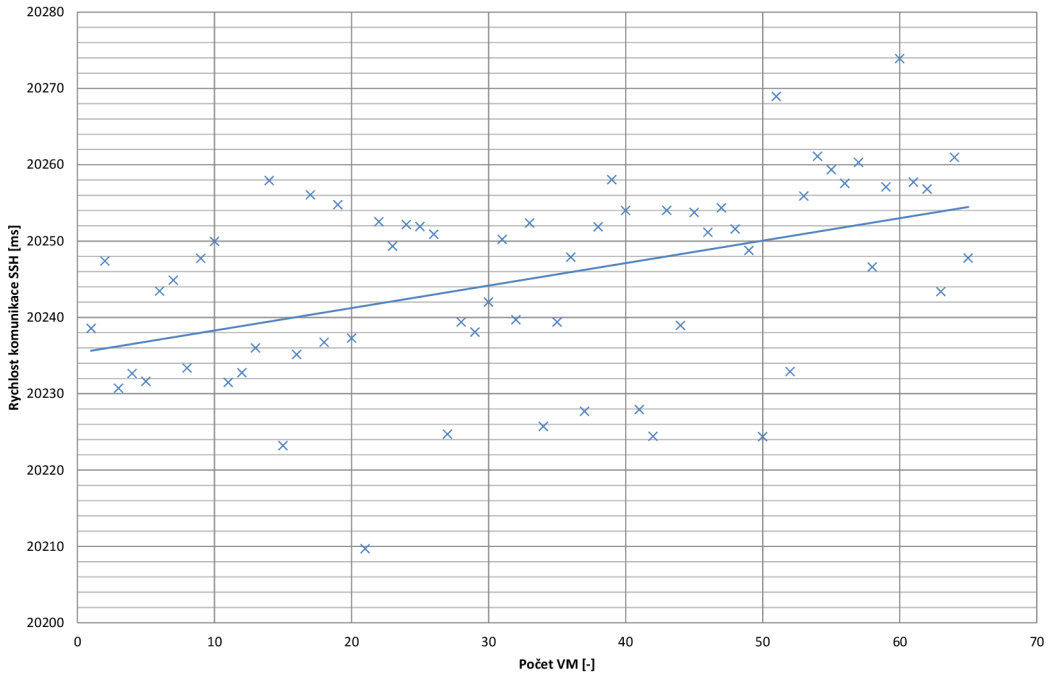
5.3.2 Výsledky měření

Po provedení veškerých měření jsem veškeré získané hodnoty vynesl do grafů. První graf, viz obr. 5.5, představuje závislost latence z portu 22 na počtu virtuálních stanic. Z grafu lze vyčíst, že hodnota latence se příliš neměnila a pohybovala se stále kolem hodnoty 0,8 milisekund. Hodnota byla vždy pod hranicí 1 milisekundy, což je podle mého názoru přijatelné a zcela zanedbatelné zpoždění.

Druhý graf, viz obr. 5.6, se již věnoval závislosti rychlosti komunikace SSH protokolu na počtu virtuálních stanic. Kde byla celá komunikace od navázání spojení, přes napsání příkazu `ifconfig`, až po ukončení spojení. Hodnota zpoždění se s rostoucím počtem virtuálních strojů lineárně zvyšuje. Ovšem maximálně o 20 milisekund až pro 65 virtuálních strojů, což je zpoždění trochu významné, ale ne až tak velké, jak bylo očekáváno.



Obr. 5.5: Graf závislosti latence z portu 22 na počtu virtuálních stanic pro VMware vSphere.



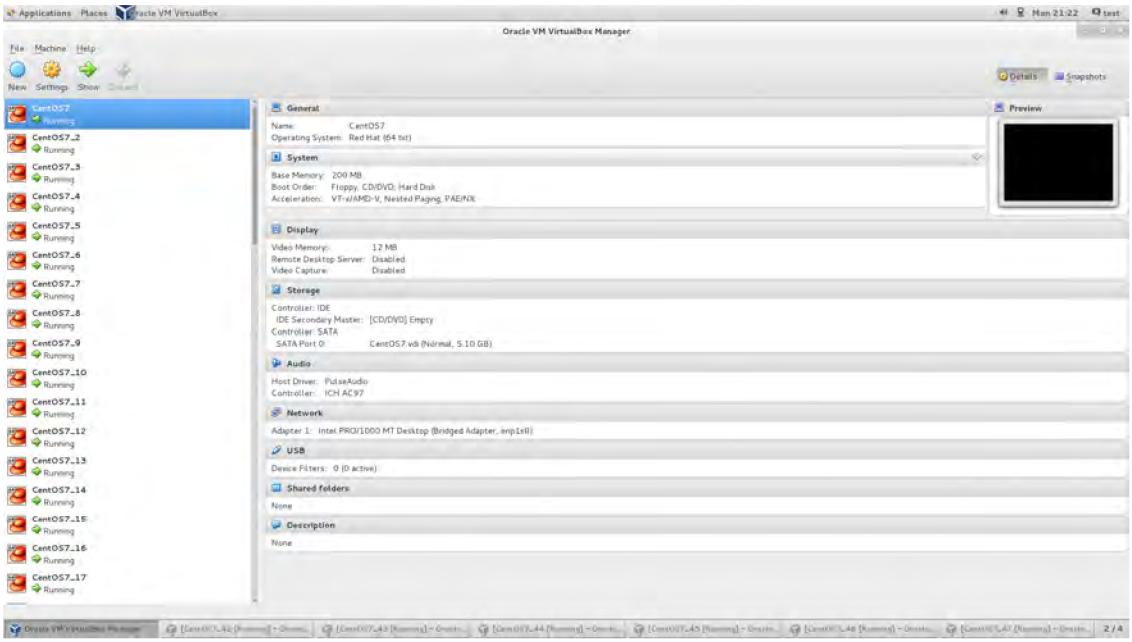
Obr. 5.6: Graf závislosti rychlosti komunikace SSH protokolu od navázání po ukončení TCP spojení na počtu virtuálních stanic pro VMware vSphere.

5.4 Virtualbox

Virtualbox patří do modelu softwarové virtualizace a podle předpokladů by měl skončit jako nejhorší virtualizační metoda.

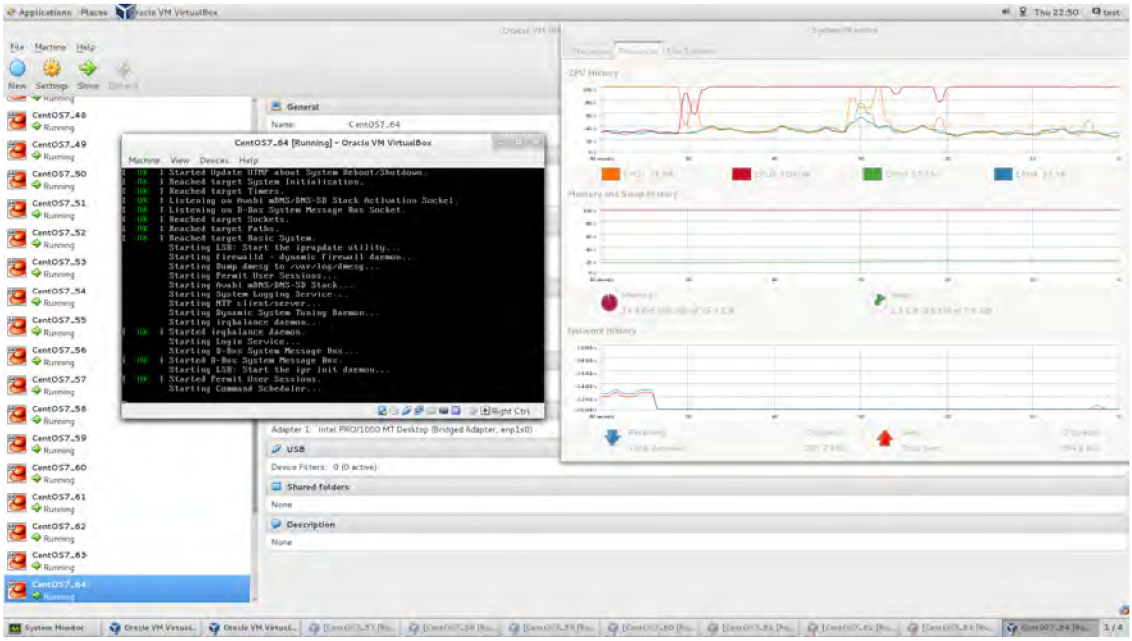
5.4.1 Vlastní měření

Na server jsem nainstaloval systém CentOS 7 s grafickým uživatelským rozhraním, dále následovaly aktualizace celého systému a v poslední řadě ověření celého systému. Na takto připravený systém byla nainstalovaná aplikace Virtualbox pomocí balíčku ve verzi 4.3. V této aplikaci se přes grafické uživatelské rozhraní, viz obr. 5.7, jednoduše vytvořily potřebné virtuální stroje, které byly pomocí Virtualboxu propojeny virtuálním přepínačem a zapojeny tak do „měřící“ sítě.



Obr. 5.7: Grafické uživatelské rozhraní aplikace Virtualbox.

Následně jsem mohl již začít s vlastním měřením, kde jsem zapnul první virtuální stroj, provedl obě měření a takto jsem opakoval pro všech 65 virtuálních strojů. Na obr. 5.8 jde vidět zatížení systému při startu 64. virtuálního stroje, kde se vymění jednotlivá jádra procesoru, ale i tak procesor zůstává nezatížen. Ovšem operační paměť již dosahuje svého maxima a aplikace Virtualbox není příliš schopná reagovat na toto zatížení.

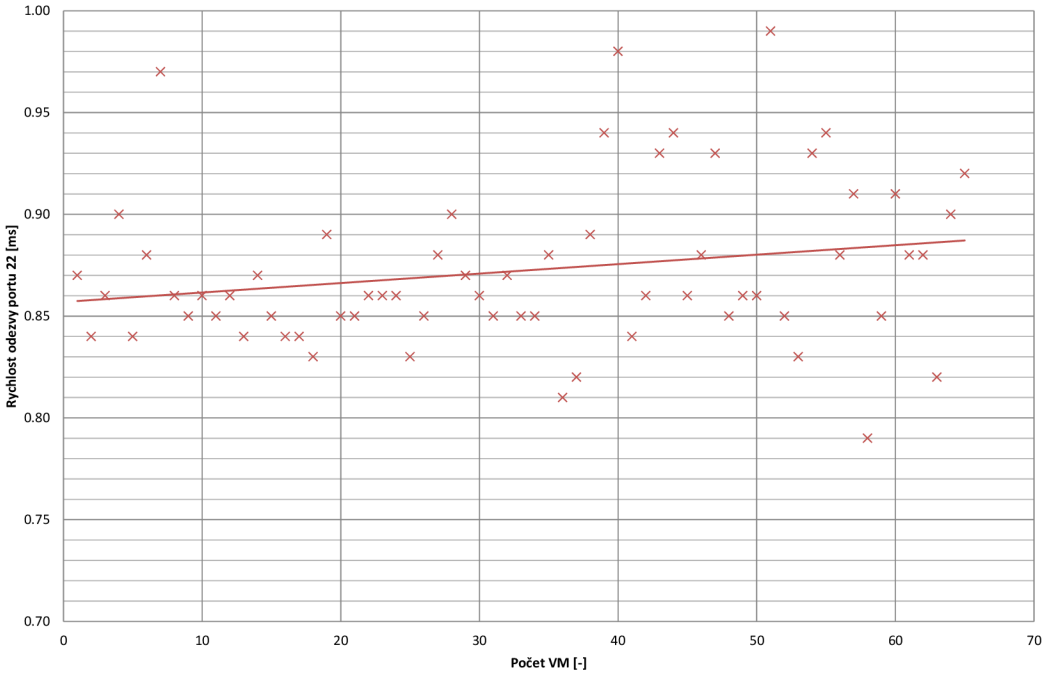


Obr. 5.8: Zatížení serveru při startu 64. virtuálního stroje u Virtualboxu.

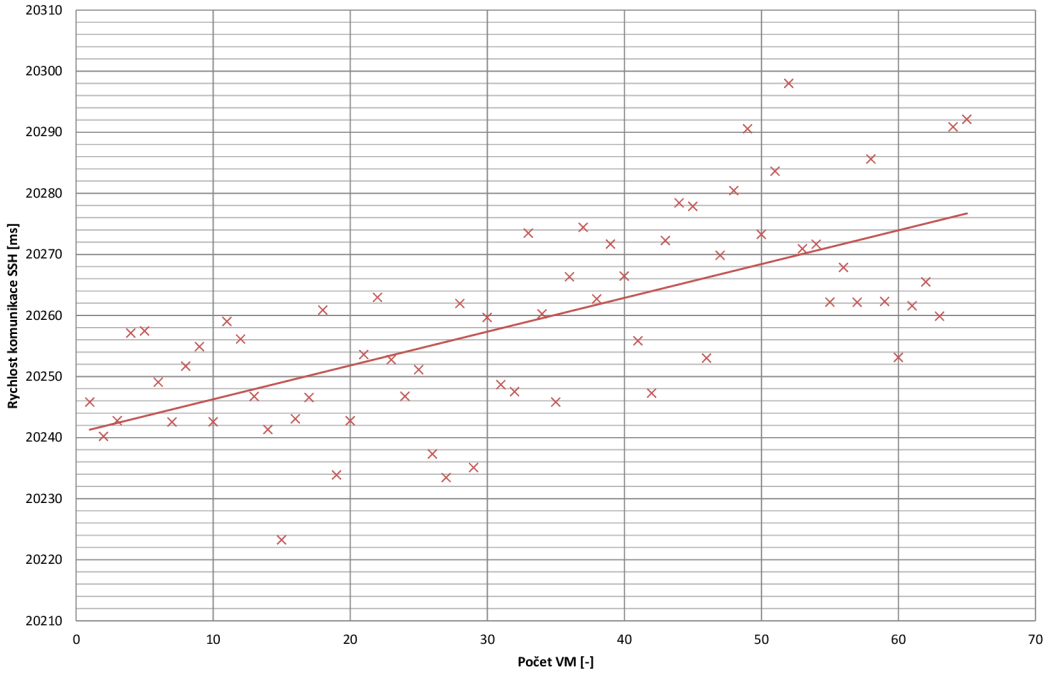
5.4.2 Výsledky měření

Po proběhnutí všech měření jsem zpracoval veškeré získané hodnoty a opět jako u metody VMware vSphere je vynesl do grafů. První graf, viz obr. 5.9, znázorňuje závislost latence z portu 22 na počtu virtuálních stanic. Z tohoto grafu lze určit, že hodnota latence se příliš neměnila, obdobně jako u měření s metodou VMware vSphere. Výsledná křivka má lineárně lehce se zvyšující tendenci a pohybuje se kolem hodnoty 0,87 milisekund. Stejně jako u předchozí metody se hodnota nedostala nad hranici 1 milisekundy.

Druhý graf, viz obr. 5.10, zobrazuje graf závislosti rychlosti komunikace SSH protokolu na počtu virtuálních stanic. Podle křivky se hodnota zpoždění lineárně zvyšuje o přibližně 40 milisekund pro 65 virtuálních strojů, což je už zpoždění dvojnásobné, oproti virtualizační metodě VMware vSphere.



Obr. 5.9: Graf závislosti latence z portu 22 na počtu virtuálních stanic pro Oracle Virtualbox.



Obr. 5.10: Graf závislosti rychlosti komunikace SSH protokolu od navázání po ukončení TCP spojení na počtu virtuálních stanic pro Oracle Virtualbox.

5.5 KVM

KVM je virtualizační metoda, která se instaluje na operační systém vložením do jádra již existujícího Linuxového operačního systému.

5.5.1 Vlastní měření

V první řadě se znovu na server nainstaluje system CentOS 7 s grafickým uživatelským rozhraním, kde se posléze provedou veškeré aktualizace a ověří jeho celková funkčnost. Na takto nachystaný systém se již mohou nainstalovat potřebné balíčky:

Kód 5.7: Instalace balíčků s KVM.

```
yum install qemu-kvm libvirt libvirt-python
yum install libguestfs-tools virt-install
```

Tento příkaz tedy nainstaluje KVM a veškeré libvirt balíčky. Poté se tyto moduly musí povolit a spustit skrze příkaz:

Kód 5.8: Povolení modulů libvirt.

```
systemctl enable libvirtd && systemctl start libvirtd
```

Jakmile je KVM plně funkční, tak je potřeba nastavit již pouze síťové nastavení. Od základu virtuální stroje mají přístup jen do vnitřní sítě vytvořené na serveru. Což lze vyřešit síťovým mostem, který virtuální stanici propojí přímo k vnější síti mimo server.

Dále jsem na tento server nainstaloval grafické uživatelské rozhraní, viz obr. 5.11, pomocí balíčků:

Kód 5.9: Instalace Grafického uživatelského rozhraní pro KVM.

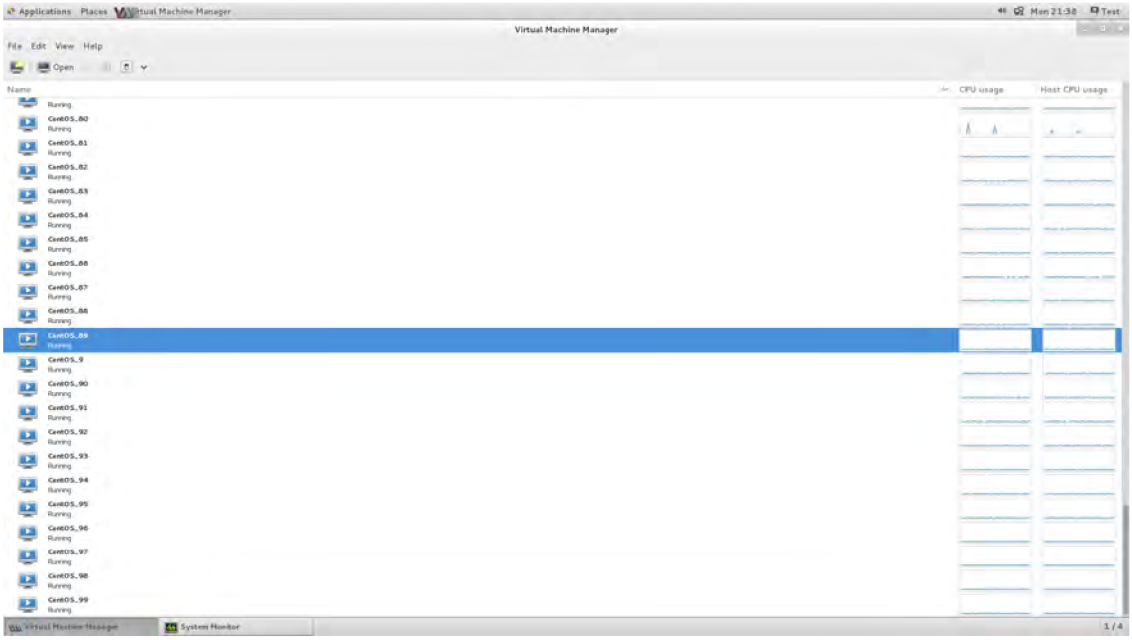
```
yum install virt-manager virt-viewer
```

které dokáží zobrazit jednotlivé virtuální stroje a spravovat je.

Vytvoření jednotlivých virtuálních strojů se provádí pomocí příkazové řádky:

Kód 5.10: Vytvoření virtuálního stroje.

```
virt-install \
  --name CentOS \
  --ram=200 \
  --vcpus=1 \
  --disk path=/home/test/images/vm1.img,size=3 \
  --cdrom /home/test/centos7minimal.iso
```



Obr. 5.11: Grafické uživatelské rozhraní programu KVM.

V první řadě se pojmenuje nový virtuální stroj, čili „CentOS“, následně se nastaví velikost operační paměti a počet procesorů přiřazených k virtuálnímu stroji. Následuje umístění a pojmenování obrazového souboru virtuálního stroje, kterému se zaznamenává i velikost v gigabajtech. `Cdrom` označuje už jen umístění instalačního souboru operačního systému. Po zadání těchto hodnot začne instalace virtuálního stroje.

Po ověření funkčnosti virtuálního stroje jsem vytvořil skript pro zjednodušení klonovacího procesu k vytvoření 65-ti virtuálních strojů.

Kód 5.11: Klonování virtuálních strojů.

```
for (i=2 ; i<=65 ; i++)
{
   virt-clone \
   --connect qemu:///system \
   --original CentOS \
   --name CentOS_\$i \
   --file /home/test/images/CentOS_\$i.img
}
```

Zde je nastavený cyklus pro naklonování virtuálních strojů až do počtu 65. Originální virtuální stroj jsem nastavil původní, dále se nastaví jméno nového virtuálního stroje a místo, kam bude stroj uložen.

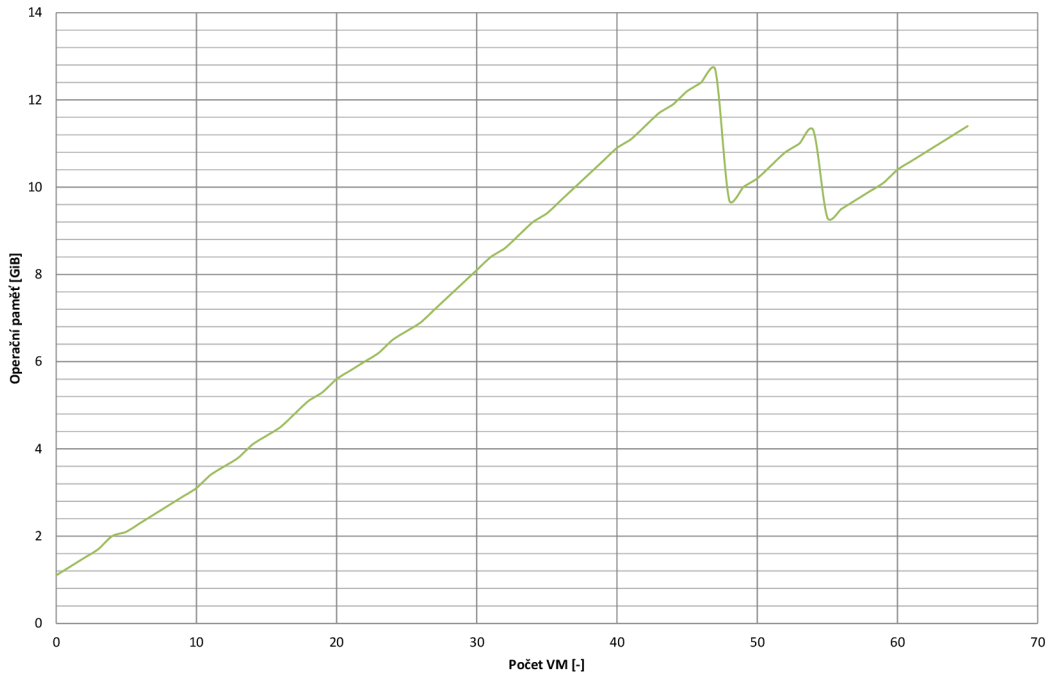
Vytvořené virtuální stroje jdou vzdáleně spustit pomocí cyklu:

Kód 5.12: Spuštění virtuálních strojů.

```
for (i=41 ; i<=60 ; i++)
{
    virsh start CentOS_\$i
    sleep 25
}
```

Tento cyklus spustí 41. až 60. virtuální stroj a vždy po startu každého virtuálního stroje vyčká 25 sekund než spustí start dalšího, což je ošetření proti přetížení hardwaru při startu více virtuálních strojů.

Virtualizační program KVM dokázal reagovat na počet spuštěných virtuálních strojů a při větším využití operační paměti ji pak ubíral jednotlivým spuštěným strojům, aby zde mohlo vzniknout místo pro nové stroje a nezatížil se příliš hardware. Na obr. 5.12 lze vidět, že až do počtu 47 virtuálních strojů byla křivka lineární, ovšem poté bylo zabráno 12,7 GiB operační paměti. Toto je hranice, u které začal virtuální nástroj již reagovat na zatížení.

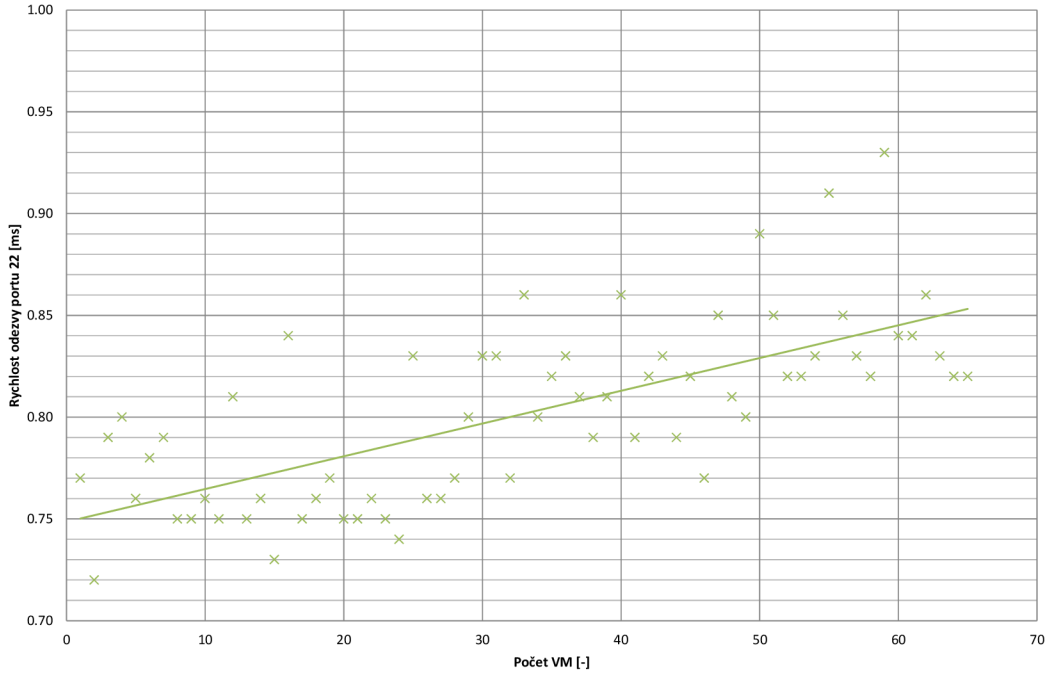


Obr. 5.12: Graf závislosti velikosti využité operační paměti na počtu virtuálních stanic pro virtualizaci pomocí KVM.

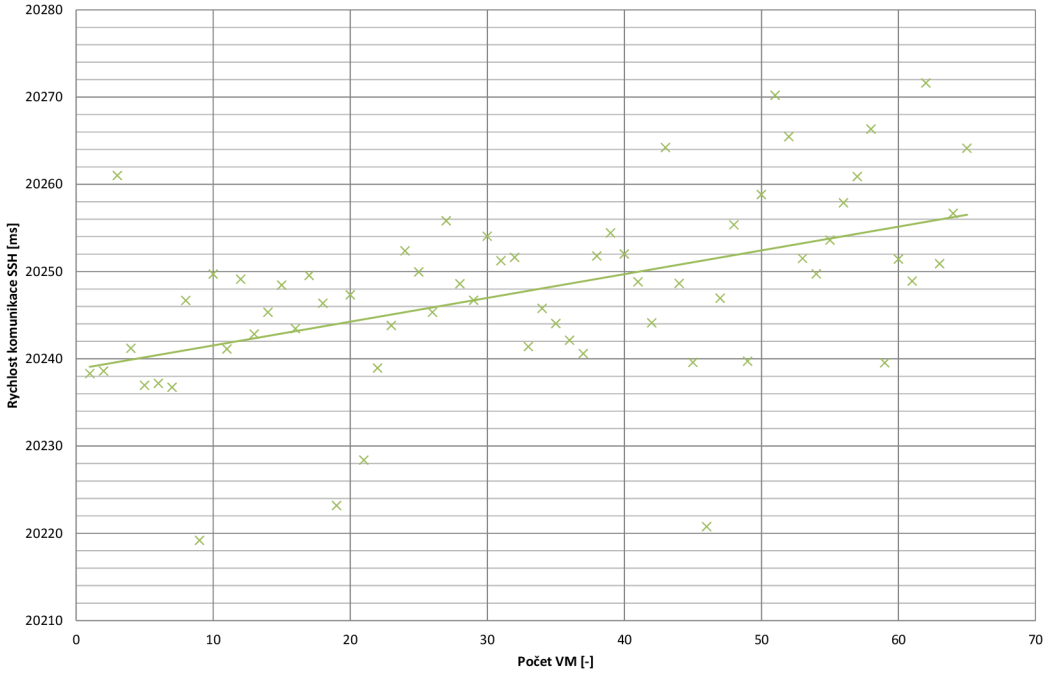
5.5.2 Výsledky měření

Veškeré změřené výsledky jsem vynesl do grafů stejně jako u předchozích měření. První graf, viz obr. 5.13, zobrazuje závislost latence z portu 22 na počtu virtuálních stanic. Podobně jako u předchozích se hodnota nezvýšila přes hranici 1 milisekundy. Křivka lineárně roste a mění svou hodnotu o 0,1 milisekund, což je zcela zanedbatelné zpoždění.

Druhý graf, viz obr. 5.14, zobrazuje graf, kde se měří časová závislost mezi příznaky SYN a FIN. Křivka zde lineárně roste se zvyšujícím počtem spuštěných virtuálních strojů. Hodnota od první po 65. spuštěný virtuální stroj se zvýší o přibližně 18 milisekund. Což je podobné jako u metody VMware vSphere a znovu poloviční jako u metody Virtualbox.



Obr. 5.13: Graf závislosti latence z portu 22 na počtu virtuálních stanic pro KVM.



Obr. 5.14: Graf závislosti rychlosti komunikace SSH protokolu od navázání po ukončení TCP spojení na počtu virtuálních stanic pro KVM.

5.6 Porovnání naměřených výsledků

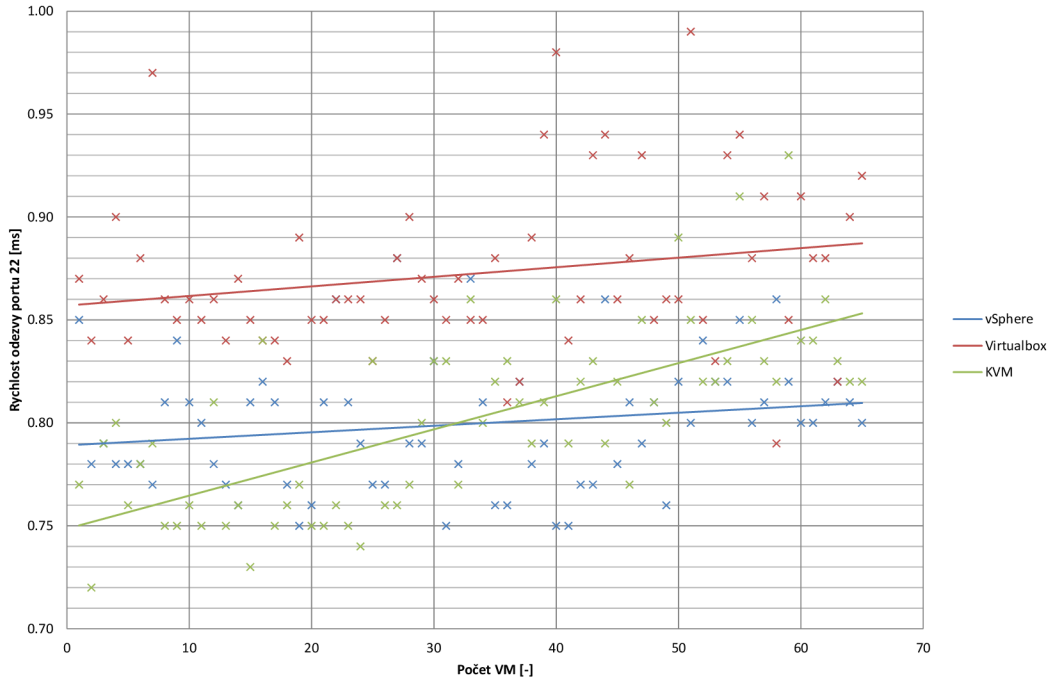
Všechny tři metody jsem následně zobrazil v jednom grafu, abych je následně mohl porovnávat. Podle předchozích výsledků by měla nejhůře skončit metoda Virtualbox, která jediná byla jen aplikací nainstalovanou na běžný operační systém.

První graf, viz obr. 5.15, zobrazuje závislost latence z portu 22 na počtu virtuálních stanic. Zde lze vidět, že nejvíce závislá na počtu virtuálních stanic je metoda KVM, která prudce stoupá oproti ostatním metodám. Ovšem nejhůře zde vyšel Virtualbox, který měl po celou dobu nejvyšší hodnoty latence. VMware vSphere se stále pohybuje kolem hodnoty 0,80 milisekund a při zvyšujícím se počtu virtuálních strojů vykazuje téměř stejné zpoždění. Pohybujeme se u všech metod ale v rozmezí dvou desetin milisekundy, což je zcela zanedbatelné zpoždění.

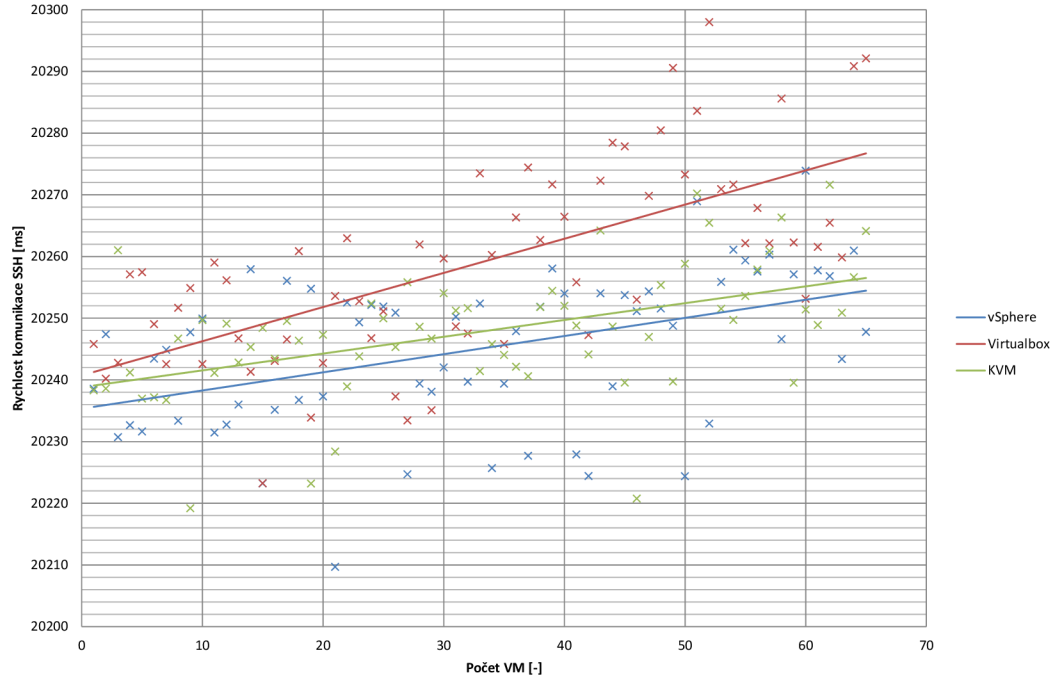
Druhý graf, viz obr. 5.16, zobrazuje graf závislosti rychlosti komunikace SSH protokolu od navázání po ukončení TCP spojení na počtu virtuálních stanic pro všechny tři metody. Zde je již změřená celá relace od příznaku SYN po příznak FIN a daleko více se zde promítlo zpoždění vznikající při rostoucím počtu spuštěných virtuálních strojů. Zde znovu s nejvyšší hodnotou zpoždění byl Virtualbox, který není vytvořený pro takovéto účely se spuštěním na serveru, ale spíše pro testovací, či laboratorní prostředí, kde je potřeba mít spuštěný minimální počet virtuálních strojů.

Podobně zde dopadly virtualizační nástroje KVM a vSphere. Tyto nástroje měli podobnou křivku, ovšem KVM mělo stále zpoždění větší o 2 až 3 milisekundy.

Obě tyto měření jsem provedl i pro stejný systém CentOS 7 ve verzi Minimal bez virtualizace, kde vyjdou referenční hodnoty pro porovnání. Pro první měření zde vyšla hodnota 0,77 milisekund, což je hodnota, ke které se podle obr. 5.15 přibližuje jen VMware vSphere s rozdílem 0,04 milisekund při 65-ti spuštěných virtuálních strojích. Druhé měření vyšlo 20231,87 milisekund, což je doba o přibližně 22 milisekund menší než pro metodu VMware vSphere a o 24 milisekund menší než u KVM pro 65 virtuálních strojů. Virtualbox zde byl o 55 milisekund pomalejší, než referenční metoda bez virtualizace.



Obr. 5.15: Graf závislosti latence z portu 22 na počtu virtuálních stanic.



Obr. 5.16: Graf závislosti rychlosti komunikace SSH protokolu od navázání po ukončení TCP spojení na počtu virtuálních stanic.

5.6.1 Analýza komunikace

Pokud se ale podíváme pozorněji na získané hodnoty, tak zjistíme, že je zde přes 20 sekund velké zpoždění, které přetrvává pro jakýkoliv počet virtuálních strojů, dokonce i pro měření bez virtualizace. Na obr. 5.17 lze vidět, že je toto zpoždění způsobeno z největší části kvůli vyhledávání reverzního DNS záznamu. Což je dáno výchozím nastavením u SSH serveru. Upravíme tedy na SSH serveru konfigurační SSH soubor umístěný ve složce `/etc/ssh/sshd_config`. Do tohoto souboru přidáme řádek `UseDNS␣no` a `GSSAPIAuthentication␣no`, kde první nám zakáže používání DNS a druhý zakáže GSSAPI autentizaci na SSH serveru, která prověřuje DNS záznam klienta připojujícího se na server.

No.	Time	Source	Destination	Protocol	Length	Info
360	423.094544	192.168.1.50	192.168.1.3	SSHv2	146	Server: Encrypted packet (len=80)
361	423.131280	192.168.1.3	192.168.1.50	TCP	66	42670→22 [ACK] Seq=1522 Ack=1998 Win=35200 Len=0 TSval=727845 TSecr=3410684
362	424.995284	fe80::5ef9:ddff:fe6…	ff02::fb	MDNS	105	Standard query 0x0000 PTR 50.1.168.192.in-addr.arpa, "QM" question
363	424.995386	192.168.1.3	224.0.0.251	MDNS	85	Standard query 0x0000 PTR 50.1.168.192.in-addr.arpa, "QM" question
364	433.000509	fe80::5ef9:ddff:fe6…	ff02::fb	MDNS	105	Standard query 0x0000 PTR 50.1.168.192.in-addr.arpa, "QM" question
365	433.000612	192.168.1.3	224.0.0.251	MDNS	85	Standard query 0x0000 PTR 50.1.168.192.in-addr.arpa, "QM" question
366	443.116786	192.168.1.3	192.168.1.50	SSHv2	210	Client: Encrypted packet (len=144)
367	443.130190	192.168.1.50	192.168.1.3	SSHv2	98	Server: Encrypted packet (len=32)
368	443.130242	192.168.1.3	192.168.1.50	TCP	66	42670→22 [ACK] Seq=1666 Ack=2030 Win=35200 Len=0 TSval=732844 TSecr=3430720

Obr. 5.17: Komunikace v programu Wireshark se zpožděním u sestavení SSH komunikace.

Zakázání reverzního překladu DNS jsem upravil na stroji se systémem CentOS 7 Minimal bez virtualizace. Poté jsem provedl znovu obě měření, kde u prvního s měřením latence z portu 22 vyšlo měření 0,71 milisekund, kde podle předpokladů úprava SSH serveru měla jen malý vliv na zpoždění. Ovšem u druhého měření už měření vyšlo pouze 0,2133 sekund. Z čehož jde s drobnou odchylkou vyvodit závěr, že reverzní vyhledávání DNS způsobuje zpoždění 20 sekund. Na obr. 5.18 lze vidět výstup z programu Wireshark, kde díky odpadnutí vyhledávání reverzního DNS zde zmizelo zpoždění o velikosti 20 sekund.

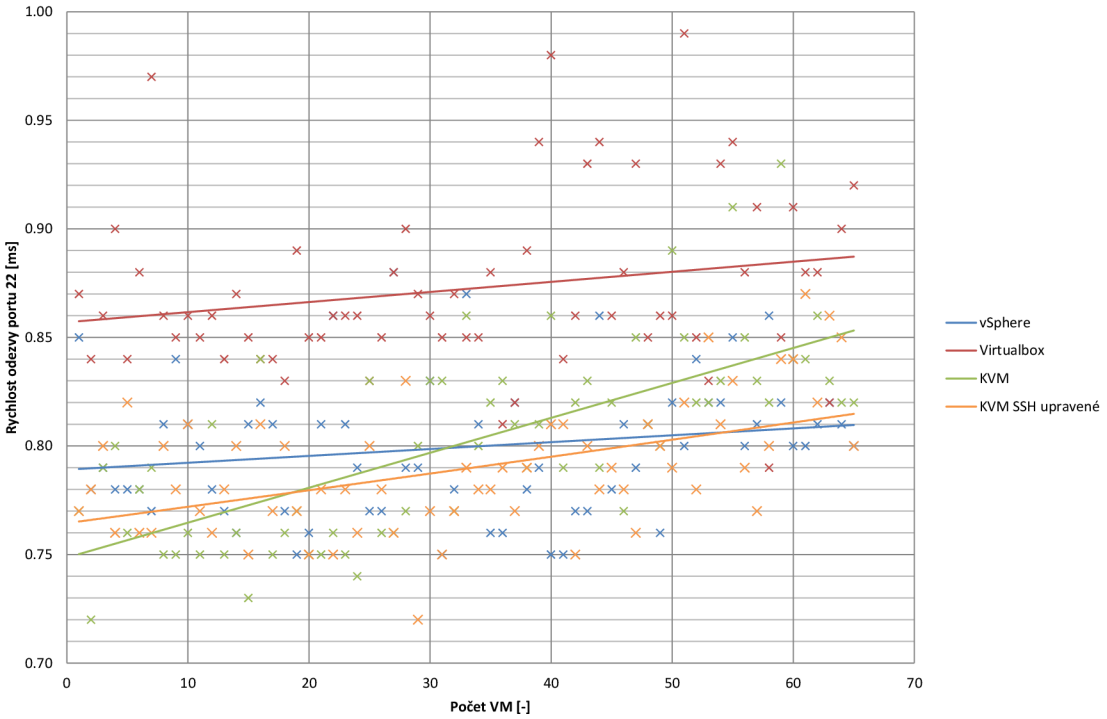
86	0.289787	192.168.1.3	192.168.1.50	SSHv2	82	Client: New Keys
87	0.329637	192.168.1.50	192.168.1.3	TCP	66	22→42664 [ACK] Seq=1870 Ack=1410 Win=17408 Len=0 TSval=4294773909 TSecr=393613
88	0.329695	192.168.1.3	192.168.1.50	SSHv2	114	Client: Encrypted packet (len=48)
89	0.329826	192.168.1.50	192.168.1.3	TCP	66	22→42664 [ACK] Seq=1870 Ack=1458 Win=17408 Len=0 TSval=4294773909 TSecr=393623
90	0.329848	192.168.1.50	192.168.1.3	SSHv2	114	Server: Encrypted packet (len=48)
91	0.332842	192.168.1.3	192.168.1.50	SSHv2	130	Client: Encrypted packet (len=64)
92	0.333288	192.168.1.50	192.168.1.3	SSHv2	146	Server: Encrypted packet (len=80)
93	0.336935	192.168.1.3	192.168.1.50	SSHv2	210	Client: Encrypted packet (len=144)
94	0.347722	192.168.1.50	192.168.1.3	SSHv2	98	Server: Encrypted packet (len=32)
95	0.347977	192.168.1.3	192.168.1.50	SSHv2	194	Client: Encrypted packet (len=128)
96	0.387650	192.168.1.50	192.168.1.3	TCP	66	22→42664 [ACK] Seq=2030 Ack=1794 Win=22528 Len=0 TSval=4294773967 TSecr=393628

Obr. 5.18: Komunikace v programu Wireshark bez zpožděním u sestavení SSH komunikace.

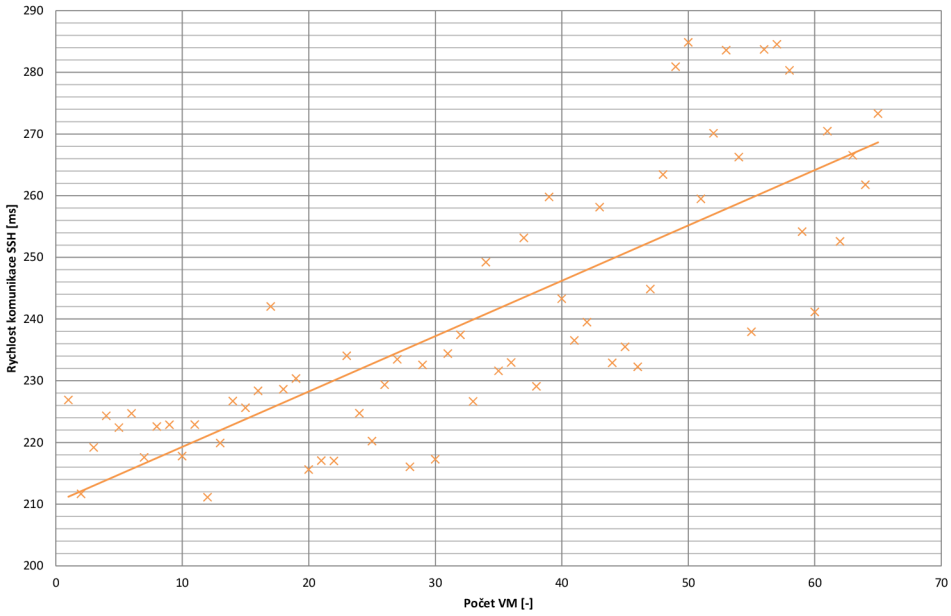
Stejnou úpravu jsem provedl u virtualizačního nástroje KVM a provedl znovu měření pro 65 virtuálních strojů. Veškeré výsledky jsem zaznamenal a znovu vynesl do grafů. První graf, viz obr. 5.19, zobrazuje závislost latence z portu 22 na počtu virtuálních stanic, kde jsem pro porovnání zobrazil i veškeré předchozí metody. Křivka je velice podobná křivce pro metodu VMware vSphere a nemá tak

strmé stoupání jako KVM bez upraveného SSH serveru. Ovšem tyto hodnoty se liší v rámci setin milisekundy, což je zcela zanedbatelné.

Graf na obr. 5.20 zobrazuje závislost rychlosti komunikace SSH protokolu od navázání po ukončení TCP spojení na počtu virtuálních stanic pro KVM s upraveným SSH serverem. Zde lze vidět, že pro všechny spuštěné virtuální stroje zmizelo zpoždění 20 sekund. Křivka se pohybuje v rozmezí 210 až 270 milisekund, má prudké stoupání v závislosti na počtu spuštěných virtuálních strojů a pohybuje se v rozmezí 60-ti milisekund.



Obr. 5.19: Graf závislosti latence z portu 22 na počtu virtuálních stanic pro všechny virtualizační metody.



Obr. 5.20: Graf závislosti rychlosti komunikace SSH protokolu od navázání po ukončení TCP spojení na počtu virtuálních stanic pro KVM s upraveným SSH serverem.

5.6.2 Porovnání hodnot

Porovnání hodnot podle výsledných křivek je v tab. 5.1. První tři řádky s hodnotami v tabulce určují výsledky pro první měření, pomocí aplikace psping, kde v první řadě jsou hodnoty pro spuštěný 1 virtuální stroj s výjimkou měření bez virtualizace, kde je měření jen na nevirtualizovaný systém. Další dva řádky jsou pro 35 a 65 virtuálních strojů. Dále je zde vypočítaný medián a směrodatná odchylka pro všechny typy vritualizací a pro stejný typ měření latence. Druhý blok tabulky zobrazuje časový rozdíl mezi měřením pro 1 virtuální stroj a vždy porovnávanou hodnotou, zobrazuje o kolik se zpoždění zpozdilo pro více virtuálních strojů u každé metody zvlášť. Stejným způsobem jsou zobrazeny i další dva bloky, ale jsou pro druhé měření, kde se měřil časový rozdíl mezi příznaky SYN a FIN.

Tab. 5.1: Tabulka srovnání průměrných naměřených hodnot u všech metod.

		Bez virtualizace	vSphere	Virtualbox	KVM	KVM SSH upravené
Měření latence z portu 22	1 VM t [ms]	0,77	0,79	0,86	0,75	0,77
	35 VM t [ms]	–	0,80	0,87	0,81	0,79
	65 VM t [ms]	–	0,81	0,89	0,85	0,82
	medián [ms]	–	0,80	0,86	0,80	0,79
	směrodatná odchylka σ[ms]	–	0,03	0,04	0,04	0,03
Měření latence z portu 22	1 VM Δt [ms]	–	0,00	0,00	0,00	0,00
	35 VM Δt [ms]	–	0,01	0,01	0,06	0,02
	65 VM Δt [ms]	–	0,02	0,03	0,10	0,05
Měření rychlosti komunikace SSH	1 VM t [ms]	20232	20236	20241	20239	211
	35 VM t [ms]	–	20246	20260	20248	242
	65 VM t [ms]	–	20255	20277	20257	269
	medián [ms]	–	20248	20259	20249	233
	směrodatná odchylka σ[ms]	–	13	16	10	21
Měření rychlosti komunikace SSH	1 VM Δt [ms]	–	0	0	0	0
	35 VM Δt [ms]	–	10	19	9	31
	65 VM Δt [ms]	–	19	36	18	58

6 ZÁVĚR

Z prezentovaných výsledků a grafů lze vidět, že řešení VMware vSphere potvrdilo svoji roli na poli virtualizace a jako nejosvědčenější řešení ukázalo, že pro počet 65-ti virtuálních stanic, zde nemá až na malou odchylku zpoždění v řádech deseti milisekund, žádný vliv na zpoždění komunikace pomocí protokolu SSH. O dvě milisekundy pro 65 virtuálních stanic bylo od této metody KVM, které dosahovalo obdobných výsledků jako vSphere, ovšem s tím rozdílem, že KVM je bezplatný balíček, který se dá nainstalovat do jakéhokoliv systému Linux. Nejhůře skončila metoda Virtualbox od firma Oracle, která ale patří do typu softwarových virtualizací a není určená pro takové množství virtuálních stanic.

Během komunikace jsem i objevil zpoždění o velikosti přibližně dvaceti sekund, které způsobuje vyhledávání reverzního překladu DNS. Pokud tedy tuto službu vyloženě nepotřebujeme, tak ji lze zakázat pomocí jednoduché úpravy konfiguračního souboru, jak je popsáno výše. Což nám dosti urychlí navázání komunikace se vzdáleným systémem.

V poslední části měření jsem provedl úpravu konfiguračního souboru SSH serveru u metody KVM. U prvního typu zpoždění pro aplikaci psping tato metoda měla lepší výsledky, než měření bez upraveného konfiguračního souboru. Druhé měření časového rozdílu mezi příznaky SYN a FIN dosahovalo pro počet 65-ti virtuálních strojů hodnoty zpoždění 269 milisekund. Což je oproti původní hodnotě 20,257 sekund obrovský rozdíl.

Velkou nevýhodou virtualizační metody KVM je fakt, že je závislá na použitém hardwaru a na přítomnosti operačního systému Linux. Během vlastního měření se mi několikrát stalo, že se server restartoval většinou kvůli kritické chybě integrované grafické karty, se kterou například systém Windows od firmy Microsoft nemá žádný problém.

LITERATURA

[1] RUEST, D. RUEST, N. *Virtualizace: podrobný průvodce*. Brno: Computer Press, 2010. Vydání první, 408 s. ISBN 978-80-251-2676-9.

[2] KAMALI, N. *How Virtualization Reduce Maintenance and Repair Cost of Computer Systems*. Neyshabur: 2008. Dostupné z URL: <http://www.academia.edu/1170817/How_Virtualization_Reduce_Maintenance_and_Repair_Cost_of_Computer_Systems>.

[3] Microsoft: *Windows Server Evaluations* [online]. 2014 [cit. 26. 11. 2014]. Dostupné z URL: <http://www.microsoft.com/en-us/evalcenter/evaluate-hyper-v-server-2012-r2>.

[4] Oracle VM VirtualBox: *User Manual* [online]. 2004-2014 [cit. 27. 11. 2014]. Dostupné z URL: <https://www.virtualbox.org/manual/UserManual.html>.

[5] LOWE, S. *Mistrostvi ve VMware vSphere 4: kompletní průvodce profesionální virtualizací*. Brno: Computer Press, 2010. Vydání první, 662 s. ISBN 978-80-251-2915-9.

[6] ROSA, J. a kol. *KVM Virtualization in RHEL 7 Made Easy* [online]. 2014 [cit. 14. 4. 2015]. Dostupné z URL: <http://linux.dell.com/files/whitepapers/KVM_Virtualization_in_RHEL_7_Made_Easy.pdf>.

[7] DOSTALEK, L. a kol. *Velký průvodce protokoly TCP/IP: Bezpečnost*. Praha: Computer Press, 2003. Druhé aktualizované vydání, 571 s. ISBN 80-7226-849-X.

[8] Information Sciences Institute: *TRANSMISSION CONTROL PROTOCOL*. 1981. Dostupné z URL: <https://www.ietf.org/rfc/rfc793.txt>.

[9] ODVARKA, P. *TCP handshake krok za krokem* [online]. 2000 [cit. 27. 11. 2014]. Dostupné z URL: <http://www.svetsiti.cz/clanek.asp?cid=TCP-handshake-krok-za-krokem-3122000>.

[10] POSTEL, J. *INTERNET CONTROL MESSAGE PROTOCOL*. 1981. Dostupné z URL: <https://tools.ietf.org/html/rfc792>.

[11] YLONEN, T. *The Secure Shell (SSH) Protocol Architecture* [online]. 2006 [cit. 29. 11. 2014]. Dostupné z URL: <http://www.ietf.org/rfc/rfc4251.txt>.

[12] Cisco: *Protocol Basics: Secure Shell Protocol* [online]. [cit. 29. 11. 2014]. Dostupné z URL: <http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_12-4/124_ssh.html>.

[13] OpenBSD: *OpenSSH keeping your communiques secret.* [online]. 2015 [cit. 30. 4. 2015]. Dostupné z URL: <http://www.openssh.com/>.

[14] CentOS: *Configuring an OpenSSH Server.* [online]. [cit. 30. 4. 2015]. Dostupné z URL: <https://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-openssh-server-config.html>.

[15] RUSSINOVICH, M. *PsPing v2.01* [online]. 2014 [cit. 29. 11. 2014]. Dostupné z URL: <http://technet.microsoft.com/en-us/sysinternals/jj729731.aspx>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ACK	Acknowledgement
AMD	Advanced Micro Devices
API	Application Programming Interface
BSD	Berkeley Software Distribution
BSOD	Blue Screen of Death
CPU	Central Processing Unit
DNS	Domain Name System
DSA	Digital Signature Algorithm
FIN	Finish
GB	Gigabyte
Gb/s	Gigabyte per second
GSSAPI	Generic Security Services Application Program Interface
IBM	International Business Machines Corporation
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
KVM	Kernel-based Virtual Machine
LTS	Long Term Support
MAC	Macintosh
MB	Megabyte
OEM	Original Equipment Manufacturer
OpenSSH	OpenBSD Secure Shell
PC	Personal Computer
PSH	Push Function

QEMU	Quick EMUlator
RAM	Random-access memory
RDC	Remote Desktop Connection
RSA	Rivest, Shamir, Adleman
RST	Reset
SEQ	Sequence number
SFTP	SSH File Transfer Protocol
SLA	Service-level agreement
SOCKS	Socket Secure
SSH	Secure Shell
SYN	Synchronise
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URG	Urgent
URL	Uniform Resource Locator
USB	Universal Serial Bus
VDI	Virtual Desktop Infrastructure
VM	Virtual Machine
Windows NT	Windows New Technology
Windows XP	Windows eXPerience
X11	X Window System