Česká společnost uživatelů otevřených systémů EurOpen.CZ
Czech Open System Users' Group
**www.europen.cz**

**44. konference**

# Sborník příspěvků

**Školicí středisko Herbertov**
**11.–14. 5. 2014**

**Programový výbor:**

Zdeněk Šustr
Pavel Šimerda
Jiří Sitera
Jan Kynčl

Sborník příspěvků z 44. konference EurOpen.CZ, 11.–14. 5. 2014

Plzeň 2014. První vydání.
Editor: Vladimír Rudolf
Sazba a grafická úprava: Ing. Miloš Brejcha – Vydavatelský servis, Plzeň
e-mail: servis@vydavatelskyservis.cz

Tisk: Typos, tiskařské závody, s. r. o.
Podnikatelská 1160/14, Plzeň



Příspěvky neprošly redakční ani jazykovou úpravou.

ISBN 978-80-86583-27-3

# Obsah

# OpenNebula – cloud na počkanie

**Boris Parák**

E-mail: xparak@mail.muni.cz

Tutoriál uviedol účastníkov do problematiky cloudu, stručne osvetlil bežné pojmy a všeobecné koncepty potrebné na pochopenie jeho fungovania. Po krátkom úvode nasledovalo predstavenie open source cloudovej platformy OpenNebula, jej architektúry, technických možností a typov nasadenia.

V praktickej časti tutoriálu si účastníci vyskúšali kompletnú inštaláciu a konfiguráciu platformy OpenNebula vo virtualizovanom prostredí na vlastných notebookoch. Následne si každý mohol vyskúšať základné použitie, či už cez grafické webové rozhranie Sunstone alebo shellové nástroje. Súčasťou demonštrácie bola tvorba virtuálnych sietí, obrazov a šablón virtuálnych strojov, zakončená ich spustením.

Záver tutoriálu sa stručne venoval problematike interoperability vo svete cloudu. Bol predstavený štandard OCCI (Open Cloud Computing Interface) a jeho implementácia pre platformu OpenNebula – projekt rOCCI. Po predstavení bola predvedená inštalácia a konfigurácia komponenty rOCCI-server a použitie klienta rOCCI-cli na spustenie virtuálnych strojov.

Praktická časť od účastníkov vyžadovala inštaláciu a použitie virtualizačného nástroja VirtualBox. Pre záujemcov, ktorí sa tutoriálu nemohli zúčastniť osobne alebo sa zúčastnili a chceli by vedieť viac, odporúčame nasledovné zdroje informácií:

- Dokumentácia cloudovej platformy OpenNebula [1]
- Inštalačné a konfiguračné tutoriály od autorov OpenNebuly [2]
- Špecifikácia štandardu OCCI [3]
- Zdrojáky a inštalačná dokumentácia komponenty rOCCI-server na GitHube [4, 5]
- Zdrojáky a dokumentácia komponenty rOCCI-cli na GitHube [6]

# Literatura

[1] http://opennebula.org/documentation

[2] http://opennebula.org/documentation/tutorials/

[3] http://occi-wg.org/about/specification/

[4] https://github.com/EGI-FCTF/rOCCI-server

[5] https://github.com/EGI-FCTF/rOCCI-server/wiki/rOCCI-Server-Admin-Guide

[6] https://github.com/EGI-FCTF/rOCCI-cli

# Ekonomika a cloud

**Jan Panoch**

E-mail: jan@panoch.net

**Abstrakt**

*Pojem cloud je dnes módním buzzwordem s bezpočtem významů, který pokrývá škálu od synonyma pro virtualizované servery, přes škálovatelný pronájem výpočetního výkonu až po ukládání dat na internetu. Všude a stále znovu je zmiňována a opakována jeho modernost, výhodnost, škálovatelnost a bezpečnost (případně nebezpečnost). Pokusím se vám přinést komplexnejší pohled a vysvětlit, že vše výše uvedené je v podstatě pravda – skoro vždy a skoro všude, ale ne vždy a ne pro každého. Použití cloudových služeb vám může ušetřit mnoho času a starostí, může pomoci vaší firmě rychle vyrůst nebo bez problémů přežít vašemu eshopu vánoční svátky – ale také může udělat pěknou díru do rozpočtu, pokud si nedáte pozor.*

## 1 Úvod

Termín cloud se dnes používá velmi často a pod tímto označením jsou chápány různé věci a služby. Můžeme si pod tímto termínem představovat pouze škálovatelný výpočetní výkon, který je dostupný snadno a každému, kdo si ho objedná a zaplatí. Můžeme jít o něco dál a vidět za ním kromě surového výpočetního výkonu celou paletu dalších služeb, jak je nabízí například Amazon a jeho AWS. Můžeme mít tento termín spojený s Apple iCloud, který slouží k ukládání a sdílení vašich dat z telefonu, tabletu a počítače někde v síti, nebo to může být jen úložiště pro data různých aplikací, kde jejich tvůrce chtěl nabídnout ukládání Vašich dat na svých serverech za účelem data-miningu a nazývá toto úložiště moderním termínem cloud.

## 2 Co je vlastně cloud?

Zkusme tedy popsat co tento termín znamená z několika základních pohledů.

### Pohled z hlediska servisního nebo distribučního modelu

**IaaS** (Infrastructure as a service, infrastruktura jako služba) je spolu s PaaS (Platform as a service, platforma jako služba) a populárním SaaS (Software as a service, software jako služba) jedním z distribučních modelů cloud computingu. Někdy je označován jako jeden z nejméně vyvinutých distribučních modelů. Oproti vlastnictví, nebo outsourcingu si zde zákazník pronajímá škálovatelnou infrastrukturu, kde platí za využití informačních technologií, podobně jako za telefon. Platí se zde většinou za množství uložených dat, nebo čas-procesoru. Příjemce většinou nezajímá a ani nemůže zjistit, kde se pronajímaný hardware fyzicky nachází.

**PaaS** – platforma jako služba (z „Platform as a Service“) – poskytovatel v modelu PaaS poskytuje kompletní prostředky pro podporu celého životního cyklu tvorby a poskytování webových aplikací a služeb plně k dispozici na Internetu, bez možnosti stažení softwaru. To zahrnuje různé prostředky pro vývoj aplikace jako IDE nebo API, ale také např. pro údržbu. Nevýhodou tohoto přístupu je proprietární uzamčení, kdy může každý poskytovatel používat např. jiný programovací jazyk. Příkladem poskytovatelů PaaS jsou Google App Engine nebo Force.com (Salesforce.com).

**SaaS** – software jako služba (ze „Software as a Service“) – aplikace je licencována jako služba pronajímaná uživateli. Uživatelé si tedy kupují přístup k aplikaci, ne aplikaci samotnou. SaaS je ideální pro ty, kteří potřebují jen běžné aplikační software a požadují přístup odkudkoliv a kdykoliv. Příkladem může být známá sada aplikací Google Apps, nebo v logistice známý systém Cargopass.

### Pohled z hlediska implementačního modelu

Veřejný cloud – služby jsou poskytovány veřejně, uživatel nebo zákazník nemá žádný vztah a kontrolu nad hardwarem a systémy, na kterých je cloud provozován a obvykle ani neví, kde se jeho data nacházejí

Privátní cloud – hardware cloud je ve Vaší vlastní správě a majetku, máte nad ním plnou kontrolu, staráte se o jeho provoz, opravy, rozšiřování, zálohování apod.

Hybridní cloud – spojuje výhody a privátního a veřejného cloudu, kdy můžete část služeb provozovat ve veřejném a část v privátním. Obvykle je možná i jednoduchá migrace systémů a služeb mezi privátní a veřejnou částí

## 3 Srovnání způsobů provozu IT ve firmě

Pro ilustraci se pokusím charakterizovat a zhodnotit výhody a nevýhody různých způsobů provozu informačních technologií ve firmě.

## Provoz vlastního IT

- Veškeré vybavení je majetkem společnosti
- Vlastní IT oddělení, vlastní správce IT

**Výhody:**

- V případě finančních problémů firmy schopnost „přežít“ s tím, co firma má.
- Nezávislost na externím dodavateli.

**Nevýhody:**

- Pravidelné investice do provozu IT.
- Udržování finanční rezervy pro nepředvídatelné události nebo poruchy a pravidelný upgrade HW a zajišťování drahých licencí.
- Skryté náklady na provoz lokálního řešení IS (např. vysoká spotřeba elektrické energie).
- Rychlé morální zastarávání lokálních počítačů, nutnost jejich častější výměny, upgrade a pořizování novějších licencí.
- Vysoké provozní náklady (nákup HW/SW, mzdové náklady na správce, zajištění silné internetové konektivity, spotřeba elektrické energie, údržba a provoz „serverovny“ a lokálních stanic)
- Ve většině případů časté a dlouhé výpadky provozu IT v důsledku poruch, nebo chybovosti správce IT, špatné zálohování dat.
- Nebezpečí ztráty nebo zneužití firemních dat v důsledku žádného, nebo málo bezpečného zálohování, nebezpečí požáru, vloupání, neopatrnosti při úklidu, nebo v důsledku nekalých úmyslů vlastních zaměstnanců, vlivu virů, škodlivých programů a síťového útoku hackerů, přepětí (například při bouřce), záloha elektrického napájení nad rámec UPS.
- Nízká odbornost a specializace IT správce. Jeden člověk nemůže pokrýt širokou škálu odborností, které je v IT potřeba. IT správce by měl alespoň 4× do roka absolvovat školení a zkoušky. Toto se neděje prakticky v žádné společnosti, která si IT provozuje sama.

## Provoz vlastního IT, s externí správou

**Charakteristika:**

- Veškeré vybavení IT je majetkem společnosti
- Správu systému zajišťuje externí firma

**Výhody:**

- Odborný způsob správy systému
- Zajištění dostupnosti správce 24 hodin denně, 7 dní v týdnu

- Odpovědnost za provoz a licencování spadá na externího IT správce
- Nižší náklady na zajištění správy IT, oproti vlastnímu správci IT (mzdové náklady a podobně)

**Nevýhody:**

- Pravidelné investice do provozu IT.
- Udržování finanční rezervy pro nepředvídatelné události nebo poruchy a pravidelný upgrade HW a zajišťování drahých licencí.
- Skryté náklady na provoz lokálního řešení IS (např. vysoká spotřeba elektrické energie).
- Rychlé morální zastarávání lokálních počítačů, nutnost jejich častější výměny, upgrade a pořizování novějších licencí.
- Vysoké provozní náklady (nákup HW/SW, mzdové náklady na správce, zajištění vysoké internetové konektivity, spotřeba elektrické energie, údržba a provoz „serverovny“ a lokálních stanic)
- Ve většině případů časté a delší výpadky provozu IT v důsledku nutnosti cestování externího IT správce a v případě potřeby zajištění náhradních dílů pro znovuuvedení infrastruktury do provozu.
- Nebezpečí ztráty nebo zneužití firemních dat v důsledku žádného, nebo málo bezpečného zálohování, nebezpečí požáru, vloupání, neopatrnosti při úklidu, přepětí (například při bouřce), nebo v důsledku nekalých úmyslů vlastních zaměstnanců, vlivu virů, škodlivých programů a síťového útoku hackerů.

## Provoz vlastního IT, s externí správou s umístěním serverů do datového centra (housing)

**Charakteristika**

- Veškeré vybavení IT je majetkem společnosti
- Vlastní IT oddělení, vlastní správce IT nebo externí IT správce

**Výhody:**

- Zajištění vysoké dostupnosti konektivity infrastruktury i pro potřeby vně společnosti (zaměstnanci v terénu apod.)
- Vysoké zabezpečení infrastruktury v datovém centru (fyzický přístup s ostrahou, protipožární systém, energetické zajištění, ochrana proti přepětí, vysoká čistota vzduchu, kvalitní chlazení a další)
- Rychlá dostupnost, například personálem datového centra, který je schopen zajistit výměnu dílů v případě poruchy, nebo jen jednoduchý restart, nebo zapnutí serverů.
- Možnost zálohování IS na úložiště poskytnuté datovým centrem

**Nevýhody:**

- Pravidelné investice do provozu IT.
- Udržování finanční rezervy pro nepředvídatelné události nebo poruchy a pravidelný upgrade HW a zajišťování drahých licencí.
- Skryté náklady na provoz lokálního řešení IS (např. vysoká spotřeba elektrické energie pro potřeby lokálních stanic).
- Rychlé morální zastarávání lokálních počítačů, nutnost jejich častější výměny, upgrade a pořizování novějších licencí.
- Vysoké provozní náklady (nákup HW/SW, mzdové náklady na správce, údržba a provoz lokálních stanic i serverů v datovém centru)
- Ve většině případů časté a delší výpadky provozu IT v důsledku nutnosti cestování externího IT správce a v případě potřeby zajištění náhradních dílů pro znovuuvedení infrastruktury do provozu.
- Nebezpečí ztráty nebo zneužití firemních dat v důsledku v důsledku nekalých úmyslů vlastních zaměstnanců, vlivu virů, škodlivých programů a síťového útoku hackerů.
- Závislost na externím dodavateli

## Provoz IT v cloudu formou služby, se vzdálenými přístupy do pracovních stanic

**Charakteristika:**

- Provoz IT v plné míře zajišťuje poskytovatel
- Přístup do počítačů uživatelů je zajištěn terminálově, prostřednictvím internetu. To znamená, že uživatelé se připojují ke svému počítači vzdáleně, pomocí terminálu. Terminálem může být: vlastní pracovní počítač – i velmi starý, nebo notebook s jakýmkoli operačním systémem, tenký klient, tablet, mobilní telefon
- HW a SW zajišťuje v podstatné míře poskytovatel, formou pronájmu

**Výhody:**

- Provoz IT má předvídatelné a fixní náklady, bez nutnosti výraznějších investic. Investice se mohou týkat například jen nákupem tiskových zařízení a terminálů.
- Finanční prostředky za provoz IT jsou tzv. „provozní náklady“
- Nízké celkové provozní náklady za IT.
- Je zajištěná vysoká bezpečnost IS – vyplývá již ze způsobu jejich provozu.
- Přístup do počítačů je zajištěn odkudkoli. Všichni uživatelé mají přístup do sdílených složek, možnost tisku odkudkoli, kdekoli ve firmě.

- Přístup do uživatelských počítačů je multiplatformní. To znamená, že se uživatel může do svého počítače připojit pomocí jakéhokoli počítače (PC, notebook, netbook, tablet, mobilní telefon) s jakýmkoli operačním systémem (Windows, Mac OS, Linux, Android, Symbian a pod.).
- Nezávislost uživatelských virtuálních počítačů (běžící ne serverech datového centra) na místním HW. Uživatel se může z pracoviště připojit ke svému počítači například z místní pracovní stanice, doma se může připojit pomocí notebooku, nebo tabletu. I rozpracovanou práci je možné dokončit z jiného – i cizího počítače.
- Tento způsob provozu IT je velmi pružný, kopíruje nové trendy, HW i SW jsou pravidelně modernizované. Data jsou v bezpečí datového centra.
- V případě poruchy terminálového zařízení není nutné, aby uživatel po dobu jeho servisu neměl přístup ke svému virtuálnímu počítači. Může se připojit z jakéhokoli dostupného počítače, notebooku, tabletu a podobně.
- Provozovatelé služby musí splňovat nejpřísnější kritéria bezpečnosti, vyloučit chyby jednotlivce. Vše musí být podloženo certifikacemi a prověrkami NBÚ.
- V případě, že poskytovatel zajišťuje i správu systému, je zodpovědný za licenční správnost všech SW v systému.
- Garantované SLA (Service-level agreement), možnost využití SLA 24 hodin, 7 dní v týdnu.

**Nevýhody:**

- Závislost na přístupu k internetovému připojení.
- Závislost na externím dodavateli.

# 4 Závěr – Co je lepší?

Na tuto otázku neexistuje jednoduchá odpověd. Výše uvedené srovnání vzbuzuje zdání, že pro provoz informačních technologií běžné firmy by mělo být lepší a levnější nepořizovat vlastní hardware, ale pronajímat si potřebné služby v cloudu. Kamenem úrazu bývá často právě slovíčko „běžné“ – pokud se vaše specifické potřeby nevejdou do té správné škatulky standardně poskytovaných služeb, tak tento pronájem buď nebude možný vůbec nebo jeho ekonomická výhodnost nemusí splňovat Vaše očekávání.

Samostatnou kapitolou bývá provozování systémů, které vyžadují stabilní vysoký výkon bez velkých výkyvů, nevyžadují škálování a kromě toho máte šikovného a schopného správce – v tomto případě stále obvykle bývá levnější vlastní hardware umístěný ve Vaší firmě nebo ve vhodném datacentru.

# Principy budování datového centra VŠB-TU Ostrava

**Martin Pustka**

E-mail: Martin.Pustka@vsb.cz

Datové centrum VŠB-TU Ostrava bylo od počátku budováno jako technický celek, který lze skládat z jednotlivých komponent (LAN i SAN sítě, serverové systémy, disková úložiště, virtualizační technologie). Při návrhu i realizaci byl kladen důraz na dobrou integraci těchto technologií a využití moderních IT prostředků, jako jsou například technologie konvergovaného Ethernetu, virtualizačních serverů, virtuální přepínačů, chytrých diskových polí apod.

## Historie budování DC

Počátky budování dnešní podoby datového centra byly započaty zhruba v druhé polovině roku 2010, kdy proběhly první praktické testy konvergovaného Ethernetu v prostředí sítě VŠB-TUO. Následně proběhly interní diskuse o technické podobě a službách nového datového centra, ze kterých vzešly následující základní technicko-koncepční požadavky:

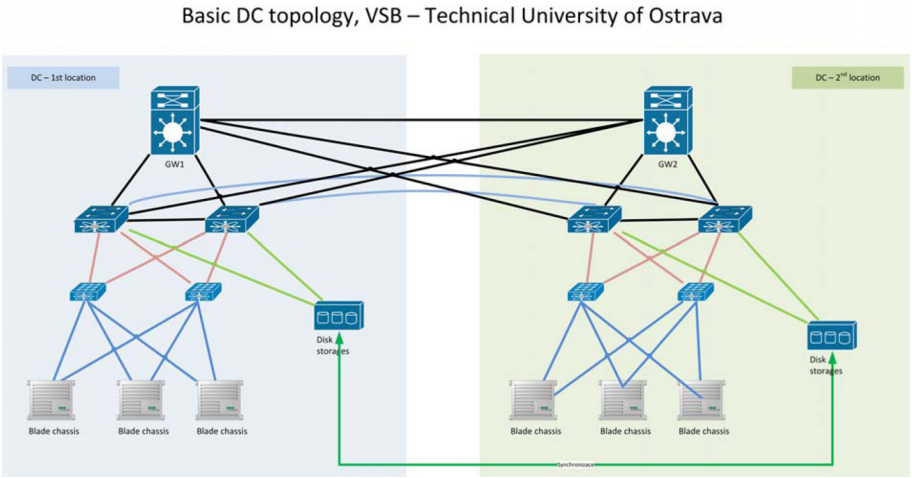
- použití konvergovaných 10GE technologií
- virtualizace serverové infrastruktury
- zajištění redundance $N + 1$ v každé technické vrstvě
- implementace chytrých škálovatelných diskových polí s podporou konvergovaného ethernetu, thin provisioningu, deduplikací a poskytování dat přes protokoly FC a IP
- požadavek na technicko-geografické rozdělení DC technologií do min. 2 center
- jednotná správa fyzické serverové infrastruktury s podporou konvergovaného ethernetu
- rozšiřování kapacit DC bez výpadku provozovaných technologií
- poskytování služeb DC i mimo oddělení Centra informačních technologií
- snaha o maximální integraci a další využití již pořízených technologií

Proběhly také testy několika technologií, ze kterých později vzešly další základní kameny datového centra – disková úložiště NetApp a serverová infrastruktura Cisco UCS. Později proběhlo doplnění této fyzické infrastruktury o další kapacity – serverové systémy, diskové police a proběhlo přepojení interní infrastruktury diskových polí NetApp do technologií tzv. metroclusteru, který umožňuje geografické rozdělení.

## Technické prostředky na datového centra

Celé datové centrum je složeno z několika částí, které lze v budoucnu obměnit tak, aby služby celku byly poskytovány bez významnějších dopadů do jiných částí. Celé datové centrum je od počátku budováno tak, aby mohlo být rozděleno do dvou geograficky oddělených lokalit. Dále je rozděleno do následujících částí – vrstev:

- směrovače IP sítě
- L2 přepínače datového centra s podporou konvergovaného ethernetu a FC protokolu
- disková pole
- fyzická serverová infrastruktura
- virtuální infrastruktura
- virtuální L2 přepínače




Obr. 1

## Směrovače počítačové sítě

Směrovače jsou tvořeny dvěma L3 přepínači Cisco Catalyst 6500. Oba jsou v režimu active-active, kde záložní systém v rámci datového centra zajišťuje zálohu primárnímu systému prostřednictvím protokolů HSRP/VRRP.

Z pohledu datového centra tyto přepínače podporují také technologii VRF (Virtual Routing and Forwarding). Tato technologie je využívána pro směrování sítí, které nejsou přímou součástí interní počítačové sítě VŠB a není tak potřeba zajišťovat pro tyto účely další aktivní prvky.

## Datacentrové přepínače

Přepínače datového centra jsou reprezentovány dvěma DC přepínači Cisco Nexus 5500. Přepínače podporují klasický LAN/Ethernet provoz, SAN/FC provoz a také technologie konvergovaného 10GE. Přepínače mají oddělenou správu tak, aby i konfigurační chyba jednoho z prvků neohrozila provoz DC jako celku. Směrem k okolním prvkům využívají technologie vPC (virtual port-channel), takže ani výpadek jednoho z přepínačů nemá vliv na výpadek provozu.

## Disková pole

Do infrastruktury DC byly pořízena disková pole NetApp, v redundantním zapojení, s podporou deduplikací, thin provisioningu, podporou protokolu NFS a s podporou redundantního geografického rozdělení do dvou lokalit.

## Fyzická serverová infrastruktura

Poměrně zásadní změnou v serverové infrastruktruře bylo nasazení konsolidovaného řešení. Základní požadavky na serverovou infrastrukturu obnášely podporu konvergovaného Ethernetu, jednotnou a vzdálenou správu serverových systémů, flexibilitu při změnách. Těmto požadavkům nejlépe vyhovělo řešení Cisco UCS. Toto řešení kromě výše uvedených vlastností přišlo také s velmi flexibilním přístupem k provozu a konfiguraci fyzických serverů.

## Virtuální infrastruktura

Jako platformu pro virtualizační infrastrukturu jsme již dlouhodobě používali technologie VMWARE vSphere ve verzi Full Enterprise. Největší změnou tak bylo doplnění virtualizační infrastruktury o distribuovaný softwarový přepínač Cisco Nexus 1000V a také nová multiplatformní služba „USB over IP“.

Virtuální přepínače Cisco Nexus1000V jsou virtuální přepínače, které umožňují správě sítě přístup až k jednotlivým virtuálním systémům. Změny v nastavení sítě se aplikují okamžitě i do rozhraní VMWARE vSphere a tak správce

virtuální infrastruktury má přehled o stavu a zároveň správa počítačové sítě má dostatek možností konfigurace sítě, které jsou téměř shodné s fyzickou infrastrukturou.

Technologie USB over IP nám umožnila ve virtuální infrastruktuře zprovoznit např. licenční servery, které vyžadují fyzické USB klíče, ale také SMS bránu. Server poskytující USB zařízení konkrétním virtuálním strojům je klasický fyzický server s OS Debian/GNU Linux a USB hubem. Klientská aplikace podporuje OS Linux i Windows a více než dvouletá provozní zkušenost s programem firmy Eltima ukazuje, že se jedná o velmi stabilní řešení.

# Poskytování služeb

V rámci datového centra poskytujeme následující služby:

- dedikovaný fyzický server
- virtuální server nebo celé virtuální infrastruktury
- diskové kapacity pro prostředky provozované v datovém centru
- zálohování, monitoring a notifikace

Dedikovaným serverovým systémem se rozumí server umístěný v blade šasi integrované serverové infrastruktury. Do provozu jsme schopni technicky zapojit i jiné systémy, ale u těchto nejsme schopni zajistit jejich vzdálenou správu a redundantní systém pro případ poruchy.

Virtuálním serverovým systémem se rozumí klasický virtuální systém s definovaným počtem vCPU, vRAM a diskovou kapacitou umístěnou na diskových polích datového centra.

Diskové kapacity diskových úložišť jsme schopni poskytnout na fyzické systémy serverové infrastruktury protokoly FC, alternativně také protokolem NFS.

Jednotlivé serverové systémy lze zálohovat na zálohovací jednotky a to klasickým způsobem, kdy v rámci provozovaného OS je instalován agent a záloha se pak provádí každý den programovým vybavením HP DataProtector.

Doplňkovou službou je také monitoring systémů a služeb. Pro monitoring je využit open-source nástroj Nagios, nezávislá SMS brána komunikující přímo s GSM sítí mobilního operátora a nezávislý SMTP server.

# Praktické zkušenosti z provozu

## Finanční úspory

Už při kalkulacích, které jsme dělali před pořízením technických prostředků datového centra, se ukazovalo, že dojde k úsporám. K úsporám došlo v oblasti

investic do hardware, protože jsme nemuseli pořizovat dvojitou infrastrukturu LAN/SAN sítí a s tím související kabeláže, transceivery atd.

Druhou úsporou byly náklady na provoz. Masivní virtualizací, použitím energeticky vhodně navržených prvků jsme výrazně zmenšili počet fyzické techniky a náklady na energie a chlazení. Také ceny technicky relativně jednoduchých serverů jsou nižší než při pořizování samostatných serverů, zřejmě i díky použití CNA karet (místo LAN/FC rozhraní).

Mezi provozní úspory lze zařadit i úsporu na licencích a také servisních poplatcích za hardware, u kterého jsme si díky dostatečné redundanci mohli dovolit horší a levnější typ servisních smluv.

Díky lepšímu sdílení (virtualizace, disková pole, počítačová síť) došlo k vyššímu využití pořízeného hardware, jen deduplikace dat v praxi ukazuje úsporu cca 30 %.

## Rozvoj datového centra v dalších letech

Datové centrum je platforma, nad kterou jsme schopni velmi pružně budovat další služby i komplexní virtualizované celky. V nejbližší době počítáme s ověřením konceptů, popř. i testovacím provozem následujících služeb:

- služby počítačové sítě ve virtuální infrastruktuře (VPN, směrovače)
- propojování datových center
- propojování virtualizačních infrastruktur

Tím, že datové centrum je složeno z několika částí, tak lze při dalších obnovách v budoucnu řešit obnovu jednotlivých částí. Obnovou jedné z části není narušen koncept celého datového centra a není nutno realizovat celkovou obnovu, čímž lze rozložit v čase investice do obnovy.

## Závěr

Obnovu datového centra, která započala před několika lety lze hodnotit jako poměrně úspěšný krok. Na jeho konci jsme docílili kvalitativně lepších služeb, došlo také k úsporám v investicích a zejména ke snížení nákladů na provoz a údržbu infrastruktury.

Tyto nové technologie umožňují také flexibilněji a vzdáleně řešit provozní problémy a spolu s nasazením virtuální infrastruktury učinit celou IT infrastrukturu výrazně pružnější a škálovatelnější i odolnější vůči poruchám.

Tato infrastruktura však vyžaduje kvalitní technické znalosti technického personálu, protože případné problémy se obvykle řeší napříč provozovanou infrastrukturou a není vždy na první pohled jednoznačně vidět, kde se nachází v celém řetězci problém.

V oblasti počítačových sítí jsme vyřešili problémy s kapacitně nedostatečnými 1GE linkami, které nešlo principiálně vyřešit ani sdružováním 1GE kanálů, a také s redundantním zapojením významných serverů.

Díky nasazení virtuálních přepínačů ve virtuální infrastruktuře se podařilo vyřešit problémy s jednotnou správou počítačové sítě, díky které mohou síťoví administrátoři aplikovat síťové politiky a řešit síťové problémy virtuálních systémů přímo až na virutálních rozhraních virtualizovaných systémů.

# CFEngine – správa cloudu

**Marek Petko**

E-mail: petko.marek@gmail.com

**Abstrakt**

*Masový rozvoj internetu a online služeb vede k neustálému zvyšování počtu serverů na celém světě. Od okamžiku, kdy k zajištění jedné služby přestal stačit jeden server, jsou vyvíjena různá distribuovaná řešení jako clustery, virtualizace a v poslední době cloud. Tato řešení mají společně za cíl distribuovat zátěž mezi více uzlů, přičemž celý systém musí být vysoce odolný proti výpadku. Virtualizace a cloud toto dále rozšiřuje o vysokou přenositelnost a flexibilitu. Ve výsledku je možné navyšovat výkon systému navýšením počtu serverů v řádu minut dle obchodních požadavků. Technické překážky tedy byly překonány, ale jak spravovat takto rozsáhlé systémy efektivně? Pokud počet serverů v čase roste exponenciálně, není možné, aby počet administrátorů byl přímo úměrný počtu spravovaných serverů. Pokud by tomu tak bylo, pak by firmy jako Google, Microsoft, Akamai, Facebook, Amazon a další pravděpodobně nikdy nedosáhly úspěchu. Cílem administrátora tedy musí být oprostit se od opakujících se činností, aby se mohl soustředit na vytváření nových řešení. K tomuto účelu slouží nástroje pro distribuovaný konfigurační management jako CFEngine, Puppet, Chef a další.*

*V prezentaci jsou nejprve představeny hlavní nástroje pro konfigurační management, dále se prezentace zabývá pouze nástrojem CFEngine 3. Popisuje postup základní instalace z různých zdrojů, základní principy distribuce konfiguračních politik a jejich zpracování. Přináší náhled na základní konstrukce deklarativního jazyka CFEngine a základní knihovnu. V další části se zabývá vhodnou organizací konfiguračních souborů a jejich správou verzovacím systémem GIT. Porovnává vlastnosti komunitní a komerční verze. Na závěr prezentuje zkušenosti z testovacího a produkčního prostředí na Západočeské univerzitě v Plzni.*

## 1 Úvod

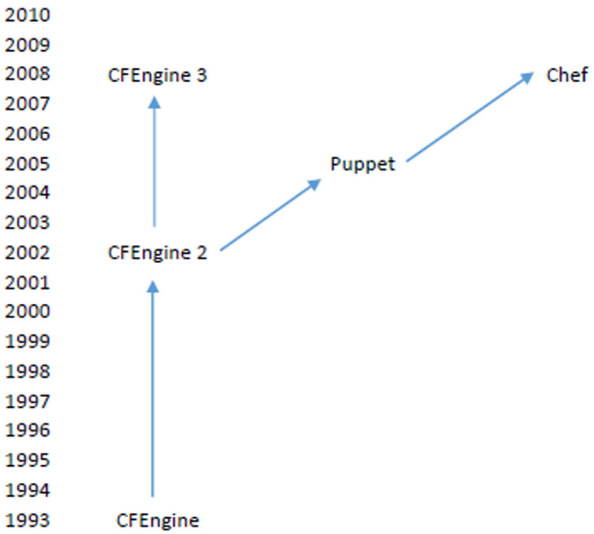
Lidé jsou velice efektivní ve vymýšlení nových řešení a v rozhodování, ale nejsou příliš spolehliví při provádění opakovaných operací. Naproti tomu stroje nejsou příliš dobré v rozhodování, ale o to více jsou spolehlivé při provádění předem

připravených řešení. S rostoucím počtem spravovaných zařízení se tyto rozdíly ještě více prohlubují. Cílem hromadného konfiguračního managementu je obě tyto role oddělit. Uživatel zná požadovaný cíl a umí ho deklarovat, stroj umí takovou deklaraci přeložit na posloupnost kroků a vykonat ji.

# 2 Technologie

Nejznámějšími nástroji pro hromadný konfigurační management je tzv. triáda konfiguračního managementu:

- CFEngine
- Puppet
- Chef




Obr. 1

Všechny tři nástroje jsou hojně rozšířeny a vychází z CFEngine 1, viz časová osa na Obr. 1. Díky společnému původu všechny nástroje sdílí podobný přístup ke konfiguraci. Ani jeden z nástrojů neprovádí jednorázové spuštění skriptů pro dosažení požadovaného stavu. Namísto toho je požadovaný stav definován konfigurační politikou a soulad s ní je automaticky pravidelně ověřován. Pokud je detekován nesoulad, jsou podniknuty kroky pro jeho odstranění. Tím se systém stává do jisté míry samo-spravujícím. V běžném systému by byl nežádoucí stav detekován monitorovacím softwarem (např. Nagios), na který by

administrátor musel reagovat a ručně zasáhnout. V případě zmíněných nástrojů je oprava provedena automaticky.

Všechny tři nástroje jsou dostupné v open source verzi a placené verzi.

Protože jsou si nástroje velice podobné, volba některého z nich může být komplikovaná. Každý z nástrojů má specifické vlastnosti, které zastánci a odpůrci považují za více či méně důležité.

CFEngine je postavený na pevném teoretickém základu teorie slibů. Je vytvořený v C a je tudíž velice efektivní a nepotřebuje dodatečný software (mimo knihoven při kompilaci). Pro zápis politik používá vlastní deklarativní jazyk, který je jednotný a vychází z teorie slibů. Pochopení jazyka je hodnoceno jako relativně složité. Dokumentace není příliš rozsáhlá, stejně tak příspěvky uživatelské komunity. Jednou z klíčových vlastností CFEngine je vysoký důraz kladený na bezpečnost a žádné zaznamenané bezpečnostní incidenty pro verzi 3. V open source variantě neobsahuje rozhraní pro zpětnou vazbu.

Puppet je vytvořený v jazyce Ruby, je tedy závislý na tomto prostředí a od něj se odvíjí i jeho výsledná efektivita. Pro zápis politik používá vlastní deklarativní jazyk, u kterého je kladen důraz na jednoduchost, není ale jednotný. Dokumentace je velice rozsáhlá, stejně jako uživatelská komunita. V databázi NIST je pro Puppet vedeno několik desítek záznamů o bezpečnostních mezerách. V open source variantě obsahuje základní webové rozhraní pro zpětnou vazbu.

Chef je taktéž vytvořený v jazyce Ruby a je na něm podobně závislý jako Puppet. Pro zápis politik používá Ruby, tím pádem je tvorba kódu velice snadná, ale postrádá výhody deklarativního jazyka. Dokumentace je dobrá, ale uživatelská komunita je malá. Databáze NIST obsahuje pro Chef jen několik záznamů. V open source variantě poskytuje webové rozhraní pro zpětnou vazbu.

Pro testování na CIV ZČU byl vybrán CFEngine 3, protože pro svůj běh nepotřebuje dodatečný software, k tomu instalace klienta zabírá pouze 20 MB místa. Solidní teoretický základ a jednotný programovací jazyk je pro použití v akademickém prostředí výhodou. Další důležitou výhodou je vysoká úroveň bezpečnosti.

# 3 CFEngine

CFEngine (ConFiguration Engine) je historicky prvním nástrojem pro konfigurační management. [1] Projekt CFEngine začal v roce 1993 na *University of Oslo* v Norsku, zpočátku pro usnadnění práce administrátora Unix systémů. V roce 2008 byla dokončena poslední přepracovaná verze CFEngine 3, která je založena na teorii slibů (Promise Theory) Marka Burgese. Současně byla založena společnost CFEngine AS, která začala vyvíjet komerční verzi jako rozšíření otevřené komunitní verze. [2]

## 3.1 Komerční a komunitní verze

CFEngine 3 je dostupný ve dvou variantách:

- Community Edition
- Enterprise Edition (Nova)

*Community Edition* je verze vyvíjená jako open source projekt s licencí *GNU General Public License 3*. Zdrojové kódy jsou veřejně dostupné na serveru Github.com[1]. [3]

*Enterprise Edition* je komerční verze rozšiřující funkcionalitu komunitní verze. Základem je komunitní zdrojový kód, který je dále rozšířen o neveřejný kód implementující placené funkce. [4]

Komerční verze je rozšířená o:

- zpětný sběr dat od klientů,
- *Mission Portal GUI* rozhraní,
- funkce pro orchestraci distribuovaného systému,
- kompresi a pokročilé šifrování komunikace,
- podporu dalších platforem.

V komerční verzi je dále v ceně zahrnuta technická podpora a konzultace při nasazování. [5]

## 3.2 Podporované operační systémy

CFEngine *Community Edition* může být přeložen a spuštěn na většině POSIX kompatibilních operačních systémů. Zdrojový kód je běžně překládán a testován na operačních systémech:

- GNU/Linux,
- Solaris,
- Windows s použitím MinGW.

Kód je testován a překládán na platformách:

- x86,
- x86-64,
- SPARC. [6]

[1] https://github.com/cfengine

Pro rychlou instalaci a aktualizaci nových verzí jsou udržovány repositáře s instalačními balíky pro operační systémy Debian[2], Red Hat[3] a jejich klony. [7]

CFEngine *Enterprise Edition* je dostupný pouze jako přeložený instalační balík. Podporované operační systémy jsou rozděleny do tří úrovní, podle četnosti testování a úrovně automatizace procesu vydávání nových verzí.

Úroveň 1 jsou operační systémy:

- Red Hat,
- Solaris – pouze klient.

Úroveň 2 jsou operační systémy:

- SUSE Enterprise Server,
- Debian/Ubuntu,
- Windows Server 2008 – pouze klient.

Úroveň 3 jsou operační systémy, které jsou podporovány s podmínkou, že některé funkce nemusí pracovat správně a že mohou pracovat pouze jako klient. Tyto systémy jsou:

- IBM AIX,
- HP-UX,
- Open Indiana,
- SmartOS.

Dále je možné si nechat vytvořit instalační balík pro další operační systémy, pokud splňují podmínky pro překlad *Community Edition*. [8]

## 3.3 Princip funkce

Systém CFEngine lze přirovnat k orchestru. Sestává se z více počítačů (hudebníků), kde každý má svoji vlastní kopii not a ví jak je zahrát. Může, ale nemusí mít dirigenta, který pomáhá koordinovat samostatné jedince.

Komponenty CFEngine samostatně běží na každém počítači. Pokud potřebují, mohou spolu komunikovat. Pokud dojde k přerušení síťového připojení, CFEngine komunikaci přeskočí a pokračuje ve své činnosti.

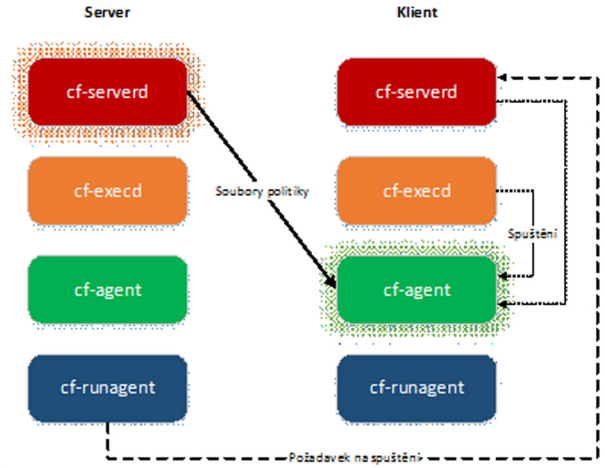
Z pohledu CFEngine není technický rozdíl mezi klientem a serverem. Všechny instalace CFEngine obsahují shodné komponenty a mohou zastávat všechny činnosti[4]. Volba, který z uzlů bude server, zůstává na administrátorovi. Server ale stále zůstává klientem, který používá sám sebe jako svůj server. Ostatní klienti mohou, pokud je to politikou vyžádáno, zastávat funkci serveru.

[2] http://cfengine.com/pub/apt/packages
[3] http://cfengine.com/pub/yum/
[4] Neplatí v plném rozsahu pro Enterprise Edition.

Každá instalace CFEngine obsahuje následující komponenty:

- cf-serverd, démon pracující jako fileserver;
- cf-execd, démon spouštějící v pravidelných intervalech cf-agent;
- cf-agent, agent vykonávající politiku;
- cf-runagent, pomocný program na ruční spuštění cf-agent na jiném počítači.

Schéma komunikace mezi komponentami je znázorněno na Obr. 2. [9]




Obr. 2

Administrátor obvykle uloží politiku na jeden z počítačů a tím z něj vytvoří server. Ostatní počítače jsou nastavené jako klienti a znají jeho IP adresu. Na serveru běží démon cf-serverd, na klientovi pak démon cf-execd. Démon cf-execd spouští v pravidelných intervalech (výchozí 5 min.) nástroj cf-agent, který se připojí k cf-serverd a ze serveru stáhne soubory s politikou. Potom politiku interpretuje. Pokud soubory s politikou nebyly změněny nebo se nelze připojit k serveru, pak cf-agent soubory nestahuje a pouze interpretuje již dříve stažené.

Z toho vyplývají dvě důležité vlastnosti CFEngine:

- Politika je interpretována lokálně bez použití serveru. Server slouží pouze jako úložiště pro soubory politiky. Politika může být na klienta zkopírována jakkoliv jinak a bude fungovat, dokonce i když bude odpojený od sítě. Politika může na klientovi být i vytvořena.
- Klient politiku stahuje dobrovolně metodou pull, nikoliv push. Tím je zaručena vysoká bezpečnost.

Program cf-agent lze vzdáleně spustit nástrojem cf-runagent. Ten kontaktuje cf-serverd na klientovi, který spustí cf-agent mimo nastavený interval. Tak lze simulovat přenos push.

## 3.4 Jazyk

CFEngine pracuje na principu teorie slibů (promise theory). Umožňuje správci definovat požadovaný stav IT systému, který je vyjádřen politikou zapsanou pomocí deklarativního jazyka. Politika (policy) je soubor slibů (promises), které deklarují záměr něco vykonat nebo se chovat daným způsobem.

Cokoliv v CFEngine může být chápáno jako slib. Příkladem slibu může být webserver, který slibuje, že na něm bude nainstalována a poběží služba httpd na TCP portu 80.

Sliby mohou být vztaženy k různým subjektům, jako jsou např. soubory a jejich práva, příkazy, přístupová oprávnění, atd.

Každý slib se skládá z objektu, ke kterému se slib vztahuje (promiser) a z atributů, které musí být pro tento objekt splněny. Každý atribut se skládá z typu atributu a hodnoty. [10]

Příklad slibu:

```
files:

      "/tmp/test_file"
         comment => "Soubor test_file musí existovat a mít
                     práva 644.",
          create => "true",
           perms => m("644");
```

Slib je typu `files:` a definuje, že musí existovat soubor `test_file` v adresáři `/tmp`, který bude mít práva 644. Pokud neexistuje, má být vytvořen. Komentář je součástí slibu a popisuje, co slib vyjadřuje.

Pro každý typ slibu existuje velký počet definovatelných atributů. Seskupení souvisejících atributů do znovu použitelné komponenty, se nazývá **body**.

Slibů CFEngine rozeznává několik typů, podle typu subjektu, ke kterému se slib vztahuje:

- vars:
- classes:
- files:
- commands:
- processes:
- services:
- packages:

- methods:
- reports:

Seskupení souvisejících slibů do pojmenovaného celku se nazývá **bundle**. Např. skupina slibů vztahujících se k webovému serveru může být sdružena do bundle pojmenovaného webserver. [11]

Příklad bundle pro ntp server:

```
bundle agent ntp
   {
      files:

         "/etc/ntp.conf"
             create => "true",
          copy_from => secure_cp("/repo/config-files/ntp.conf",
               "daidalos.civ.zcu.cz");

      services:

         "ntp"
            service_policy => "start";
   }
```

Bundle ntp seskupuje dva sliby související s během *ntp* démona:

1. V sekci slibů typu files: je slib, podle kterého musí existovat konfigurační soubor, který je totožný se souborem uloženým v repositáři na serveru. Pokud tomu tak není, CFEngine soubor stáhne a opraví tím soubor do požadovaného stavu.
2. V sekci slibů typu services: je slib, podle kterého musí být spuštěna služba *ntp*. Pokud tomu tak není, CFEngine službu spustí.

Často používané body a bundle jsou komunitou seskupovány do **standardní knihovny** (COPBL). Cílem je umožnit uživateli psát politiky bez nutnosti vytvářet vlastní body a bundle pro často opakované činnosti. Některá body slouží jako znalostní báze o různých operačních systémech, balíčkovacích systémech apod. Není tudíž nutné zkoumat umístění konkrétních spustitelných souborů v konkrétním OS, vstupy a výstupy různých balíčkovacích systémů apod.

Sliby je možné aplikovat pouze na konkrétní prostředí s pomocí **tříd**. Slib může být aplikován např. pouze na systémy typu Linux, pouze ve zvolený den v týdnu, nebo pokud proměnná nabývá určité hodnoty.

Příklad použití třídy:

```
bundle agent tvrde_tridy
   {
```

```
    reports:

       linux::
          "Tento klient používá Linux!";

       solaris::
          "Tento klient používá Solaris!";

       windows::
          "Tento klient používá Windows!";
}
```

## 3.5 Bezpečnost

Jednou z klíčových vlastností CFEngine 3 je důraz kladený na bezpečnost. Architektura, komunikační protokoly a použitý programovací jazyk C, který nezávisí na externích knihovnách skriptovacích jazyků, dělají CFEngine velice odolným proti potenciálním útokům. [12]

CFEngine 3 nemá žádný záznam v databázi zranitelností NVD amerického národního institutu pro standardizaci a technologie NIST. Pro CFEngine 2 je v databázi evidováno 9 záznamů[5].

Aby byla bezpečnost udržována stále na vysoké úrovni, řídí se vývojáři CFEngine následujícím kodexem:

1. Už v návrhu by mělo být zaručeno, že není možné CFEngine klientovi zaslat data, která by měnila politiku. Každý klient by si měl zachovat právo kdykoliv navrhovanou politiku odmítnout. To se nazývá model dobrovolné spolupráce.
2. CFEngine by měl vždy podporovat šifrování přenášených dat.
3. Každý host by měl pokračovat ve své funkci, tak dlouho jak to bude možné, bez nutnosti komunikace s ostatními uzly.
4. CFEngine by měl používat jednoduchý model důvěryhodnosti na bázi klíčů (podobný SSH). Neměla by být použita žádná centrální autorita. SSL a TLS by nemělo být použito.
5. CFEngine by měl vždy poskytovat bezpečné výchozí hodnoty, které neposkytují přístup k ostatním uzlům.

Splnění bodů kodexu lze velice snadno ověřit:

- Body 1., 3. a 5. jsou splněny díky principu funkce CFEngine. Každý klient stahuje politiku ze serveru dobrovolně, v dobrovolně zvoleném čase. Směr

[5] http://web.nvd.nist.gov/view/vuln/search-results?query=cfengine&search_type=all&cves=on

přenosu politiky je vždy pull, nikoliv push. Politika je vykonávána lokálně bez jakýchkoliv vazeb na server nebo ostatní klienty. Politika je prováděna i v okamžiku, kdy je klient odpojen od sítě.

- Bod 2. je splněn do té míry, že přenášené soubory jsou šifrovány volitelně. V případě Community Edition je používána šifra Blowfish 128, v Enterprise Edition je používán AES 256.
- Bod 4. je splněn použitím ověřováním s pomocí veřejného RSA 2048 klíče. [13]

## 3.6 Vliv na bezpečnost IS

Použití CFEngine má zásadní vliv na zvýšení bezpečnosti informačního systému jako celku. CFEngine pracuje autonomně na pozadí a ve stanoveném intervalu (výchozí 5 min) ověřuje soulad klienta s definovanou politikou. Pokud je u některého slibu zjištěn nesoulad, pak je tento opraven s možností nahlášení události administrátorovi. Taková situace může nastat:

- nevhodným zásahem administrátora,
- činností nainstalovaného software,
- manipulací útočníkem.

V krátkém intervalu je tedy prováděn kompletní a pravidelný bezpečnostní audit všech uzlů systému s okamžitou automatickou nápravou.

CFEngine může pomoci zvýšit bezpečnost IS neustálou kontrolou:

- běžících procesů,
- otevřených portů,
- obsahu a oprávnění souborů,
- nastavení SSH,
- firewallu,
- verzí balíčků,
- atd. [14]

## 3.7 Dokumentace

Dokumentace CFEngine 3.5 je strukturována do tří základních kategorií:

- základní manuál[6],
- referenční manuál[7],
- příklady[8].

[6] https://cfengine.com/docs/3.5/manuals.html
[7] https://cfengine.com/docs/3.5/reference.html
[8] https://cfengine.com/docs/3.5/examples.html

Základní manuál popisuje architekturu, design, komponenty, princip fungování a koncepci jazyka. Slouží k prvotnímu seznámení se s prostředím CFEngine a k pochopení jeho fungování.

Referenční manuál do detailu popisuje jednotlivé prvky jazyka a slouží především jako programátorská příručka pro vytváření politik.

Příklady řeší ukázkové problémy od nejjednoduššího uvedení CFEngine do provozu po konkrétní praktické problémy. Cílem je procvičení použití programovacího jazyka, kde lze čerpat inspiraci pro vlastní řešení.

Nevýhodou dokumentace verze 3.5 je upřednostnění designu před přehledností a účelností. V některých případech je vhodnější použít původní dokumentaci verze 3.0[9], která je přehlednější a ucelenější. Původní dokumentace obsahuje také více příkladů a například popis standardní knihovny existuje pouze v této verzi[10].

# 4 Zkušenosti

## 4.1 Instalace serveru a klientů

První instalace CFEngine 3 Community Edition na ZČU proběhla na Debian GNU/Linux přes apt z balíku cfengine3 ze standardního repositáře. Tato instalace měla ihned z počátku několik nevýhod:

1. Nainstalovaná verze CFEngine 3.2.4 byla zastaralá proti aktuální verzi 3.5.2. Ve verzi 3.4 a především ve verzi 3.5 byly do CFEngine implementovány některé důležité nové funkce např. podpora slibů typu `methods`: (od 3.4).
2. Byla výrazně změněna adresářová struktura. Soubory byly přesunuty z `/var/cfengine` do `/var/lib/cfengine`, adresář `/var/cfengine/inputs` byl změněn na symbolický odkaz na umístění `/etc/cfengine`.
3. Čistá instalace neobsahovala žádné předem připravené politiky, pouze prázdný adresář *masterfiles*. Kritická byla především absence souborů s nastavením serveru, souborů pro aktualizování klientů a absence knihoven.

Na základě těchto nevýhod bylo od používání balíku z repositáře Debianu upuštěno. Pro další testování byl využíván balík `cfengine-community` z oficiálního repositáře[11] projektu CFEngine. Oficiální repositář odpovídá formátu pro apt, je tedy možné jeho adresu vložit do `/etc/apt/sources.list`:

```
# cfengine.com
   deb http://cfengine.com/pub/apt stable main
```

[9] https://cfengine.com/archive/manuals
[10] https://cfengine.com/archive/manuals/CfengineStdLibrary
[11] http://cfengine.com/pub/apt/packages

Instalace nebo aktualizace CFEngine na libovolném serveru s operačním systémem Debian je pak triviální:

```
apt-get install cfengine-community
```

Spárování se serverem a první aktualizace politiky klienta se provádí spuštěním programu cf-agent s parametrem `--bootstrap`:

```
cf-agent --bootstrap --policy-server 192.168.1.8
```

Pokud `cf-agent` detekuje vlastní IP adresu, začne se považovat za CFEngine server. Použití IP adresy lokální smyčky (127.0.0.1) není doporučováno. [15]

## 4.2 Správa verzí politiky

Systém správy verzí politiky není v CFEngine standardně obsažen, ale jeho používání je doporučené. (16) Politika v deklarativním programovacím jazyce je zapsána v plaintextu, může tak být použit některý z běžných verzovacích systémů. V prostředí CIV ZČU byl použit systém správy verzí GIT.

Kromě běžných výhod verzovacích systémů, kterými jsou:

- udržování kompletní historie úprav,
- víceuživatelský přístup,
- vytváření vlastních větví konfigurace,

je využito distribuovaného principu fungování GITu pro lokální vývoj a testování politik.

Základní požadavky na produkční nasazení verzovacího systému na ZČU byly:

1. Použít centrální repositář GITu na AFS.
2. Umožnit transparentní členění CFEngine klientů do různých vývojových větví.
3. Automaticky rozšiřovat změny nahrané do GIT repositáře na CFEngine klienty.
4. Zachovat distribuci souborů na klienty přes CFEngine.
5. Umožnit lokální vývoj a testování politik na libovolném počítači v síti ZČU.

Byly stanoveny dvě základní vývojové větve, do kterých mohou být CFEngine klienti přiřazeni a to:

- *master* – testovací větev,
- *production* – produkční větev.

Klienti jsou rozděleni do větví administrátorem, přičemž:

- vývojová větev *master* by měla obsahovat 1 až 5 testovacích serverů a měla by být použita pro otestování nové politiky,
- vývojová větev *production* by měla obsahovat všechny zbývající produkční servery.

Stanovené počty serverů nejsou kritické, pouze větev *master* by měla být udržována dostatečně velká na to, aby dostatečně pokryla heterogenitu prostředí (různé typy a verze OS, různé aplikační použití, atd.) a zároveň dostatečně malá, aby rozsah případné havárie byl pouze omezený a bylo možné případné způsobené škody opravit manuálně.

Aby bylo možné klienty rozdělovat do vývojových větví a byl automatizován proces vydávání nových verzí politiky, bylo nutné změnit výchozí adresářovou strukturu CFEngine a upravit proces aktualizace politik klientů v souboru update.cf[12].

Stahování jednotlivých větví CFEngine klienty je možné řešit dvěma různými způsoby:

1. Klient si stáhne všechny větve a pak se sám v okamžiku běhu *promises.cf* rozhodne, kterou větev spustit, podle *class*, do které patří.
2. Klient si stáhne pouze soubory konkrétní větve, podle class, do které patří.

Pro prostředí ZČU byl zvolen druhý způsob. Ten je výhodný, protože vlastní běh *promises.cf* s výkonnou politikou je naprosto odstíněn od rozhodování o větvích a je tak naprosto transparentní. Rozhodování o stažení větve je provedeno už při aktualizaci politiky v souboru *update.cf*. Tím se minimalizuje možnost narušení výběru větví uživatelskou chybou v souboru *promises.cf*.

## 4.3 Struktura politiky

Strukturování politiky není v CFEngine nijak striktně dáno ani nijak předem připraveno, jako je tomu u konkurenčních produktů. Struktura tedy závisí zcela na fantazii administrátora a na jeho pochopení fungování CFEngine. V prostředí ZČU byla vytvořena struktura, která umožňuje přehledně přiřazovat služby počítačům nebo jejich skupinám.

Politika je chápána jako service-oriented, tzn. že bundle jsou vytvářeny pro danou službu. Vždy existuje alespoň jeden vstupní bundle pojmenovaný podle služby. Tento bundle může, ale nemusí volat další bundle.

Provoz služby webserver se může skládat z činností:

- instalace balíčku webserveru,
- instalace podpůrných balíčků (např. PHP),

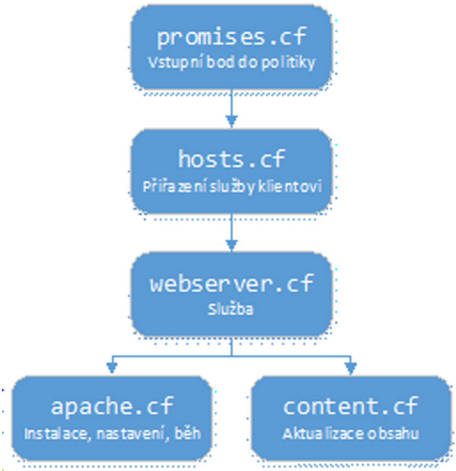
[12] Zde je zejména potřeba dát pozor na soubor validated_updates_ready, který klientům signalizuje změnu v souborech politiky na serveru. Po úpravě pro GIT přestane fungovat a musí být nahrazen vlastním řešením.

- kopírování a úprava nastavení,
- kopírování a úprava obsahu,
- spuštění služby a dohled nad ní.

V tomto případě by tedy byl vytvořen bundle nazvaný webserver a ten by obsahoval sliby pro jednotlivé činnosti. Činnosti by v případě většího rozsahu mohly být odděleny do samostatných bundle.

Dalším krokem je přiřazení služby klientovi nebo klientům. Je vhodné toto přiřazení provádět v jednom souboru např. hosts.cf, namísto použití tříd uvnitř souboru služby. Vše je na jednom místě a elegantně se oddělí konfigurace od implementace. Služba se z tohoto pohledu považuje za černou skříňku.

Strom volání bundle pak bude vypadat jako na Obr. 3.




Obr. 3

Organizace do složek může být potom následující:

```
/promises.cf
/hosts.cf
/services/webserver/webserver.cf
/services/webserver/apache.cf
/services/webserver/content.cf
```

Tato struktura politiky byla shledána optimální pro přehledné vytváření a nasazování politik.

# 5 Závěr

Na ZČU v Plzni byly otestovány jak CFEngine 3 Community Edition v testovacím prostředí na OS Debian, tak CFEngine 3 Enterprise Edition v heterogenním prostředí s OS Debian, Solaris, Windows a Raspbian na Raspberry PI. Testování bylo úspěšné a tak byla verze Community Edition později nasazena do produkčního prostředí s OS Debian. Velkou výhodou je minimální stopa na serverech a minimální HW nároky bez nutnosti instalovat další software. Deklarativní jazyk CFEngine výborně plní svůj účel, ale jeho učící křivka není příliš strmá, vyžaduje trochu praxe. Dokumentace CFEngine podléhá designu a občas je komplikované najít správnou stránku v manuálu. I přes tyto nevýhody je CFEngine excelentním nástrojem pro hromadný konfigurační management.

# Literatura

[1] Tsalolikhin, A.: Automating System Administration with Cfengine 3: An Introduction. *Vertical Sysadmin.* [Online] 21. 12. 2009. [Citace: 4. 1. 2014.] http://www.verticalsysadmin.com/cfengine3/.

[2] CFEngine AS: The History of CFEngine. *CFEngine.* [Online] CFEngine AS. [Citace: 1. 4. 2014.] http://cfengine.com/the-history-of-cfengine.

[3] CFEngine AS: CFEngine Community Edition. *CFEngine.* [Online] CFEngine AS. [Citace: 4. 1. 2014.] http://cfengine.com/community.

[4] CFEngine AS: CFEngine 3 Nova Owner's Manual. *CFEngine Documentation Archive.* [Online] 21. 10. 2011. [Citace: 4. 1. 2014.] https://cfengine.com/manuals_files/NovaOwnersManual.pdf.

[5] CFEngine AS: Which CFEngine Edition is Right for Me? *CFEngine.* [Online] [Citace: 4. 1. 2014.] http://cfengine.com/cfengine-comparison.

[6] Burgess, M, a další: INSTALL. *Github.com.* [Online] 19. 11. 2013. [Citace: 19. 1. 2014.] https://github.com/cfengine/core/blob/master/INSTALL.

[7] CFEngine AS: CFEngine 3 Linux Distributions. *CFEngine.* [Online] CFEngine AS. [Citace: 19. 1. 2014.] http://cfengine.com/cfengine-linux-distros.

[8] CFEngine AS: Supported Platforms and Versions. *CFEngine.* [Online] CFEngine AS. [Citace: 19. 1. 2014.] https://cfengine.com/docs/3.5/getting-started-supported-platforms.html.

[9] CFEngine AS: CFEngine 3 Concept Guide. *CFEngine Documentation Archive.* [Online] CFEngine AS. [Citace: 20. 4. 2014.] https://cfengine.com/archive/manuals/cf3-conceptguide.

[10] CFEngine AS: Language Concepts — Promises. *CFEngine Manuals.* [Online] CFEngine AS. [Citace: 15. 2. 2014.] https://cfengine.com/docs/3.5/manuals-language-concepts-promises.html.

[11] CFEngine AS: Language Concepts — Bundles. *CFEngine Manuals.* [Online] CFEngine AS. [Citace: 15. 2. 2014.] https://cfengine.com/docs/3.5/manuals-language-concepts-bundles.html.

[12] CFEngine AS: CFEngine 3 Security. *CFEngine.* [Online] CFEngine AS. [Citace: 19. 4. 2014.] https://cfengine.com/security.

[13] CFEngine AS: CFEngine Architecture and Security. [Online] 7 2010. [Citace: 19. 4. 2014.] http://cfengine.com/manuals_files/SpecialTopic_Security.pdf.

[14] CFEngine AS: The Impact of Configuration Management on Security in IT Operations. [Online] 17. 4. 2013. [Citace: 19. 4. 2014.] http://info.cfengine.com/rs/cfengineab/images/20130417%20-%20Whitepaper%20-%20The%20Impact%20of%20CM%20on%20Security%20in%20IT%20Operations.pdf.

[15] Zamboni, D.: CFEngine Tip #004: How to Bootstrap a CFEngine Client. *News about "Learning CFEngine 3" book.* [Online] 25. 9. 2012. [Citace: 15. 3. 2014.] http://blog.cf-learn.info/cfengine-tip-004-how-to-bootstrap-a-cfengine/.

[16] CFEngine AS: Version Control. *CFEngine Manuals.* [Online] CFEngine AS. [Citace: 9. 2. 2014.] https://cfengine.com/docs/3.5/manuals-writing-policy-version-control.html.

# Virtualizace síťových prvků

**Martin Pustka**

E-mail: Martin.Pustka@vsb.cz

Poslední léta v IT infrastrukturách byly ve znamení nemalých změn, zejména v oblasti virtualizace serverových systémů. Tato oblast je již poměrně stabilní a tak v poslední době začíná virtualizace pronikat také do oblasti síťové infrastruktury.

Síťová infrastruktura jako celek se nedá virtualizovat do takové míry jako serverové systémy, protože vždy bude její část tvořit fyzickou část samotných datových center. Přesto lze najít aplikace, které je vhodné virtualizovat jako celek nebo jejich části.

## Základní pojmy

Pro potřeby tohoto příspěvku je potřeba si definovat základní pojmy.

**Virtualizační infrastruktury** jsou takové infrastruktury, které tvoří prostředí pro virtualizaci, nejčastěji se jedná o produkty a produkty VMWARE vSphere, KVM, Hyper-V a XEN.

**Virtuální systém** (virtual machine), někdy se také používá termín **virtualizovaný systém**, je obvykle reprezentován virtuálním serverem, virtuální stanicí nebo virtuálním směrovačem, která je provozována ve virtualizační infrastruktuře. **Virtuální síťové prvky** jsou takové virtuální systémy, které ve virtuálních infrastrukturách plní funkci síťových prvků.

**Virtuální infrastruktury** nebo také **virtualizované infrastruktury** jsou takové infrastruktury, které jsou tvořeny virtuálními systémy, tvoří provozní celek a jsou instalovány ve virtualizačních infrastrukturách.

## Jak na škálovatelnou fyzickou síťovou infrastrukturu DC

V datovém centru řešíme na počátku zejména problematiku dostatečných fyzických kapacit, které budou jednoduše a bezvýpadkově rozšiřitelné. Z tohoto pohledu tak není příliš perspektivní stavět středně velká a větší moderní datová

centra na technologiích 1GE, vhodnější je již od počátku plánovat nasazení 10GE technologií. Tato moderní technologie dnes podporuje i konvergovaný ethernet a poskytuje dostatečné síťové pásmo pro provozované aplikace. Výhodou konvergovaných technologií je budování jedné sítě a jednotné správy a jednoduché rozšiřování kapacit sítě, ať už upgradem technologií nebo sdružováním fyzických kanálů.

Pro další rozvoj a růst sítě datového centra je velmi důležité postavit síť dostatečně jednoduše a škálovatelně takovým způsobem, aby mohla být síť dále rozvíjena a to bez výpadku a dopadu na provoz běžících virtuálních systémů.

Virtualizační infrastrukturu je vhodné z pohledu počítačové sítě budovat jako L2 infrastrukturu. Síťová rozhraní virtuálních systémů (servery, směrovače nebo jiné prvky) jsou zapojovány do vybraných virtuálních sítí (VLAN). A tyto virtuální sítě pak můžeme propojovat do různých L3 sítí, které mohou být obsluhovány lokálními nebo i vzdálenými směrovači.

## Virtuální síťové prvky a jejich požadavky na VI

Virtualizační infrastruktury poskytují služby virtuálních serverů, na které je možno instalovat operační systémy postavené na platformě x86. Tato HW platforma je v současné době stále více využívána i pro fyzické síťové prvky a tak výrobci se čím dál více objevují prostor pro virtualizaci těchto síťových prvků do prostředí VMware vSphere, Hyper-V, KVM a XEN. Z pohledu síťových infrastruktur poskytují moderní virtualizační infrastruktury zejména tyto výhody:

- škálovatelnost technických prostředků (CPU, RAM, síťové pásmo)
- stabilita provozu
- snadná údržba a aktualizace
- snadná obměna HW
- flexibilní síťová infrastruktura
- finanční úspory

Mezi nevýhody virtualizace síťových prvků patří:

- neakcelerovaný síťový provoz bez přímého přístupu k hardware
- obvykle menší maximální výkon než který poskytují specializovaná síťová zařízení
- virtuální síťový prvek je méně efektivní než dedikované síťové zařízení
- závislost na dalších prvcích a jejich správné funkci

Obecně lze říci, že požadavky virtualizovaných síťových infrastruktur jsou odlišné od virtualizovaných serverových infrastruktur. Obvykle je menší požadavek na diskové kapacity, diskový IO výkon i RAM, ale převládá požadavek na CPU výkon a zejména pak na kvalitu počítačové sítě z pohledu přenosových zpoždění (latencí), propustnosti a zejména pak flexibility.

Proto je vhodné mít ve virtualizačních infrastrukturách kvalitní správu a podporu počítačové sítě (např. technologické koncepty VM-FEX nebo Cisco Nexus1000V), která funkčně i výkonnostně rozšiřuje základní možnosti nedistribuovaných i distribuovaných přepínačů (switchů) virtualizačních řešení.

## Minimální nutné podmínky pro virtualizaci síťového prvku

V principu lze virtualizovat poměrně dost síťových prvků a funkcí, ale měli bychom se při virtualizací držet selského rozumu a uvažovat o virtualizaci pouze těch prvků a služeb, které splňují níže uvedené minimální podmínky:

1. Virtualizační infrastruktura nesmí záviset na virtuálním síťovém prvku a jeho správné funkci.

2. Správa virtualizační infrastruktury nesmí záviset na virtuálním síťovém prvku a jeho správné funkci.

## Škálovatelnost a stabilita virtualizačních prostředí

Velmi dobrá škálovatelnost virtualizačních prostředí, zejména v oblasti technických parametrů, umožňuje jednoduché posílení přidělených kapacit. Škálovatelnost nám u některých síťových aplikací umožňuje zvyšovat postupně v čase výkon s nárůstem požadavků nebo naopak vyhradit výkon jen v čase, ve kterém musíme vykrýt nárůst požadavků na kapacitu služby. Nároky síťových prvků rostou s růstem síťových infrastruktur (např. s růstem směrovacích tabulek), popř. s růstem užití (objem zpracovávaného provozu nebo počtu uživatelů).

Pokud máme virtualizační i fyzické infrastruktury dobře a redundantně navrženy a postaveny, tak další nezanedbatelnou výhodou moderních virtualizačních infrastruktur je jejich schopnost vyrovnat se s plánovanými i neplánovánými odstávkami fyzických částí infrastruktur. Přepojení fyzických kabeláží, programový upgrade, výměna nebo porucha fyzického HW tak obvykle neznamená odstávku běžících služeb.

U virtualizačních infrastruktur lze s úspěchem využít i různé techniky a módy pro zvýšení dostupnosti (HA), které se umí automatizovaně vyrovnat s výpadky části virtualizačních infrastruktur (např. výpadek fyzického serveru, na kterém aktuálně běží příslušný virtuální systém), tak i s výpadky provozovaných virtuálních systémů. V kombinaci se standartními prostředky vysoké dostupnosti v sítích tak můžeme zajistit velmi kvalitní a stabilní prostředí.

## Údržba a aktualizace virtualizovaných síťových prvků

Nástroje virtualizačních infrastruktur poskytují nástroje pro snapshoty provozovaných systémů. Tyto nástroje jsou vhodné v případech, kdy aktualizujeme software běžícího systému nebo nasazujeme novou verzi systému. V prvé řadě jsou tyto aktualizace nepoměrně snadnější, protože můžeme novou verzi testovat bez potřeby dalšího HW.

V praxi tak lze vytvořit kopii běžícího systému, připravit si změny na této kopii, tedy např. výměna programového vybavení nebo větší konfigurační změny. Poté lze původní systém odstavit nebo odpojit od sítě a spustit aktualizovaný systém. V případě nutnosti lze původní stav velmi snadno obnovit. Můžeme tedy vzdáleně realizovat různé změny – řešit poruchy, aktualizace SW, přidání dalších síťových rozhraní, rekonfigurace IP adres, posílení HW. To je další z výhod, které nám prostředí virtualizační infrastruktury poskytuje – zásahy i změny můžeme realizovat v provozně vhodnou dobu, vzdáleně a to bez nutné osobní přítomnosti administrátora.

## Efekty a úspory

Výše uvedené vlastnosti nám umožňují stavět služby velmi efektivně a škálovatelně z technického i finančního pohledu. Úspory mohou být různého charakteru, zejména jsou to následující:

- rychlejší řešení poruch
- obvykle menší počáteční finanční investice, nekupují se hardwareové prostředky a je možnost postupného doplnění kapacit, výkonu nebo funkcí až v průběhu času
- možnost dočasného nebo postupného navýšování kapacit
- úspory v nerealizovaných redundantních HW prvcích
- nižší OPEXové náklady - úspora na spotřebě elektrické energie a chlazení

# Síťové služby vhodné pro virtualizaci

Pro virtualizaci jsou vhodné takové síťové služby, které nevyžadují zásadní HW akcelerace síťového provozu. Tedy nepředpokládejme, že by nám nějaké výhody poskytovala virtualizace dnešních výkonných HW směrovačů nebo prvků, na jejichž dostupnosti závisí samotná virtualizační infrastruktura. Vhodné naopak může být implementace následujících služeb:

- VPN koncentrátory pro uživatele i site-to-site spojení
- jednoduché směrovače (i s funkcí NAT/PAT)
- bezpečnostní prvky (např. IDS/IPS)
- směrovače nebo firewally virtuálních serverů
- BGP směrovače (RTBHR, route reflectory)
- prvky pro rozkládání provozu mezi virtuální systémy (load-balancery)
- ISATAP směrovače pro IPv6

Z pohledu správců síťových infrastruktur je nebezpečným prvkem jejich nadměrné užívání a tvorba zbytečně složitých, komplikovaných nebo řetězených síťových infrastruktur. Mělo by být snahou administrátorů sítě virtualizovat především takové služby, které stojí na kraji sítě a netvoří jejich jádro. Nesmí se stát, že samotná virtualizační infrastruktura, která je na funkci sítě závislá, bude v podstatě závislá na službě, kterou sama svou funkcí zajišťuje.

Tedy není vhodné virtualizovat takové směrovače, které jsou součástí páteřní sítě, směrovače, jejichž datový tok je příliš velký s ohledem na použitou virtualizační infrastrukturu a její kapacity, a také takové služby, které je vhodné realizovat na dedikovaných aktivních prvcích počítačové sítě provozovaných mimo virtualizační infrastrukturu.

## VPN koncentrátory

U VPN služeb, jejichž jednou z nejvíce náročných částí je výkon CPU při šifrování, můžeme dedikovat dostatečné množství vCPU a paměti. V exponovaných časech bude tato dedikovaná kapacita využita a v méně exponovaných čase bude naopak k dispozici jiným aplikacím. Očekáváme-li v nějakém čase vyšší nárůst požadavků, můžeme tyto vyhrazené kapacity naopak přechodně ještě zvýšit a to v mnoha případech lze i bez výpadku služby.

V podstatě jediným pádným argumentem proti virtualizaci této služby může hovořit velký datový tok, který závisí na míře využití konkrétní implementace. V takovém případě je potřeba zajistit dostatečné síťové kapacity, které by tento tok byly schopny zvládnout.

## Směrovače s funkcí NAT

V mnoha organizacích jsou k dispozici části sítě, které jsou zapojeny mimo běžnou provozní infrastrukturu. Příkladem mohou být např. sítě pro návštěvníky organizace. Pro tyto účely může být vhodné implemetovat směrovač s funkcí překladu IPv4 adres (NAT/PAT), popř. s podporou směrování IPv6. Takto budované infrastruktury vyžadují dostatečné síťové kapacity, které budou schopny garantovat dostatečné volné pásmo nejen pro virtualizovaný směrovač, ale i pro ostatní systémy provozované na příslušném serverovém systému.

Proti virtualizaci výše uvedených služeb mohou hovořit příliš velké datové toky, které vyžadují příliš velkou datovou konektivitu, popř. akcelerované síťové porty.

## BGP směrovače

Často jsou, zejména ve větších sítích, nasazovány směrovače, jejichž hlavním úkolem není směrování provozu, ale podpora směrování. Úkolem těchto síťových prvků je např. výpočet nebo distribuce směrovacích cest mezi ostatní směrovače. Typickou aplikací jsou BGP route reflectory, RTBHR směrovače nebo BGP route servery.

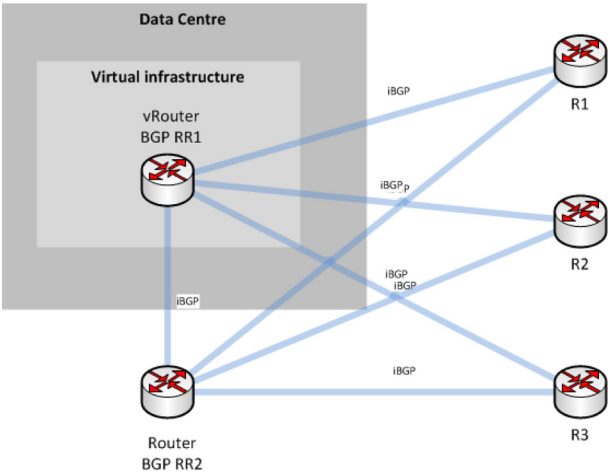
Požadavky na tyto směrovače jsou poměrně jednoznačné – dostatečný výkon CPU a dostatek RAM. Obvykle si tyto směrovače vystačí s jedním fyzickým síťovým rozhraním. V minulosti se pro tyto účely musely v mnoha případech pořizovat výkonné a také drahé směrovače, které disponovaly dostatečným výkonem, ale málo rozhraními.

### BGP route reflectory

Směrování je velmi citlivou věcí a výpadek datového centra by neměl ovlivnit fukčnost celého směrování. Proto je vhodné mít v provozu minimálně jeden fyzický BGP route reflector nezávislý na virtualizační infrastruktuře. Druhou možností je provoz několika BGP route reflectorů, které je vhodné mít v různých datových centrech.

### RTBHR servery

*Router-triggered black hole routing* je techologie, která se používá jako jeden z bezpečnostních mechanismů pro omezení provozu v počítačových sítích s více směrovači. Technologie využívá směrovací protokol BGP. V síti existuje jeden směrovač, který prostřednictvím protokolu BGP šíří na ostatní směrovače o IP adresách nebo rozsazích, které budou v síti blokovány. V obvyklém nasazení této technologie nehrozí v případě výpadku RTBHR serveru omezení samotného provozu.




Obr. 1: Zapojení BGP route reflectorů

### ISATAP router

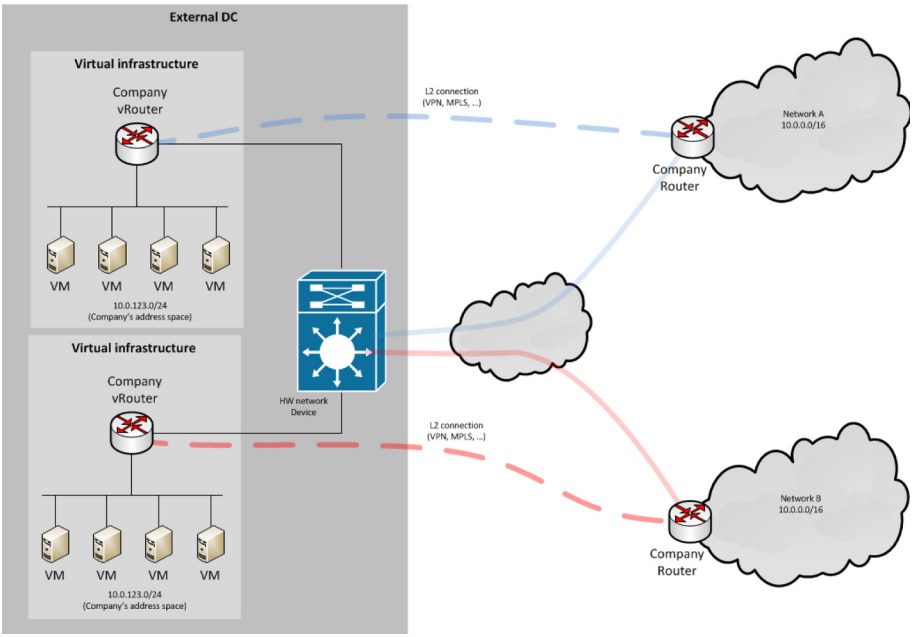
ISATAP (*Intra-Site Automatic Tunnel Addressing Protocol*) je mechanismus, který zpřístupňuje IPv6 konektivitu systémům, které mají nativní pouze IPv4 konektivitu. Koncové systémy posílají IPv6 pakety zabalené v IPv4 paketech dedikovanému systému počítačové sítě. Takto realizované tunelování není stavové. Implementujeme-li IPv6 v počítačové síti, lze prostředky pro přechodové mechanismy, kterým je i ISATAP, provozovat ve virtuální infratruktuře. Provoz ISATAP serveru není příliš náročný na výkon CPU. Podle počtu uživatelů však může být náročný na síťový provoz. V případě dostatečně dimenzovaných síťových připojení se jedná o aplikaci vhodnou pro virtualizaci.

## Síťové propojení virtuálních infrastruktur

V praxi můžeme být postaveni před otázku, jakým způsobem připojit do vlastní sítě virtualizované servery provozované v cizím datacentru nebo cizí virtuální infrastruktuře takovým způsobem, aby byly adresovány z IP rozsahů naší vlastní sítě.

Ukazuje se, že jedním z vhodných řešení může být zprovoznění virtuálního směrovače, který bude propojen do síťové infrastruktury mateřské organizace některou z dostupných VPN technologií – lze využít různé typy VPN, tunelovacích mechanismů, datovým či fyzickým okruhem. Tím získá provozovatel virtualizované infrastruktury plnou kontrolu také nad virtuálním směrovačem, nad síťovou adresací i datovými toky a to nezávisle nad provozovateli virtualizační infrastruktury datového centra.

Taková infrastruktura je pak velmi jednoduše přenositelná do jiného datového centra a pro její zprovoznění stačí přenést existující obrazy virtuálních serverů a směrovačů a zajistit propojení do vnější sítě. Transparentně mohou být zachovány v tomto případě veškeré IP adresy i další nastavení týkající se počítačové sítě.




Obr. 2: Síťové propojení hostované virtuální infrastruktury v externím datovém centru

Výhodou je také lokální propojení mezi virtuálním směrovačem a virtuálními servery v datovém centru. Tak lze snadněji řešit případnou redundanci nebo složitější propojení a v podstatě tak kopírujeme klasické přístupy, které byly vždy aplikovány i ve fyzické infrastruktuře. Síťoví administrátoři tak mají možnost konfigurovat a řídit přístupy k virtualizovaným serverům hostovaným v externích infrastrukturách (např. ACL, firewall pravidla, parametry QoS atd.) i přidělování IP adres.

Propojení z externího datového centra do hlavní sítě provozovatele lze realizovat několika způsoby, které se nebudou lišit od dnešních přístupů používaných u fyzických infrastruktur. Budou tedy záviset zejména na možnostech připojení v samotném datovém centru. Univerzální řešení je tunelování L2 provozu přes veřejnou IP síť nebo různé typy datových okruhů.

# Praktická nasazení virtualizovaných síťových prvků

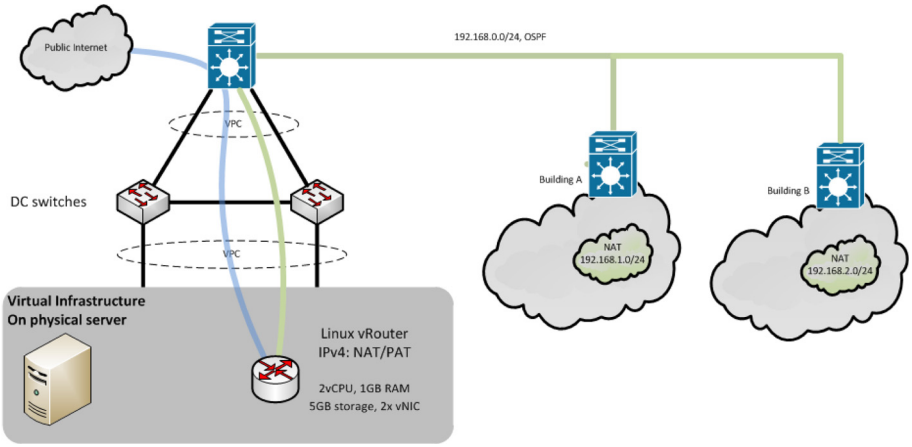
V níže uvedených případech jsou popsány praktické případy nasazení virtualizovaných prvků počítačové sítě v rutinním provozu. Jedná se o případy nasazení v univerzitní síti VŠB-TU Ostrava, která má cca 28 tisíc uživatelů, páteřní síť postavenou na 10GE technologiích, přibližně 1 000 aktivních síťových prvků pevné i bezdrátové počítačové sítě.

## NAT prvek

V univerzitním prostředí VŠB-TU Ostrava jsou provozovány privátní sítě určené pro návštěvníky univerzity. Tyto sítě pro hosty jsou mimo hlavní LAN síť univerzity a jejich lokální směrování je realizováno v oddělené směrovací instanci (VRF) na všech L3 prvcích počítačové sítě.

Všechny lokality jsou vybaveny prvky Cisco Catalyst 6500 ve variantách se supervisory SUP2T, SUP720-3B a SUP32. Tyto L3 přepínače jsou v oddělené směrovací instanci VRF propojeny k nadřazenému virtuálnímu směrovači a své lokální adresní prostory propagují prostřednictvím protokolu OSPF. Virtuální směrovač v protokolu OSPF šíří výchozí cestu (0/0).

Na virtuálním směrovači je realizován překlad adres (NAT/PAT) a provoz je směrován následně do veřejného Internetu.

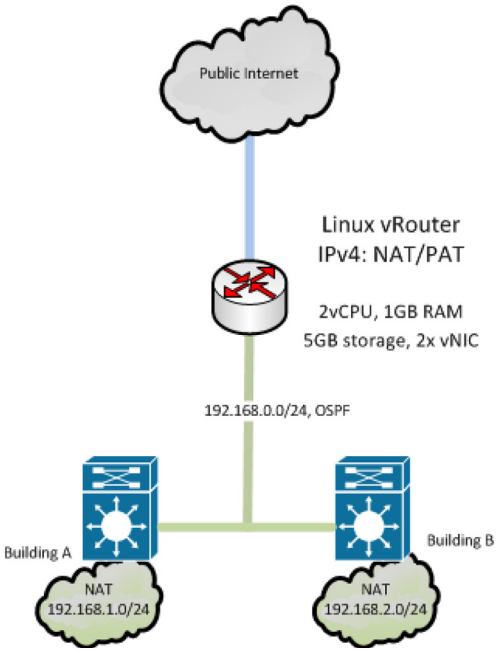



Obr. 3: Technická topologie sítě

Virtuální směrovač není příliš náročný na kapacity virtualizační infrastruktury. Přiděleny jsou dva virtuální procesory, 1 GB RAM, 5 GB na diskovém

úložišti a dvě síťová rozhraní. Na virtuálním směrovači je použita linuxová distribuce GNU/GPL Debian Linux a pro dynamické směrování je použit směrovací démon Quagga také šířený pod licencí GNU/GPL.

Minimální podmínka nezávislosti samotné sítě i nezávislosti virtualizační infrastruktury na virtualizovaném směrovači je zde splněna, tedy službu lze virtualizovat.




Obr. 4: Logická topologie sítě

Provoz virtualizovaného směrovače zajišťuje virtualizační infrastruktura postavená na VMWARE vSphere serverech, která obsahuje celkem 11 fyzických uzlů. Ve výše uvedeném nákresu je situace zjednodušena a vykreslen je pouze jeden uzel.

V rámci virtualizační infrastruktury je provozován distribuovaný přepínač Cisco Nexus1000V. Jednotlivé serverové systémy jsou připojeny do datacentrové sítě dvěma nezávislými 10GE spoji ukončenými na dvou různých přepínačích Cisco Nexus 5548UP s podporou technologie vPC (virtual port channel), čímž je zajištěno redundantní propojení síťové infrastruktury.

Směrovač může generovat informace o probíhajícím provozu (např. technologií NetFlow), kterou lze zpracovávat na Netflow kolektorech a využívat pro analýzu anomálního provozu.

Výhodou systému je jeho velmi stabilní provoz, dostatečná propustnost a také velmi rychlá možnost posílení kapacit provozovaného směrovače v případě nárůstu výkonu. Restart systému trvá necelou jednu minutu, takže i případné nutné aktualizace systému nezpůsobují delší výpadky služby a lze je realizovat v domluvených servisních oknech. V případech aktualizací jsou využívány možnosti virtualizační infrastruktury, před aktualizací je proveden snapshot systému, ke kterému se lze velmi jednoduše vrátit.

V tomto nastavení s výše uvedenými výkony byl bez problémů obsloužen provoz sítě s cca 3 000 nekonkurentně pracujími uživateli a datovým tokem cca 200 Mbps. Do tohoto řešení je zapojena i centralizovaná bezdrátová síť (cca 300 WiFi AP), která je využívána i pro obsluhu univerzitních konferencí a dalších akcí.

## Prvek pro bezpečné IPsec propojení

V univerzitním prostředí VŠB-TU Ostrava je provozován cluster serverů, který přistupuje do několika zabezpečených sítí. Z těchto sítí jsou sbírána data, popř. jsou do těchto sítí data poskytována.

Těchto zabezpečených sítí, se kterými se komunikuje zabezpečeným protokolem IPsec, je více a v čase se mění. Není vhodné instalovat IPsec podporu na všechny systémy serverového clusteru a udržovat všechna nastavení, což je systémově náročnější řešení.

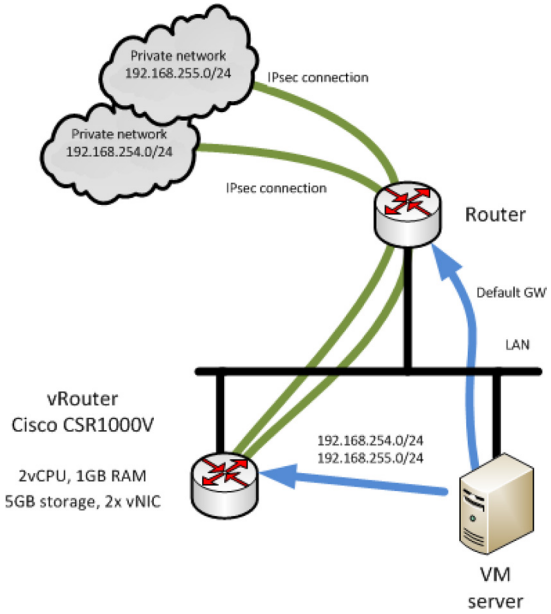
Proto se ukázalo jako praktické řešení instalovat v síti virtuální směrovač Cisco CSR1000V, na který je směrován provoz směrovaný do těchto sítí. Na jednotlivých serverových systémech je jediným systémovým nastavením upravení směrovací tabulky. Ostatní IP provoz pak jde klasickou cestou a je směrován na fyzickém směrovači počítačové sítě.

Veškeré IPsec konfigurace tak jsou uloženy na jednom místě, spravovány jsou síťovými odborníky a odborníci na serverové systémy pak jen tuto službu využívají. Výhodou je také využití stabilního prostředí, protože virtuální směrovač je provozován ve stejném prostředí jako samotné serverové systémy.

Virtuální směrovač je licencován podle objemu přes něj procházejícího provozu. Provoz nad tuto hranici již není směrovačem zpracováván. V případě růstu objemu provozu lze povýšit programovou licenci jejím dokoupením. V tomto případě tak odpadá případná výměna fyzického směrovače, která by byla nutná v případě realizace služby na fyzickém zařízení a která by byla také finančně náročnější.

Tyto limity jsou také důvodem pro to, že prostřednictvím virtuálního směrovače je směrován pouze provoz do zabezpečených sítí a ne veškerý provoz serverového clusteru.

Minimální podmínka nezávislosti samotné sítě i nezávislosti virtualizační infrastruktury na virtualizovaném směrovači je zde splněna, tedy službu lze virtualizovat.




Obr. 5: Směrování IPsec provozu (zobrazen jeden uzel clusteru)

## Remote Triggered Black Hole Routing

RTBHR je technologie využívaná pro blokování provozu vybraných IP adres. Spočívá v distribuci těchto IP adres prostřednictvím směrovacího protokolu BGP ostatním směrovačům počítačové sítě. Tento distribuující směrovač budeme dále nazývat RTBHR směrovač.

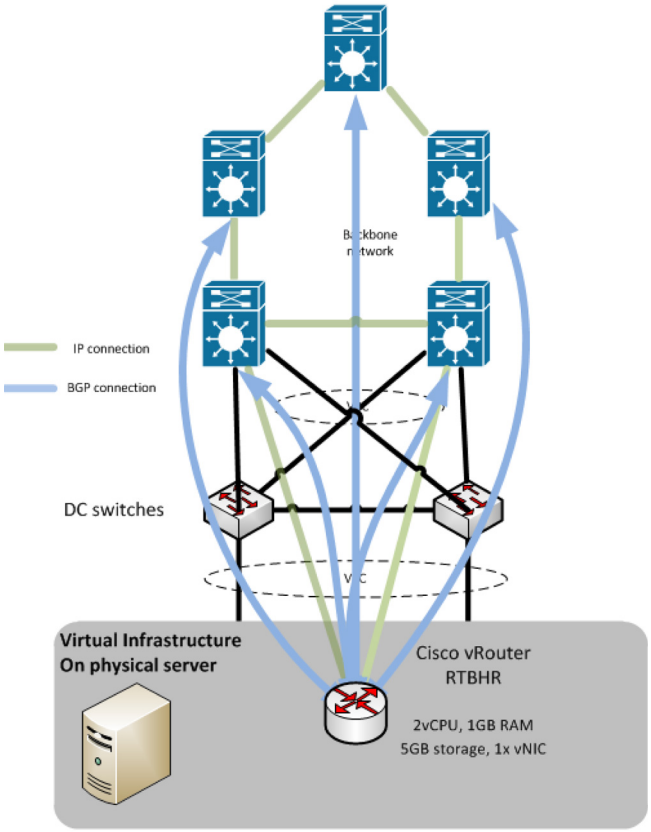
Přes RTBHR směrovač nemusí procházet žádný síťový provoz. Tedy z pohledu vytížení sítě jej můžeme bez obav virtualizovat. Funkčnost počítačové sítě i samotné virtualizační infrastruktury není závislá na funkčnosti RTBHR směrovače, tedy je splněna i druhá podmínka virtualizace síťových prvků.

Tuto techniku pro blokaci IP adres využíváme dlouhodobě v univerzitní síti VŠB-TU Ostrava. V minulosti byla provozována na vyhrazeném směrovači Cisco řady 2500, který i se svým omezeným výkonem pro tuto funkci byl naprosto dostačující.

Protože je vhodné, aby směrovač podporoval i tzv. *route tags*, kterými označujeme sítě, které se mají propagovat v systému RTBHR, tak se jako vhodné řešení se ukázalo použít v síti virtuální směrovač Cisco CSR1000V. Na tento směrovač byla přenesena konfigurace staršího fyzického prvku Cisco řady 2500, který byl pro tyto účely doposud využíván.

Konfigurace je tedy naprosto shodná jako v případě fyzického směrovače. Virtuální směrovač má v rámci počítačové sítě navázána BGP spojení k jednotlivým prvkům počítačové sítě. Propojení do počítačové sítě je realizováno přes dvě nezávislá IP propojení s využitím interního směrovacího protokolu OSPF.

Tento moderní virtualizovaný směrovač podporuje také protokol IPv6, má k dispozici i více paměti i větší procesorový výkon. Mezi další výhody proběhlé virtualizace patří nižší pořizovací i provozní náklady, stabilní prostředí centralizované infrastruktury a také odpadá nutnost rezervovat pro směrovač fyzické místo, napájení a chlazení.




Obr. 6: RTBHR směrovač a BGP spojení

# Provozní doporučení na závěr

- Mějte strategii rozvoje síťové infrastruktury. Ujasněte si, které její části je vhodné virtualizovat, které prvky naopak pro virtualizaci vhodné nejsou.
- Nečekejte zásadní investiční úspory. výhodou bývá lepší flexibilita a škálovatelnost infrastruktury, úspory jsou zejména provozního charakteru (energie, prostor).
- Při virtualizaci dbejte na kompatibilitu s existující a provozovanou síťovou infrastrukturou, ale i s vaší virtualizační infrastrukturou.
- Mějte kvalitní, ale i jednoduchou síťovou infrastrukturu v rámci datového centra. Provoz virtuálních síťových prvků jde obvykle složitější cestou než u fyzických prvků, kde je propojení obvykle realizováno jedním fyzickým kabelem. Proto některé síťové problémy se mohou hůře ladit nebo dohledávat.
- Mějte k dispozici nástroje pro mirroring a monitoring provozu v rámci virtualizační infrastruktury.
- Nevirtualizujte takové části síťové infrastruktury, které jsou nezbytné pro její inicializaci nebo její běh. Typicky nevhodnou aplikací může být například směrování sítě pro správu virtualizační infrastruktury na virtuálním směrovači běžícím v této infrastruktuře.
- Podle stavu fyzické síťové infrastruktury můžete implementovat i takové aplikace, které vyžadují větší pásmo. Ujistěte se, popř. zajistěte, že tento provoz neomezí provoz ostatních systémů ve sdílené virtualizační infrastruktuře.

# Softwarově definované sítě – otevřený networking

**Tomáš Kubica**

E-mail: kubica@hp.com, www.netsvet.cz, Twitter: @tkubica

Na počátku bylo slovo. Toto slovo, instrukce, chcete-li software, definovalo během šesti let sítě. Bude definovat storage, servery, datová centra i celé IT. Proč? Aby byznys, jeho aplikace a služby, byl spravedlivým vládcem IT, řídil dostupnost, náklady, výkon i obchodní model.

## Abstrakce: ti velcí nechť stojí na ramenou svých předchůdců

Může váš byznys být pánem datového centra? Vaše aplikace mají různou důležitost pro obchodní výsledky – to lze v Big Data analýze kvantifikovat. Aplikace a služby se dají měřit o to ne jen klasickými KPI (výkon, latence), ale lze vzít v úvahu záznamy hovorů na lince podpory, tickety, diskuse na sociálních sítích, novinové články i kamerové záznamy (například HP Autonomy IDOL a HAVEn). Na základě těchto vstupů určuje softwarově definované datové centrum (SDDC) infrastrukturu – říká na jakou architekturu aplikaci dát, do kolika uzlů, v jakých zónách dostupnosti, jak řešit ukládání dat a síťové pásmo a potom to všechno vytvoří a nainstaluje. A to živě, takže na základě zmíněných analýz uživatelů a zákazníků svá rozhodnutí pružně za běhu mění.

Takhle složitý mechanismus by nešlo sestavit bez abstrakcí. Ty jsou klíčem k úspěchu a konečně je přenášíme ze světa fyzikálního výzkumu i software na celé IT datového centra. Pokud například budete vytvářet aplikaci, která potřebuje ukládat strukturovaná data, asi nebudete tuto logiku psát – použijete nějakou otevřenou nebo komerční databázi. Pokud chcete zobrazit uživateli tlačítko OK, asi se dnes nepustíte do programování vykreslování pixelu za pixelem po řádkách. To všechno už dávno někdo udělal a perfektně – využijete toho a budete stavět nad tím. Nemusí vás trápit jaká je storage, formát dat, způsob dosažení redundance, operační systém databáze – využíváte abstrakce typu „já ti řeknu jakou chci uložit tabulku a ty to uděláš, ano?“. Teď to konečně dostáváme do samotné architektury datových center.

Byznys logika tak využívá služby SDDC a říká mu například, že aplikace běží špatně v regionu Morava. SDDC požádá další komponenty o součinnost – otevřený cloud dokáže alokovat zdroje, jak ty vlastní, tak půjčené z public cloud, aplikační automatizace nainstaluje nový uzel a infrastrukturní kontroler (postavený na OpenStack, například HP Cloud OS) vytvoří VM, požádá o odloupnutí kousku storage a zařazení do virtuální sítě s novým virtuálním routerem připojujícím ji ke klientům. Ve skutečnosti tento kontroler volá abstrahované služby, které například v případě sítě uvádí v život aplikace na SDN (Software-defined Networking) kontroleru. A ta se zas neobtěžuje s komunikací s prvky, tu pro ni dělá SDN kontroler samotný. Ten pak mluví přímo se zařízením, které abstraktní OpenFlow protokol přetransformuje do nastavení čipů. To je několik vrstev abstrakcí v praxi.

Abstrakce umožňují stavět na ramenou předchůdců a posouvat IT kupředu do oblasti Software-defined Data Center. Do světa, kde byznys, jeho služby a aplikace řídí automatizovaně celé IT. To samozřejmě šetří náklady, ale především umožňuje IT stát se skutečným partnerem byznysu a generátorem peněz.

## OpenStack: snižte náklady a chraňte investice standardizací

Byla doba, kdy server byl svázaný s proprietárním operačním systémem a omezoval rozvoj aplikací a s nimi spojených inovací a nových služeb. Později se světa zmocnily otevřené systémy a standardizace operačních systémů vyústila v bouřlivý aplikační rozvoj. Datové centrum a cloud je stejně jako server uceleným systémem (pravda, trochu větším a složitějším) a tak i ten potřebuje operační systém a příklon k otevřenosti a standardizaci. Tou je projekt OpenStack, automatizační nástroj cloudu a datového centra vyvíjený komunitou s velkým přispěním HP (k dnešnímu dni prsty HP zaměstnanců připsáno 1 300 000 řádků kódu). Jeho produktový otisk, HP Cloud OS (v základní verzi zdarma), je stavebním kamenem HP Converged Cloud použitým jak ve veřejném cloudu (hpcloud.com) tak enterprise nabídce (HP CloudSystem 8). Proč mít standardizovaný OS datového centra?

Moderní automatizační software podporuje různé hypervisory i různé operační modely (z jedné obrazovky řeší vlastní i vypůjčené zdroje) i hardware různých výrobců. To umožňuje jedním operačním modelem využívat KVM pro nenáročné úlohy (vývoj, testování) a současně VMware pro produkční prostředí (a šetřit tak náklady). Stejně tak dovoluje testovat a vyvíjet na vypůjčených zdrojích (třeba z hpcloud.com nebo Amazon) a po dokončení jedním kliknutím převést do vlastního produkčního prostředí. Standardizace a otevřená API tak umožnila soustředit se na přidanou hodnotu místo opakování toho, co už bylo vymyšleno (vracíme se k abstrakcím) – komerční nadstavby tak řeší úlohy jako

je PaaS, SaaS, VPNaaS. A mimochodem virtualizace není nutnou podmínkou. OpenStack Ironic projekt funguje i nad bare-metal platformou jako je HP Moonshot (unikátní serverová technologie s možností až 180 serverů + sítě + storage v 4, 3U) a v rámci projektu OpenStack on OpenStack (tripleO) může tento nástroj instalovat sám sebe. HP Cloud OS, distribuce OpenStack s přidanou hodnotou, tedy začíná nabootováním jeho funkčního zárodku z USB a on pak sám sebe rozšíří tak jak potřebujete. K čemu tohle všechno?

Standardizace systému automatizace datového centra (chcete-li cloud) dovoluje razantně snižovat náklady na operativu a při tom nezamykat ke konkrétním platformám a flexibilně alokovat investiční zdroje. Otevřenost a abstrakce podněcuje další inovace směrem k softwarově definovanému datovému centru.

## Zahoďme okovy železa aneb infrastruktura opravdu virtuální

Odštípnout si kousek storage, rezervovat si kousek trubky, zapojit vybrané servery do virtuální sítě nebo spustit několik serverů v jedné fyzické krabici – to jsou příklady virtualizace prvního typu. Všechny vedou k schopnosti důkladně oddělit různé typy služeb a zajistit jejich kvalitu, sdílet hardware, který tak není potřeba při každé změně fyzicky měnit a lépe využít investic a v neposlední řadě významně zjednodušuje automatizaci IT. Virtualizace druhého typu umožňuje spojit několik zařízení, ať už fyzických či virtuálních, do jednoho virtuálního většího – a to jak v serverech (clustering), sítích (např. HP IRF) i storage (Scale-out). Hlavním důvodem pro nasazení těchto technologií je redundance a jednoduchost a současně zvýšení výkonu cenově efektivním způsobem.

Existuje ještě třetí význam virtualizace a to je schopnost změnit fyzický formát zařízení – například z proprietárního hardware na standardní server nebo do virtuální appliance nasaditelné nad libovolnou infrastrukturou. V oblasti sítí mluvíme o Network Function Virtualization a možnosti například místo fyzického prvku pořídit virtuální (VM), třeba HP Virtual Service Router. Podobně ve storage nejprve docházelo k využití standardních serverů pro storage a později se přidala možnost stáhnout si uzel storage ve formě virtuální VM. Hlavní výhodou této virtualizace je především flexibilita – nový síťový prvek nebo uzel storage nainstalujete doslova jedním kliknutím a to na jakýkoli server.

Virtualizace síťových funkcí (NFV), jako trend bedlivě sledovaný zejména operátory, je v HP reprezentována nedávno oznámeným OpenNFV ekosystémem a kompletní nabídkou od železa, přes software po služby. O dynamické vkládání těchto služeb do infrastruktury se postará softwarově definovaná síť.

Virtualizace sítě a síťových funkcí kombinovaná s plným potenciálem SDN přináší novou ekonomiku do oblasti CAPEX (využití standardního hardware) i OPEX (větší flexibilita a dynamika).

# Standfordské pití čaje: konec sítě rigidní a ovládané z příkazové řádky

Nastartování těchto procesů přišlo v roce 2007 na Standfordské univerzitě za přispění Hewlett-Packard Labs. Rok na to už se datové balíčky prohánělyškolní sítí skrz prvky HP s protypem OpenFlow – dnes jediného standardizovaného protokolu pro to, co se později začalo nazývat Software-defined Networking. Sítě jsou uzavřený svět – s každým nákupem krabičky získává zákazník i proprietární operační systém i robustní sadu stovek aplikací (funkcí, protokolů), které jsou v tomto OS k dispozici. K této skříňce obdržíte manuál jak funguje a jaké parametry můžete, typicky z příkazové řádky, vložit. Jste-li inovátor s dobrým nápadem, nikdo vás dovnitř nepustí – nezbývá, než si vyrobit vlastní switch, OS, všechny funkce a pak slavnostně přidat tu vaší. To není fantazie, ale musíte být Google nebo Facebook. SDN tohle mění - pouští inovace do světa sítí, do světa, který je složitý, ve kterém se síťař i nikdo nechce mluvit, protože používají stále nějaké zkratky a všechno tam řeší zdlouhavě nastavováním přes textové příkazy.

Architektura SDN začíná u silnic, tedy prvků, ať už fyzických nebo virtuálních. HP podpora OpenFlow protokolu je k dispozici i na 5 let starých zařízeních jako softwarový update zdarma. K tomu patří řízení provozu – tím je enterprise laděný komerční HP VAN SDN controller nebo některá z open source variant jako je OpenDayLight. A nad kontrolerem to nejdůležitější – aplikace. Ty si budete moci prohlédnout a stáhnout na HP SDN App Store – kromě drobných aplikací tam budou i řešení našich velkých i malých partnerů a zásadní aplikace HP. Jedna řeší bezpečnost sítě a ochranu před Malware (HP Network Protector) a druhá automatizuje dokonalou kvalitu služby pro vaše Microsoft Lync hovory a telekonference. Další připravované zahrnují dynamické vkládání L4–L7 služeb (NFV), load-balancing linek a aplikací či sjednocení drátových a bezdrátových kontrolerů sítě. Komerční partneři nabízející aplikace v rámci ekosystému jsou Radware, GuardiCore, ECODE, Realstatus nebo BlueCat a připravují se i další. Současně je připravena federace dvou SDN světů skutečnosti a virtuálna spojením VMware NSX multi-hypervisor a HP SDN. V neposlední řadě je k dispozici řada komunitních aplikací včetně těch zdarma a s otevřeným kódem. Pro vývojáře existuje kompletní SDK, dokumentace, certifikační školení, fórum i služby – vyvíjet vlastní aplikace můžete hned dnes (sdndevcenter.hp.com).

Software-defined Networking odstartoval celé hnutí softwarově definovaného IT, protože právě sítě potřebovaly pomoci překonat uzavřený svět proprietárních systémů a otroctví příkazové řádky, která neumožňuje snadno a rychle reagovat na firemní potřeby. Softwarově definované sítě přináší flexibilitu, otevřenost, úspory a poprvé spojují svět aplikací a sítě – ta může konečně vydělávat peníze, ne je jen spotřebovávat.

# Client side DNSSEC validation

**Tomáš Hozza**

E-mail: thozza@redhat.com

**Abstrakt**

*Server software now commonly includes DNSSEC in its implementation, but client side software still lacks extensive DNSSEC support. Many client side applications can benefit from DNSSEC validated DNS responses, for example the validation of SSH fingerprints or TLS/SSL certificates. This document discuss possible problems that validating resolver on client can face. We will describe the current solution architecture used in Fedora Project, its integration with NetworkManager and reveal future plans for even better user experience.*

## 1 Introduction

Domain name system (DNS) serves as a distributed database for storing information. It is usually used to translate domain names to IP addresses. However, it can store other useful data, most notably cryptographic keys or their hashes used by various applications. Just to name a few, DNS records for public keys used by IPsec (IPSECKEY record [9]), records for DNS-based authentication of certificates used by Transport Layer Security (TLS) (TLSA record [6]), and records for verifying Secure Shell (SSH) host using standard SSH key fingerprint (SSHFP record [10]).

Applications that fetch trusted data (e.g. cryptographic keys of fingerprints) from DNS server would benefit from some security mechanism to verify the data authenticity and integrity. Moreover, the classic DNS suffers from several types of attacks, for example Packet Interception, ID Guessing and Query Prediction, and Name Chaining [2]. The DNS Security Extensions (DNSSEC) provide such mechanism and prevent such attacks.

DNSSEC applies asymmetric cryptography to sign data stored in DNS. To validate the DNS response, the hierarchical chain of trust is built from the top of the DNS tree (most commonly root servers) to the bottom (the desired domain

name). It also provides mechanisms for authenticating the non-existence of a domain name. However, it does not provide any data confidentiality service. [1]

Some network-provided DNS forwarders may do the DNSSEC validation for the client. To indicate success they set the Authenticated Data (AD) bit int the response header. Since DNSSEC protects only the resource records in the response, an attacker may intentionally change the AD bit. The client may even decide not to trust network-provided DNS server if the network is not trustworthy. Nowadays clients and workstations are mobile and often migrate between different networks. Two approaches exist to ensure the response data integrity and authenticity [2]:

- Use TSIG [12] (or equivalent mechanism) to secure the communication with the trusted remote server, which will do the validation.
- Do the DNSSEC validation locally.

Using TSIG for every DNS query would introduce high overhead due to establishing and maintaining bilateral secured connection with the remote server.

The second solution is to run validating resolver locally on the client and let the stub resolver [7] use it for all DNS queries. Since it runs locally on the same host, surely nobody will modify the DNS response header, thus there is no need to secure the communication.

This article follows with three sections. We will identify requirements on the solution using local validating resolver. Then we will describe the architecture of current and future solutions used in Fedora Project.

## 2 Requirements

Clients often work in a dynamic environment where network configuration changes. We identified multiple requirements on the solution using local validating resolver to be usable in most use-cases:

- Locally running validating resolver.
- Dynamically reconfigure the local resolver based on network configuration changes.
- Handle split DNS [3] configuration.
- Probe DHCP/VPN-provided nameservers to determine their DNSSEC support.
- Provide a fall-back solution in case DHCP/VPN provided nameservers don't fully support DNSSEC.
- Do a captive portal (hot-spot) detection and handle the log-in.

## 2.1 Local validating resolver

The solution requires a trusted validating resolver. This can be accomplished by running the server locally. It has to support reconfiguration during the runtime without disrupting the service, split DNS configuration by forwarding queries for particular DNS namespace subtree to different servers, and turning off the DNSSEC validation for a DNS subtree or completely, if desired (e.g. due to captive portal log-in).

## 2.2 Dynamic reconfiguration of local resolver

The dynamic environment and changes in network configuration require a mechanism that will reconfigure the local validating resolver properly. The configuration has to be based on the current network configuration and the static user configuration.

## 2.3 Split DNS configuration

The reconfiguration mechanism has to handle the situation when clients connect to a network (usually a VPN) and wants to use the network-provided nameservers only for some specific DNS subtree. Such nameservers may not fully support DNSSEC so the user should be able to turn off the validation for that particular DNS subtree.

## 2.4 DNS servers probing

To save network traffic and root servers load, DHCP/VPN-provided nameservers should be preferred. The solution needs to verify if such servers are able to provide DNS resource records necessary for DNSSEC validation. For the server to support DNSSEC properly, among other things it should support UDP/TCP answers, EDNS0 extension, AD and DO bits, RRSIG, DS, DNSKEY and NSEC/NSEC3 records [5]. If the server does not support DNSSEC, it may not be used for DNS lookups, unless it is the last possibility and user explicitly decided so.

## 2.5 Fall-back configuration

In case the DHCP/VPN-provided nameservers are not DNSSEC capable, the dynamic reconfiguration mechanism should include in its configuration a list of servers that are reachable on the Internet and DNSSEC capable. Such servers can be probed and preferred to root nameservers for resolution. Some networks also filter DNS packets, therefore the fall-back configuration should include nameservers running on different port (e.g. port 80) or event using SSL (e.g. running on port 443).

### 2.6 Captive portal detection

In public networks it is common that the user has to first log in using some captive portal HTTP page before connecting to the Internet. There are several techniques how the captive portal can force the redirection of the user to the desired HTTP page. One of them is redirection using DNS, which is technically cache poisoning attack.

Since DNSSEC would break the captive portal redirection technique, such situation has to be detected in time and handled correctly. One and common way to detect a captive portal is to fetch a well known HTTP page with static content from the Internet and check its content. If the content changed, probably there is a captive portal.

## 3 Solution architecture

In Fedora Project the NetworkManager [8] is the default network connection manager. Since Fedora 17, an end-to-end client side DNSSEC validation solution works with locally running unbound [11] server and dnssec-trigger [4] daemon for dynamic reconfiguration of the unbound server.

The NetworkManager is capable of handling various types of connections including VPN connections. It provides command line tool and also programming API to get the configuration of network connections. NetworkManager includes a dispatcher which runs scripts located at predefined location (in Fedora */etc/NetworkManager/dispatcher.d/*) on any connection state change (e.g. connection going up/down).

Unbound is a validating, recursive and caching DNS resolver developed by NLnet Labs. It also supports reconfiguration during the runtime. This makes it suitable for the client side DNSSEC validation solution.

The dnssec-trigger daemon dynamically reconfigures the unbound server. It communicates with NetworkManager using the dispatcher script. The script forwards the DHCP-provided forwarders to the daemon and the daemon probes the nameservers. If DHCP-provided nameservers are not DNSSEC capable, the trigger will try to contact fallback nameservers from its configuration. If this also fails, it will decide to use root servers. Only the final decision is forwarded to the unbound server. The dnssec-trigger also detects a captive portal by fetching a static HTTP page with known content from the Internet and checking its content. The user is prompted by the dnssec-trigger if necessary (e.g. a captive portal is detected and the user needs to log in, or no DNSSEC-capable nameservers can be reached).

### 3.1 Current situation

Current architecture used in Fedora 20 is illustrated on Figure 1.

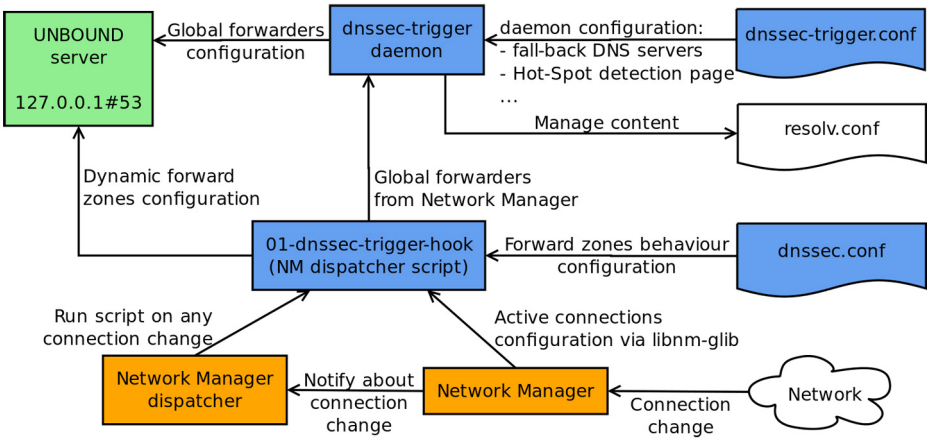



Fig. 1: The current solution architecture in Fedora 20

On any change in the network connections, NetworkManager notifies the dispatcher that then runs all dispatcher scripts. One of those scripts is the *01-dnssec-trigger-hook* script distributed with dnssec-trigger.

The script gets the current network configuration from NetworkManager using the libnm-glib API. It handles two situations:

- It decides which DNS servers should be used as global forwarders. It always takes forwarders provided by the connection marked as *default* by the NetworkManager. Such servers are then forwarded to the dnssec-trigger daemon.

- It handles the split DNS configuration. Based on configuration stored in */etc/dnssec.conf*, current network configuration obtained from NetworkManager and metadata stored on the filesystem it decides if some forward zones for some connection provided domains needs to be configured into unbound or removed.

The user is able to configure the script behavior regarding forward zones configuration. First, the user decides if configured forward zones will be DNSSEC validated or not. This can be set only globally for all forward zones. Second, the user decides if the script should configure forward zones for domains provided by a Wi-Fi connection or not. Note that the script automatically configures forward zones for domains provided by any other type of connection (e.g. wired, VPN).

On start-up the dnssec-trigger reads its configuration from *dnssec-trigger.conf* which includes the URL of site used for captive portal detection and fallback

DNS servers to use when DHCP-provided servers are not DNSSEC capable. Once the dnssec-trigger daemon gets global forwarders that should be used from the dispatcher script, it attempts the captive portal detection and starts the probing of provided forwarders. If these are not DNSSEC capable, it tries to connect nameservers from the configuration running on standard DNS port. If not successful it tries to use root servers and subsequently nameservers running on ports 80 and 443 listed in the configuration.

The dnssec-trigger daemon also manages the */etc/resolv.conf* file content. The file points the stub resolver to the locally running unbound server on address *127.0.0.1*, except the situation when captive portal is detected. In such case, the user is prompted to log in by a web browser. During the log-in, DHCP-provided nameservers are written in the *resolv.conf* file. As soon as the dnssec-trigger detects that the Internet is reachable it will point the stub resolver again to the locally running unbound server.

### 3.2 Future plans

The current solution includes three daemons (unbound, dnssec-trigger and NetworkManager). Since NetworkManager supports DNS plugins, we think a more efficient solution would be to implement DNS plugin for unbound server which will implement all the functionality that dnssec-trigger does at the moment. It will enable us to integrate the user prompting and behavior configuration better into NetworkManager user interface. Furthermore there will be one less daemon running on the system. We will be able to restrict the modification of *recolv.conf* file only to NetworkManager by a system-wide security policy. The draft of future architecture is illustrated on Figure 2.

## 4 Conclusion

DNS is an universal database suitable for storing many types of data, not only IP addresses. The possibility to store trusted data (e.g. cryptographic keys) in the DNS expect a security mechanism to provide data authentication and integrity service. This purpose is solved by DNSSEC.

Many clients and workstations often migrate between different networks, from which some of them may not be trusted. In such cases the end-to-end DNSSEC validation is one option to prevent DNS header modifications and make sure the received data were validated. We listed main requirements for a usable solution using locally running recursive validator. Besides others, the ability to handle dynamic changes in network configuration, split DNS configuration and also captive portal detection are the most important.

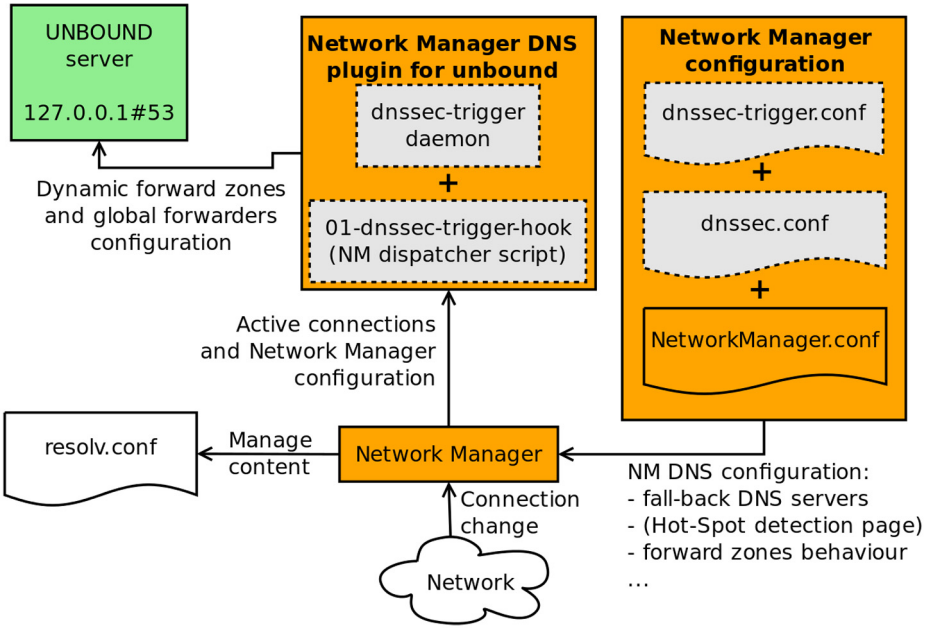



Fig. 2: The planned future architecture (not only) in Fedora

We described a solution using unbound recursive validating server, dnssec-trigger daemon used for dynamic reconfiguration of the unbound server and the NetworkManager for the network connections management. The current architecture is usable, but there is still space for improvements. Also since the end-to-end client side validation solution is not yet widely deployed, there may be some user use-cases which are not covered yet.

Our plan is to implement DNS plugin for unbound into NetworkManager which will perform the functionality currently solved by dnssec-trigger daemon. This way we will be able to reduce the number of daemons needed for the solution and also better incorporate the solution into existing NetworkManager configuration and user interface.

# References

[1] Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: DNS Security Introduction and Requirements, *Request for Comments*, No. 4033, IETF, 2005, http://www.ietf.org/rfc/rfc4033.txt.

[2] Atkins, D., Austein, R.: Threat Analysis of the Domain Name System (DNS), *Request for Comments*, No. 3833, IETF, 2004, http://www.ietf.org/rfc/rfc3833.txt.

[3] Carpenter, B.: Internet Transparency, *Request for Comments*, No. 2775, IETF, 2000, http://www.ietf.org/rfc/rfc2775.txt.

[4] dnssec-trigger project page, http://www.nlnetlabs.nl/projects/dnssec-trigger/, [Cit. 2014–04–17].

[5] Hardaker, W., Gudmundsson, O., Krishnaswamy, S.: DNSSEC Roadblock Avoidance, *Request for Comments*, IETF, 2014, http://tools.ietf.org/id/draft-ietf-dnsop-dnssec-roadblock-avoidance-00.txt.

[6] Hoffman, P., Schlyter, J.: The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA, *Request for Comments*, No. 6698, IETF, 2012, http://www.ietf.org/rfc/rfc6698.txt.

[7] Mockapetris, P. V.: Domain names — concepts and facilities, *Request for Comments*, No. 1034, IETF, 1987, http://www.ietf.org/rfc/rfc1034.txt.

[8] NetworkManager project page, http://wiki.gnome.org/Projects/NetworkManager, [Cit. 2014–04–17].

[9] Richardson, M.: A Method for Storing IPsec Keying Material in DNS, *Request for Comments*, No. 4025, IETF, 2005, http://www.ietf.org/rfc/rfc4025.txt.

[10] Schlyter, J., Griffin, W.: Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints, *Request for Comments*, No. 4255, IETF, 2006, http://www.ietf.org/rfc/rfc4255.txt.

[11] unbound server project page, http://unbound.nlnetlabs.nl/, [Cit. 2014–04–17].

[12] Vixie, P., Gudmundsson, O., Eastlake, D., Wellington, B.: Secret Key Transaction Authentication for DNS (TSIG), *Request for Comments*, No. 2845, IETF, 2000, http://www.ietf.org/rfc/rfc2845.txt.

# DNS a DNSSEC – plně distribuované řešení

**Petr Špaček**

E-mail: pspacek@redhat.com

**Abstrakt**

*Domain Name System (DNS) je distribuovaný systém s prvky částečné centralizace v rámci jedné DNS zóny. Tato centralizace v komplikuje správu DNS zóny a zotavení po havárii. Článek představuje návrh distribuované architektury, který odstraňuje toto historické omezení při zachování kompatibility se standardním DNS a DNSSEC. Navržená architektura umožňuje systému sestávajícímu z $N$ serverů zachovat plnou funkčnost (včetně modifikace dat) i při výpadku $N-1$ serverů. Podpora pro DNS již byla implementována projekty bind-dyndb-ldap a FreeIPA a koncept byl nasazen do reálného provozu. Na implementaci podpory pro DNSSEC se právě pracuje.*

## 1 Úvod

DNS standardizovaný v [8] představuje distribuovaný systém skládající se z administrativních zón. Každá zóna je samostatná jednotka s centralizovanou architekturou. Tato centralizace v rámci zóny sebou přináší jistá omezení projevující se zejména při administraci zóny.

Tento článek popisuje jednu z možností, jak lze v rámci jedné DNS zóny vytvořit distribuovanou architekturu složenou z $N$ DNS(SEC) serverů odolnou proti výpadku až $N-1$ serverů bez nutnosti manuálního zásahu. Cílem je zachování kompatibility se standardním DNS(SEC) protokolem a zároveň odstranění nevýhod plynoucích z centralizované architektury.

Předpokládáme, že distribuovaná architektura by mohla zjednodušit nasazení a administraci DNSSEC technologie v interních (např. podnikových) sítích. Očekáváme, že spolu s nasazením DNSSEC se rozšíří i nasazení příbuzných technologií, které mají potenciál zabránit některým síťovým útokům.

Již jsou standardizovány metody, jak použít DNSSEC pro distribuci veřejných klíčů pro SSH [11], IPSec [10] a také TLS [5]. Další standardy jsou v současné době v přípravě, např. standard pro distribuci veřejných klíčů používaných pro šifrování e-mailu [15].

# 2 Současný způsob nasazení DNS a DNSSEC

Tato kapitola stručně shrnuje obvyklý způsob nasazení DNS a DNSSEC a související provozní postupy. Předpokládáme, že čtenář je již obeznámen s technologiemi DNS a DNSSEC.

Ve standardním DNS každá zóna obsahuje právě jeden tzv. primární master server [13, section 1], který je autoritativním zdrojem dat pro danou zónu, a tudíž je to jediný server, na kterém lze provádět změny v datech dané zóny. Kopie dat (pouze pro čtení) mohou být dále distribuovány na tzv. slave servery.

Existují dvě základní možnosti, jak danou zónu zabezpečit (podepsat) pomocí DNSSEC:

- Master server (nebo jeden jiný server) podepisuje zónu a na slave servery jsou distribuována již podepsaná data.
- Slave servery podepisují data přijatá od master serveru. Je nutné zajistit bezpečnou distribuci klíčů a také zónových dat, aby nedošlo k jejich podvržení ještě před podepsáním.

## 2.1 Periodická aktualizace podpisů

Kryptografické podpisy (RRSIG záznamy) mají omezenou platnost [2, section 3.1.5]. Doba platnosti podpisů závisí na lokální politice nastavené správcem DNS zóny a může se pohybovat ve velkém rozsahu [6, section 4.4.2].

Validace dat selže, pokud je aktuální čas mimo interval platnosti podpisu. Doba platnosti podpisu také určuje interval, ve kterém je možné úspěšně provádět replay útoky [6, section 4.4.2]. Z toho důvodu není žádoucí používat extrémně dlouhou dobu platnosti. Tato skutečnost klade zvýšené nároky na administrátory DNS zóny (při srovnání s nasazením DNS bez DNSSEC).

**BIND verze 9.9.x** DNS server BIND verze 9.9.0 poskytuje nástroje pro jednorázové manuální podepsání DNS zóny (utilitu `dnssec-signzone`). Zároveň podporuje podepisování DNS zóny přímo DNS serverem, což umožňuje serveru přijímat DNS update [13] a generovat podpisy podle potřeby.

Omezením této implementace je, že i poslední dosud vydaná verze 9.9.5 nepodporuje správu klíčů. Tj. BIND je schopen DNS zónu podepsat pouze klíči dodanými uživatelem. Periodická výměna klíčů a generování klíčů pro nové zóny musí být zajištěny nějakým jiným mechanismem.

## 2.2 Periodická výměna podepisovacích klíčů

Publikace [3, section 11.2] doporučuje výměnu Zone Signing Keys (ZSK) každé 1–3 měsíce a výměnu Key Signing Keys (KSK) každé 1–2 roky. Důvody jsou popsány tamtéž.

**OpenDNSSEC** Projekt OpenDNSSEC se skládá z nástroje na podepisování DNS zón (tzv. *signer*) a nástroje pro správu klíčů (tzv. *enforcer*).

Enforcer automatizuje správu klíčů. Klíče jsou generovány a odstraňovány podle nastavené politiky. Úpravy dotýkající se KSK vyžadují manuální zásah administrátora, protože změna musí být koordinována s nadřazenou zónou.

Signer zajišťuje získání zónových dat (čtením ze souboru nebo zone transferem), podepsání DNS zóny dodanými klíči a dále distribuci podepsané zóny (uložení do souboru, zone transfer).

Tyto komponenty jsou relativně nezávislé a lze je využít i samostatně. Např. použít samostatný enforcer pro správu klíčů a podepisování provádět až pomocí DNS serveru BIND.

### 2.3 Aktualizace klíčů v nadřazené zóně

Jedním z problémů spojených s výměnou KSK je aktualizace Delegation Signer (DS) záznamů v nadřazené zóně. DS záznamy v nadřazené zóně je nutné aktualizovat při provádění změn dotýkajících se KSK v podřízené zóně [6]. Aktualizací se zajišťuje, že řetězec důvěry mezi nadřízenou a podřízenou zónou je stále platný. V současné době není tento proces standardizován, existuje zatím pouze návrh [7].

## 3 Identifikované nedostatky

Nasazení DNSSEC a implementace automatické správy klíčů pomocí BINDu a OpenDNSSEC vyžaduje manuální konfiguraci pro nastavení obou nástrojů. Další manuální konfigurace je nutná pro přidání či odebrání DNS zóny zabezpečené pomocí DNSSEC.

Druhé omezení je dáno centralizovanou architekturou: Master server (a případně server provádějící podepisování zóny) vyžadují speciální zacházení. V případě ztráty centrálního serveru (master nebo podepisovač) není možno provádět změny v zóně, a tedy není možné generovat nové podpisy.

Tento problém musí být odstraněn dříve, než vyprší platnost existujících podpisů. V opačném případě nebude možné validovat podpisy dat v zóně a dojde k rozpadu řetězce důvěry mezi nadřízenou a podřízenými zónami [6, section 2]. Celá postižená doména a všechny její subdomény nebudou dostupné (pro klienty provádějící validaci).

Ve standardní centralizované architektuře je nutný manuální zásah, kterým nakonfigurujeme nový master server, obnovíme šifrovací klíče ze zálohy a případně rekonfigurujeme slave servery, aby načítaly data z nového master serveru.

Maximální doba bez aktualizace podpisů, po kterou je zóna funkční, je dána nakonfigurovanou platností podpisů. Tuto dobu lze prodlužovat až do řádu let,

ale tím se zároveň prodlužuje doba, po kterou je možné úspěšně provádět replay útoky [6, section 4.4.2].

# 4 Návrh distribuované architektury

Cílem je odstranit omezení klasického způsobu nasazení DNS a DNSSEC s centralizovanou architekturou.

## 4.1 Odstranění primárního master serveru

Prvním krokem je odstranění konceptu master serveru a zavedení distribuované architektury i v rámci jedné zóny. Chceme vytvořit architekturu, která umožňuje provádět změny dat na kterémkoliv DNS serveru a tyto změny replikovat na všechny ostatní DNS servery.

Jednou z možností je uložit DNS zónu do databáze, která podporuje obousměrnou synchronizaci mezi více servery. V takovém případě DNS server zpracovávající DNS update operaci pouze zapíše data do lokální databáze. Databázový systém potom zajistí distribuci dat na ostatní servery.

### 4.1.1 SOA master name

DNS standard [9, section 3.3.13] zavedl do Start-of-Authority (SOA) záznamu pole `MNAME`, které obsahuje jméno primárního master serveru. Toto jméno může být využito DNS update mechanismem pro výběr serveru, na který bude odeslán update požadavek [13, section 4].

V distribuované architektuře již není možné označit jeden server jako primární master. Proto každý server vyplní pole `MNAME` svým vlastním jménem. Tím vznikne situace, kdy různí klienti vidí různá jména master serverů. Díky tomu DNS update požadavky mohou být distribuovány mezi více serverů.

### 4.1.2 SOA serial

Každá DNS zóna obsahuje v SOA záznamu sériové číslo [4, section 1]. Fakticky se jedná o číslo verze dat v dané zóně, které se monotonně zvyšuje po každé změně v datech zóny.

Z pohledu klienta není tento údaj významný, je využíván zejména pro zone transfer mezi DNS servery. Před každým zone transferem se slave server dotazuje master serveru na hodnotu `SERIAL`. Zone transfer je zahájen, pouze pokud verze dat na slave serveru je nižší než verze dat na master serveru [8, section 4.3.5].

V plně distribuované architektuře je nepraktické udržovat jednu globální hodnotu sdílenou všemi servery. Globální hodnota by vynucovala nákladnou syn-

chronizaci mezi všemi servery. Dalším problémem je aktualizace hodnoty v případě, kdy je jeden či více serverů nedostupných.

Implementaci lze zjednodušit zavedením `SOA SERIAL` hodnoty s lokální platností pro každý master server. Každý master server pak bude aktualizovat pouze vlastní hodnotu a není třeba řešit synchronizaci mezi servery.

Nevýhodou tohoto přístupu je, že různé master servery budou prezentovat stejná data pod jiným číslem verze. Tento stav může způsobovat problémy, když jeden slave server provádí zone transfer z více master serverů. Může dojít např. k následující situaci:

1. Master server $M_1$ publikuje zónu se `SERIAL` = 11
2. Master server $M_2$ publikuje zónu se `SERIAL` = 10
3. Slave server $S$ provede zone transfer ze serveru $M_1$
4. Proběhne aktualizace zóny a master servery si aktualizují svůj `SERIAL`
5. $M_1$ publikuje zónu se `SERIAL` = 12
6. $M_2$ publikuje zónu se `SERIAL` = 11
7. Pokus o zone transfer z $M_2$ na $S$ selže protože `SERIAL` publikovaný $M_2$ není vyšší než `SERIAL` uložený na slave serveru

Tento problém lze částečně zmírnit tak, že po každé změně v zóně se `SERIAL` nastaví na hodnotu tzv. Unix time (počet sekund od `1970-01-01T00:00:00Z`). Pokud v rámci jedné sekundy dojde k několika změnám v zóně, musí být změny sloučeny do jedné transakce. Alternativně lze aktuální hodnotu Unix time inkrementovat pro každou další změnu o 1.

Tímto algoritmem je zajištěno, že hodnoty `SOA SERIAL` na různých serverech budou velmi blízké. Jednodušší varianta s vícenásobnou inkrementací Unix time umožní slave serverům přepnout se z jednoho master serveru na jiný, pouze pokud budou splněny následující podmínky:

- Interval mezi aktualizacemi zóny $>$ 1 sekunda
- Interval mezi aktualizacemi zóny $<$ maximální přípustné zpoždění zone transferu

Poznámka: Tento problém nastává, pouze pokud je použit standardní mechanismus pro zone transfer. Fakticky se jedná o zone transfer za hranice distribuovaného systému popisovaného v tomto článku. Distribuce dat v rámci popsaného systému je řešena mechanismem specifickým pro použitou databázi a může být zcela nezávislá na hodnotě `SOA SERIAL`.

Popsaný problém je obzvlášť závažný při použití inkrementálních zone transferů. V takovém případě musí být slave servery nakonfigurovány ke stahování dat pouze z jednoho master serveru.

### 4.1.3 Konflikty při zápisu dat

V distribuované architektuře lze provádět změny dat na více serverech současně. Systém proto musí být připraven na řešení konfliktů, které mohou při současném zápisu dat nastat.

Konkrétní řešení je závislé na použitém databázovém systému a způsobu uložení dat v něm. Databáze může např. řešit „jednoduché“ konflikty automaticky a ostatní případy ponechat na obsluze systému.

Předpokládáme, že v reálném nasazení by konflikty měly být velmi řídkým jevem. Předpoklady:

- Data DNS zóny jsou uložena v jednotkách odpovídacích jedné DNS doméně (jménu) nebo menších.
- Paralelní úpravy dat nejsou v konfliktu, pokud se změny dotýkají různých databázových jednotek (DNS domén, jmen).
- Administrátoři zóny koordinují prováděné změny, aby současně neprováděli změny v jedné doméně.
- Klienti jsou nakonfigurováni, aby preferovali jeden DNS server před ostatními. Očekáváme, že díky metodě generování `SOA MNAME` popsané dříve budou DNS update požadavky podle [13] odeslány na tento preferovaný server. To by mělo zabránit současné aktualizaci dat na více serverech.

## 4.2 Distribuované podepisování zóny

Decentralizací podepisování dat zóny se zmírňuje problém s aktualizací podpisů popsaný v části 2.1. Plně distribuovaný systém s $N$ servery (kde každý server provádí podepisování nezávisle na ostatních) zůstane funkční i při výpadku $N-1$ serverů.

Nevýhodou toho řešení je nutnost bezpečně distribuovat podepisovací klíče na všechny servery a udržovat sadu klíčů na serverech synchronizovanou. Předpokládáme, že pro manipulaci s klíči lze využít standardní mechanismy jako např. PKCS#11 rozhraní a hardware security module (HSM). Detaily řešení pro distribuci klíčů jsou mimo rámec tohoto článku.

## 4.3 Distribuovaná správa klíčů

Jak již bylo řečeno v sekci 2.2, doporučuje se klíče periodicky obměňovat.

Platnost podepisovací klíčů publikovaných v DNSKEY záznamech je dle [1, section 5] dána platností RRSIG záznamů spojených s DNSKEY záznamy. Vztahuje se na ně tedy problém aktualizace podpisů popsaný v sekci 2.1. Výpadek

v procesu generování klíčů tedy není pro funkčnost zóny kritický, dokud jsou aktualizovány příslušné RRSIG záznamy.

Na druhou stranu, výměna podepisovacích klíčů je netriviální operace, která musí být pečlivě řízena, aby nedošlo k narušení řetězce důvěry mezi nadřízenou a podřízenými zónami. Proces výměny klíčů musí zohledňovat i data uložená v cache klientů (tj. historický stav zóny) [6, section 4.1]. V případě změny KSK je navíc nutné zohlednit DS záznamy v nadřízené zóně.

Z tohoto důvodu je nutné uchovávat i historii stavu klíčů v zóně. Zálohu je nutné provést minimálně v těchto případech:

- Přidání nebo odebrání DNSKEY záznamů
- Změna TTL u DNSKEY záznamů
- Změna TTL nebo doby platnosti RRSIG záznamů
- (Pouze při změně KSK) Přidání nebo odebrání DS záznamů
- (Pouze při změně KSK) Změna TTL nebo doby platnosti DS záznamů

V závislosti na počtu zón a frekvenci změn v klíčích může být periodické zálohování dat nepraktické, protože by perioda zálohování byla příliš krátká.

V takovém případě distribuovaný systém zjednodušuje zotavení po havárii tím, že udržuje více kopií klíčů a metadat o klíčích, takže i v případě ztráty $N - 1$ serverů může proces bez přerušení pokračovat.

Distribuovanost zároveň zjednodušuje plánování a administraci systému, protože odstraňuje server se speciální „rolí“. Tím se zároveň zjednodušují pracovní postupy související se správou systému a snižuje se pravděpodobnost lidské chyby.

Nevýhodou distribuovaného přístupu je nutnost zabezpečit více serverů. Předpokládáme, že při použití síťového HSM tato nevýhoda nebude zásadní.

**Konflikty při zápisu** Distribuovaný algoritmus pro správu klíčů se musí vyrovnat se situací, kdy dočasně nelze komunikovat s ostatními servery.

Je nutné zohlednit, že distribuovaný systém nemá žádný globální zámek, který by bylo možné použít pro vzájemné vyloučení. Může např. dojít k situaci, kdy komunikace mezi servery není funkční, ale servery samotné pracují. Výsledkem bude, že se např. dva servery budou domnívat, že právě protistrana není funkční a je třeba zajistit vygenerování nových klíčů.

Jeden z možných algoritmů je popsán v [12]:

1. Je stanovena perioda, ve které má být klíč generován (před začátkem jeho platnosti, např. 1 měsíc)

2. Každý server náhodně zvolí okamžik v rámci této periody, kdy se pokusí o generování klíče

3. Pokud v daném okamžiku klíč v databázi již existuje, server přijme existující klíč

4. Pokud klíč v databázi neexistuje, server jej vygeneruje a uloží do databáze

5. Pokud dojde ke kolizi, je v zóně ponechán pouze klíč, který byl vygenerován nejdříve

Při řešení konfliktů je nutné držet se zásad pro manipulaci s klíči v zóně uvedených v [6]. Velmi důležité je pravidlo, že manipulace s klíči musí zohledňovat i data uložená v cache DNS klientů.

Ve skutečnosti tedy algoritmus nemůže okamžitě vymazat přebytečné klíče. Je nutné zahájit standardní proces odstranění klíče a klíč dočasně ponechat v zóně. Za tímto účelem je nutné uchovávat některá metadata o klíčích zvlášť pro každý server. Jde např. o časy zveřejnění klíče v zóně, odstranění posledního podpisu vytvořeného daným klíčem apod.

# 5 Implementace

Jednotlivé části popsané architektury jsou postupně implementovány v rámci open-source projektů *bind-dyndb-ldap*[1] a *FreeIPA*[2].

Implementace je založena na DNS serveru BIND 9 a LDAP databázi s podporou multi-master replikace. Všechny DNS servery sdílí jednu logickou databázi, která je v reálném čase replikována na všechny stroje. Konflikty při současném zápisu dat na více serverů jsou řešeny automaticky na úrovni LDAP databáze.

Podpora DNS update [13, 14] je řešena pomocí úpravy pole `SOA MNAME` na každém serveru, jak bylo popsáno v části 4.1.1. Zároveň každý server udržuje vlastní `SOA SERIAL`, což umožňuje provádět zone transfer směrem na standardní DNS servery. Současná implementace odpovídá jednodušší variantě (popsané v části 4.1.2). DNS update operace v rámci jedné sekundy nejsou slučovány, a `SOA SERIAL` se tedy může zvýšit i několikrát za sekundu.

Empiricky jsme zjistili, že konflikty při zápisu dat (diskutované v části 4.1.3) prakticky nenastávají. DNS update je podporován od roku 2009 a dosud se nám nepodařilo zaznamenat jediný případ, kdy by standardní DNS update vedl ke konfliktu při zápisu. Bohužel je u open-source projektů velmi těžké odhadovat velikost uživatelské základny a způsob nasazení, takže nemůžeme toto pozorování podložit konkrétními daty.

[1] Dynamic LDAP database for BIND: http://fedorahosted.org/bind-dyndb-ldap/

[2] http://www.freeipa.org

Projekt FreeIPA zároveň využívá faktu, že data DNS zón jsou uložena v objektové databázi. Použitá databáze ve srovnání se standardním zónovým souborem [9, section 5] nabízí možnost velmi granulárního řízení přístupu a usnadňuje implementaci uživatelského rozhraní pro správu zóny.

**DNSSEC** V současné době[3] se připravuje podpora pro distribuované podepisování dat s využitím tzv. *in-line signing* funkce z BIND 9.9. Distribuovaná správa klíčů bude implementována integrací s upravenou komponentou *Enforcer* z projektu OpenDNSSEC.

Veškerá konfigurace DNSSEC bude integrována do uživatelského rozhraní FreeIPA, což umožní zapnout DNSSEC pro danou zónu jednoduchou změnou jednoho konfiguračního parametru. FreeIPA server poté automaticky vygeneruje potřebné klíče, podepíše zónu a bude zajišťovat aktualizace podpisů a výměnu šifrovacích klíčů. Po standardizaci [7] bude nutná administrativa omezena pouze na jednorázové nahrání DS záznamů do nadřízené zóny.

Předpokládáme, že projekt FreeIPA bude automatizovat zveřejňování veřejných klíčů pomocí DNSSEC. Zejména se jedná o integraci nástrojů pro správu strojů s [11] a správy TLS certifikátů s [5]. Např. stroj přihlášený do tzv. FreeIPA domény automaticky publikuje své SSH klíče do DNS. Zároveň by v DNS mohly být automaticky zveřejňovány TLS certifikáty vydané FreeIPA certifikační autoritou.

# 6 Závěr

Navržená distribuovaná architektura pro DNS zóny umožňuje vytvořit systém s $N$ servery odolný proti selhání až $N - 1$ strojů. Popsaný systém je přitom schopen poskytovat všechny funkce včetně aktualizace dat, generování DNSSEC podpisů a výměny DNSSEC klíčů i v případě, že je funkční pouze jediný DNS server.

Tento koncept byl pro DNS bez DNSSEC implementován v rámci open-source projektů bind-dyndb-ldap a FreeIPA. Empiricky jsme zjistili, že za podmínek uvedených v části 4.1.3 chování systému odpovídá předpokladům. Systém se tedy skutečně vyrovná se ztrátou až $N - 1$ serverů při zachování plné funkcionality.

DNSSEC části konceptu jsou právě implementovány v projektech bind-dyndb-ldap a FreeIPA, ale zatím nebyly prakticky ověřeny. Předpokládáme, že se díky tomuto konceptu podaří implementovat odolný systém s plnou podporou DNSSEC, což by v důsledku mohlo pomoci rozšíření DNSSEC i do interních (podnikových) sítí.

[3] Duben 2014

# Literatura

[1] Arends, R., Austein, R., Larson, M., aj.: Protocol Modifications for the DNS Security Extensions. RFC 4035, Březen 2005. http://www.ietf.org/rfc/rfc4035.txt

[2] Arends, R., Austein, R., Larson, M., aj.: Resource Records for the DNS Security Extensions. RFC 4034, Březen 2005. http://www.ietf.org/rfc/rfc4034.txt

[3] Chandramouli, R., Rose, S.: Secure Domain Name System (DNS) Deployment Guide. NIST Special Publication 800-81-2, Září 2013. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-81-2.pdf

[4] Elz, R., Bush, R.: Serial Number Arithmetic. RFC 1982, Srpen 1996. http://www.ietf.org/rfc/rfc1982.txt

[5] Hoffman, P., Schlyter, J.: The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, Srpen 2012. http://www.ietf.org/rfc/rfc6698.txt

[6] Kolkman, O., Mekking, W., Gieben, R.: DNSSEC Operational Practices, Version 2. RFC 6781, Prosinec 2012. http://www.ietf.org/rfc/rfc6781.txt

[7] Kumari, W., Gudmundsson, O., Barwood, G.: Automating DNSSEC Delegation Trust Maintenance. draft-ietf-dnsop-delegation-trust-maintainance-11, Duben 2014. http://www.ietf.org/id/draft-ietf-dnsop-delegation-trust-maintainance-11.txt

[8] Mockapetris, P.: Domain names — concepts and facilities. RFC 1034, Listopad 1987. http://www.ietf.org/rfc/rfc1034.txt

[9] Mockapetris, P.: Domain names — implementation and specification. RFC 1035, Listopad 1987. http://www.ietf.org/rfc/rfc1035.txt

[10] Richardson, M.: A Method for Storing IPsec Keying Material in DNS. RFC 4025, Březen 2005. http://www.ietf.org/rfc/rfc4025.txt

[11] Schlyter, J., Griffin, W.: Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints. RFC 4255, Leden 2006. http://www.ietf.org/rfc/rfc4255.txt

[12] Sorce, S.: Certmonger/oddjob for DNSSEC key maintenance. FreeIPA development mailing list, Září 2013. https://www.redhat.com/archives/freeipa-devel/2013-September/msg00047.html

[13] Vixie, P., Thomson, S., Rekhter, Y., aj.: Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136, Duben 1997. http://www.ietf.org/rfc/rfc2136.txt

[14] Wellington, B.: Secure Domain Name System (DNS) Dynamic Update. RFC 3007, Listopad 2000. http://www.ietf.org/rfc/rfc3007.txt

[15] Wouters, P.: Using DANE to Associate OpenPGP public keys with email addresses. Internet-Draft ietf-dane-openpgpkey-00, Duben 2014. http://www.ietf.org/id/draft-ietf-dane-openpgpkey-00.txt

# Router Turris – Nový hardware & OpenWrt

**Martin Strbačka**

E-mail: martin.strbacka@nic.cz

Je to již více než rok, kdy jsme se v Laboratořích CZ.NIC začali zabývat podporou IPv6 a DNSsec v domácích routerech a obecně jejich stavem.[1] Výsledkem stále probíhajícího průzkumu prozatím je, že málo který z běžně dostupných domácích routerů tyto technologie bez problémů zvládá. Možná ještě horším zjištěním je však fakt, že výrobci často zapomínají na podporu svých zařízení a jen málo z nich se dočká aktualizace firmware. Další průzkumy ukázaly[2], že navíc oficiální firmwary nezřídka obsahují závažné bezpečnostní problémy, či přímo výrobcem implementované zadní vrátka.

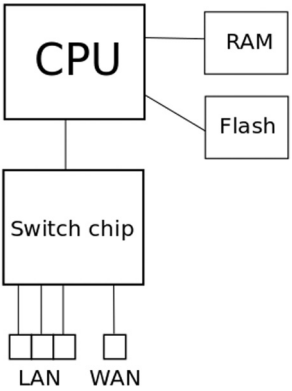
Jako odpověď na tento stav a za účelem zlepšení ochrany domácích sítí vznikl v Laboratořích CZ.NIC projekt Turris. Cílem projektu je zvýšení kybernetické bezpečnosti domácností. Tohoto stavu chceme docílit skrze zařízení – router, které bude bezproblémově podporovat výše uvedené technologie, zároveň bude mít dostatek výkonu k provozování distribuovaného adaptivního firewallu. Součástí firewallu je programové vybavení pro sběr a analýzu dat přímo v domácím routeru, software pro centrální server, který má za úkol sbírat důležitá data z jednotlivých sond – routerů, systém pro vyhodnocování získaný dat a vytváření nových firewallových pravidel a systém pro jejich distribuci na zapojená zařízení.

## Hardware routeru Turris

Výroba vlastního hardware nebyla cílem projektu Turris od samého počátku projektu. Výhodnější, snazší a časově méně náročné se jevilo, využít již existující hardware některého z renomovaných výrobců v tomto odvětví. Provedli jsme proto průzkum hardwarových řešení běžně dostupných zařízení. Jako požadovaně výkonné jsme shledali pouze zařízení z vyšších cenových relací, zpravidla určených do serverových racků, kde nízká spotřeba a hlučnost není prioritou. Menší soho routery jsou naopak zařízení nevýkonná, občas se špatným dílenským zpracováním, jejichž hardwarový návrh se dá zobecnit do následujícího blokového schéma:

[1] Viz přednáška Petra Černohouze na 42. Europenu a projekt katalogrouteru.cz.

[2] Viz csirt.cz a devttys0.com.

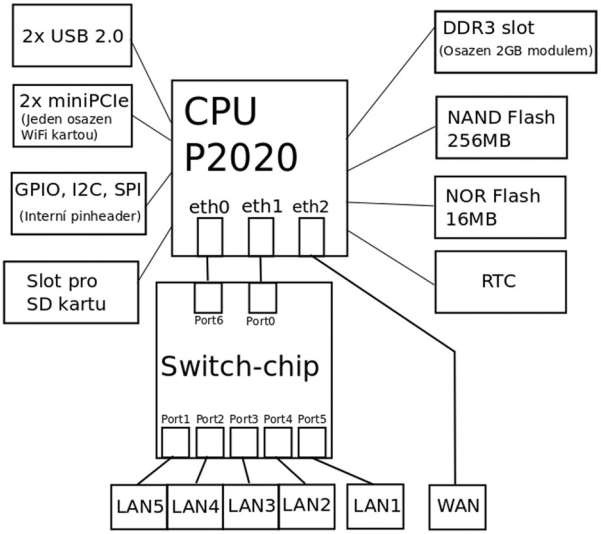



- **CPU** – Ve většině případu je architektury MIPS, jednojádrový o taktu 400 MHz až 800 MHz s integrovaným Wi-Fi modulem a jedním gigabitovým ethernetovým portem. Ostatní rozhraní implementovaná v SoC zpravidla nebývají využita ani vyvedena na konektor vyjma několika GPIO pinů, které slouží k ovládání LED diod či ovládání napájení USB portu (pokud je přítomen).
- **RAM** – Nejčastěji jde o jeden či více DDR modulů o celkové kapacitě 32 až 128 MB.
- **Flash** – Je typu NOR o kapacitě 4–16 MB, připojená přes pomalou sběrnici SPI.
- **Switch chip** – Bývá nejčastěji 5–6 portový. Jeden port je připojený do CPU, další do WAN portu a zbylé do LAN portů. Oddělené sítě pro WAN a LAN jsou realizovány pomocí VLAN. Je zřejmé, že žádný z routerů s takto zapojeným switch chipem nemůže mezi porty WAN–LAN dosáhnout rychlosti 1 Gbit.

Z výše uvedeného je zřejmé, že žádné zařízení z testovaných nesplňovalo naše představy o zařízení pro projekt Turris. Tím bylo rozhodnuto o návrhu vlastního hardwaru – routeru Turris. Jako hlavní požadavky byly stanoveny: Nízká spotřeba ($\sim$ 10 W) a hlučnost zařízení (bez aktivního chlazení), výkonné cpu s více ethernetovými porty, uživatelsky dostupná sériová konzole a další rozhraní SoC, modularita řešení (výměnný Wi-Fi a RAM modul), přítomnost RTC chipu, kompaktní rozměry.

Finálnímu řešení odpovídá následující blokové schéma:

- **CPU** – P2020 je dvoujádrový procesor architektury PowerPC taktovaný na 1,2 GHz.
- **RAM** – Je osazen pouze DDR3 slot. Standardně je Turris dodáván s 2 GB modulem.




- **Flash** – V Turrisu je použita paměť NOR i NAND. V NOR paměti je uložen zavaděč systému (u-boot) a záchranný systém. Paměť NAND obsahuje výchozí operační systém. Obě paměti jsou připojeny přes nativní paměťovou sběrnici.
- **Switch chip** – Je sedmi portový a slouží pouze pro LAN síť (WAN je připojen přes samostatné rozhraní) s procesorem je propojen dvěmi ethernetovými linkami. Toto zapojení umožňuje např. nastavení port trunkingu.
- **Wi-Fi karta** – Použita je miniPCIe Wi-Fi karta 802.11a/b/g/n s $3 \times 3$ MIMO a odnímatelnými anténami.
- **Další rozhraní** – V Turrisu lze využít volného miniPCIe slotu, slotu pro SD karty, rozhraní I2C, SPI a několika GPIO vyvedených na interní pinheader. Sériová konzole je dostupná skrze microUSB konektor skrytý za čelním panelem.

## Operační systém – OpenWrt

Díky svému zaměření na domácí routery bylo OpenWrt jasnou volbou už od samého počátku projektu Turris. Projekt OpenWrt označujeme jako linuxovou meta-distribuci či framework, což zjednodušeně znamená, že OpenWrt není primárně distribuováno v binární formě, ale jako zdrojový kód sestavovacího prostředí Buildroot-NG. Každé zařízení které je v OpenWrt podporováno musí mít

v Buildroot-NG definovaný profil, který specifikuje jak přeložit jednotlivé součásti systému a jak výsledné binární soubory (kernel, FDT, rootfs) složit v jeden soubor, který bude možné nahrát do Flash paměti cílového zařízení.

Pro přidání profilu nového zařízení do OpenWrt je nutnou podmínkou podpora samotné architektury a typu procesoru, na kterém je zařízení postaveno. V případě routeru Turris to nebyl problém, procesor P2020 spadá pod architekturu PowerPC a rodinu procesorů mpc85xx, která má v aktuální vývojové verzi OpenWrt dobrou podporu. Další podmínkou je podpora všech komponent routeru (SoC, switch chip, paměťová zařízení, rtc, atd.) v linuxovém jádře. Ta zpravidla nebývá ve vanilla kernelu na vysoké úrovni, naštěstí projekt OpenWrt spravuje vlastní patchset, který již podporu pro všechny komponenty Turrise obsahuje. Komponenty embedded systémů bývají propojeny na nejnižších hardwarových úrovních, které zpravidla neumožňují kernelu provádět žádnou (a nebo velmi omezenou) autodetekci, z toho důvodu je nutné kernel pro každé cílové zařízení náležitě zkonfigurovat. Dříve se tato konfigurace prováděla z části parametry kernelu uvedenými v zavaděči systému (např. rozdělení Flash paměti) a kernelovými patchi. Architektura PowerPC byla první, která migrovala z toho ne příliš pružného způsobu na technologii FDT. FDT (Flattened device tree) definuje syntaxi konfiguračního souboru, který popisuje nastavení komponent systému (MAC adresy, frekvence, fyzické propojení), tento soubor je následně zkompilován do binárního blobu a nahrán do zařízení spolu s kernelem. Informace uvedené ve FDT jsou pak pomocí getterů přístupné v celém kernelu a všech modulech. Sepsání FDT bohužel, není díky chabé dokumentaci úplně snadné. Nejsnazší je inspirovat se v již existujících FDT souborech příbuzných zařízení a čtením kódu kernelových modulů objevovat názvy konfiguračních proměnných.

Vytvoření profilu pro Turris bylo poměrně snadné. Jako problém se ukázala potřeba zkompilování systémů (záchranného a hlavního) pro dvě rozdílné Flash paměti Turrise. Profil zařízení v Buildroot-NG totiž definuje zařízení jako spojení jedné konfigurace kernelu s jednou technologií cílové Flash paměti. Problém byl vyřešen nepříliš elegantně zato velmi funkčně sepsáním dvou profilů (jeden pro NAND paměť druhý pro NOR) a vytvořením wrapper skriptu, který kompiluje oba profily zvlášť a výsledné soubory spojí do potřebného formátu.

Další naší úpravou, je podpora pro rolling updates. Abychom umožnili systému aktualizovat každý balíček zvlášť, bylo nutné provést dva větší zásahy do systému OpenWrt. Nejprve bylo nutné změnit typ filesystému z overlayfs na jffs2, který je celý read-write. Druhým krokem byla reorganizace kompilačních kroků v Buildroot-NG tak, aby první zkompilovanou komponentou systému byl kernel, díky tomu je možné z něj v následujícím kroku vytvořit balíček a nainstalovat jej v běžícím systému.

# Linuxový dotykový kiosek

**Miloš Wimmer**

E-mail: wimmer@zcu.cz

**Abstrakt**

*Příspěvek je věnován koncepci sítě kiosků provozované na Lékařské fakultě UK a ve Fakultní nemocnici Plzeň. Zaměřuje se na popis návrhu a stavby linuxového systému pro velké 42" interaktivní dotykové zobrazovače, které byly pro kiosky použity. Obsahuje i zkušenosti z provozu a možnosti širšího využití.*

## Úvod

Před časem mne oslovili kolegové z Lékařské fakulty UK v Plzni, zda bych neměl zájem podílet se na podávaném projektu OP VK „Modernizace didaktických metod cestou podpory systému elektronického vzdělávání (MODIM)". Cílem tohoto projektu je podpora pedagogů při tvorbě elektronických studijních materiálů a kurzů pro studenty. Důraz je kladen na použití moderních internetových a technických prostředků. Nové materiály jsou publikovány na edukačním portálu MEFANET nejčastěji ve formátu PDF, multimediální materiály ve formátu Flash.

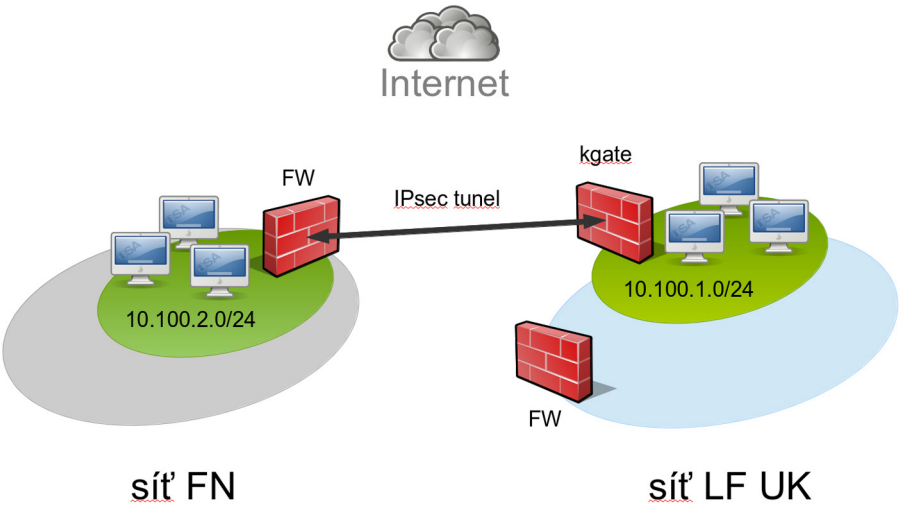
Součástí projektu je také použití velkoplošných elektronických dotykových kiosků instalovaných ve veřejných prostorách Lékařské fakulty a Fakultní nemocnice v Plzni. To byla část, které jsem se měl ujmout. Téma mne zajímalo, nabízelo přesahy do různých směrů IT, proto jsem rád souhlasil. Následně byl projekt přijat a mohl jsem se pustit do práce...

## Koncepce

Kiosky měly být zapojeny do počítačových sítí LF UK a FN. V nich vystupují jako „hostovaná" zařízení pod cizí správou. To z pohledu správců těchto sítí není ideální stav, proto jsme kiosky zapojili do izolovaných L2 sítí, z nichž

nemají přímý přístup k cílům v hostitelských sítích a nemají ani přímou konektivitu do Internetu. Veškeré síťové služby poskytuje kioskům řídící server a současně firewall kgate připojený do veřejného segmentu sítě LF a do sítě kiosků. Obě sítě kiosků jsou propojeny tunelem IPsec vytvořeným mezi serverem kgate a firewallem sítě FN.

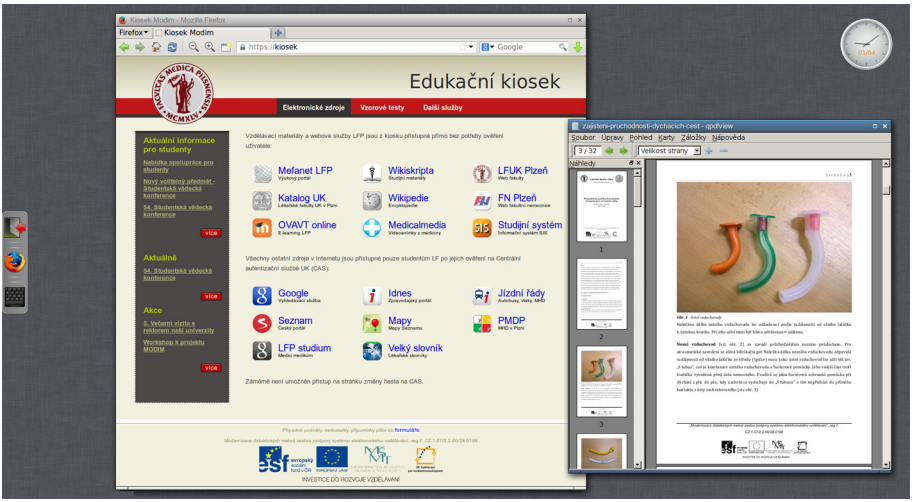
Kgate kiosky ovládá, centrálně je zapíná i vypíná a poskytuje jim řízený přístup k cílům v Internetu přes aplikační proxy server. K vyjmenovaným vzdělávacím cílům umožňuje přístup bez ověření, ke všem ostatním pouze po ověření proti autentizační službě CAS.




Obr. 1: Topologie sítě kiosků

Hlavní požadavky nositele projektu na kiosky se dají shrnout do následujících bodů:

- atraktivní grafické prostředí
- řízený přístup k prezentovaným dokumentům projektu MODIM i k jiným cílům v Internetu
- ověřování uživatelů proti autentizační službě CAS UK
- podpora formátů prezentovaných dokumentů (PDF, flash)
- podrobné statistiky přístupů
- vyřešení integrace kiosků do počítačových sítí LF a FN
- stabilita
- automatizované sledování celé infrastruktury dohledovou službou




Obr. 2: Ukázka desktopu kiosku

K nim jsem přidal své vlastní požadavky:

- nerozbitný systém
- co nejmenší závislost na infrastruktuře okolních sítí LF a FN
- zajištění bezpečného prostředí pro uživatele, tedy zejména integrity dat na lokálním disku, zabezpečeného přenosu dat sítí a přístupu k zavedenému systému ověřování uživatelů
- centralizovaná a automatizovaná správa
- všechny kiosky budou mít identický obraz disku
- otevřený svobodný systém umožňující další rozšiřování

Plocha zobrazovače kiosku je opravdu velká, navíc z povahy práce se vzdělávacími materiály vyplývá výhoda současného otevření několika dokumentů současně. Dotykový display zase uživatele automaticky vede k tabletovému ovládání. Proto jsem se rozhodl vytvořit systém, který bude mít klasické desktopové prostředí oken, ale s ovládáním tabletu. Celé prostředí je silně zaměřeno na cíl – kolemjdoucího uživatele nelze rozptylovat nejasnostmi v orientaci ani v ovládání, proto je grafické prostředí desktopu velmi srozumitelné a obsahuje jen ty aplikace, které uživatel k práci opravdu potřebuje. Těmi jsou:

- WWW prohlížeč s podporou flashe
- PDF prohlížeč
- virtuální klávesnice
- animovaný spouštěcí panel

Po uplynutí doby několika minut bez aktivity a bez předchozího odhlášení dochází k automatickému odhlášení uživatele a celé prostředí desktopu se nastaví do výchozího stavu. Při delší neaktivitě se kiosek přepíná do režimu elegantního zobrazování novinek portálu MEFANET a aktuálních zpráviček ze života fakulty.

## Systém kiosku

Pro kiosky bylo vybráno zařízení **Elo TouchSystems ET4200L**, které tvoří velký 42" dotykový LED panel s rozlišením 1 920 × 1 080 bodů s počítačem mini PC zabudovaným do těla zobrazovače v podobě vyjímatelného kovového šuplíku. Do stran zobrazovače jsou integrovány reproduktory. Všechny kiosky jsme nainstalovali na stěny a doplnili je zajištěním proti demontáži a přístupu k portům.

Počítač PC obsahuje procesor Intel Core Duo/3 GHz, 2 GB RAM, disk 160 GB, gigabitovou ethernetovou kartu, akcelerovanou grafickou kartu Intel a USB porty.

Celý kiosek jsem postavil nad operačním systémem Linux Debian amd64. Nabízí otevřený systém s rozsáhlými možnostmi volby komponent, je to pro mne známé prostředí a vůbec je prostě nejlepší ;-) Celý hardware kiosku je přímo podporován distribučním jádrem 3.x.

Operační systém jsem se snažil udržet co možná nejmenší. I proto jsem na pozici grafického prostředí použil namísto běžných, ale objemných systémů GNOME, MATE, KDE nebo Xfce minimalistický systém založený na standardním **xserveru xorg**, správci oken **openbox**, kompozitním správci stínování **xcompmgr**, animovaném spouštěcím panelu **cairo-dock** a virtuální klávesnici **florence**. Dojem tabletového ovládání navozuje neviditelný kurzor myši v podobě transparentního tématu **xcursor-transparent**.

Komunikaci s uživatelem zajišťuje dialogové prostředí **zenity** založené na rozhraní GTK+. Funkci automatického odhlašování i spouštění režimu zpráviček provádí program **xautolock**. Zprávičky zobrazuje efektním prolínáním obrazovek **xscreensaver** s upraveným modulem **gslideshow**.

Na pozici WWW prohlížeče jsem zvolil **Mozilla Firefox**, který disponuje velkým množstvím rozšiřujících doplňků. Pro kiosek zásadní je zejména **Grab and Drag**, který dovoluje ovládat obsah okna prstem tabletovým stylem „chytni a táhni" i dynamickými gesty „švihnutím". Doplněk **Personal Menu** dovoluje omezit ovládací menu prohlížeče jen na určité položky a doplněk **Public Fox** znemožňuje uživateli provádět změny konfigurace prohlížeče i přistupovat k nastavení about:config.

Pro práci s dokumenty PDF jsem zvolil prohlížeč **qpdfview**. Poskytuje mnohem lepší zpracování PDF souborů než prohlížeč PDF integrovaný ve Firefoxu a přitom podporuje ovládání obsahu okna prstem stylem „chytni a táhni", takže práce s ním je konzistentní s ovládáním Firefoxu.

## Souborový systém

Operační systém kiosku je uložen na jeho lokálním pevném disku. Disk jsem rozdělil na čtyři oddíly. Oddíl sda1 (velký 10 GB, použito 1.4 GB) obsahuje filesystém s daty systémového disku /. Oddíl sda2 (2 GB) slouží pro swap. Filesystém na oddílu sda3 (10 GB) se používá pro adresáře /tmp, /var/log, /var/tmp a /home/kiosk. Poslední oddíl sda4 obsahuje filesystém s daty alternativního systémového disku, který se používá jen při servisních operacích, jako např. při aktualizaci hlavního systému. Všechny filesystémy jsou typu xfs.

Všechny kiosky mají identický obraz všech oddílů disku a kvůli možnosti použití stejného konfiguračního souboru grubu mají i stejná UUID jednotlivých filesystémů. Pro dosažení „nerozbitnosti“ systému je systémový filesystém připojený jako read-only, takže se nemůže dostat do nekonzistentního stavu. Kiosky získávají základní síťová nastavení z DHCP serveru. Standardní skripty upravují podle nich systémové soubory, ale je-li systémový disk chráněn proti zápisu, nemohly by být úspěšné. Snažil jsem se minimalizovat zásahy do distribučního systému, a proto jsem použil techniku překrývajících souborů, kdy při startu systému vytvořím v tmpfs v RAM soubory, kterými pak příkazem typu

```
mount - -bind /run/kiosk/etc/hostname /etc/hostname
```

překryji původní soubory. Tím získám přepisovatelné systémové soubory na read-only systémovém disku bez potřeby úprav standardních skriptů, které s nimi pracují.

Pro adresáře vyžadující přepisování větším objemem dat (např. /var/log, /tmp, /home/kiosk) používám adresáře na filesystému oddílu sda3, který se vytváří při každém startu kiosku znovu a je připojen s běžnými právy zápisu. Tyto adresáře jsou pak k systémovému disku připojeny výše uvedeným způsobem.

Grafické prostředí xserveru běží pod identitou uživatele kiosk. Celá jeho konfigurace je zapsána v domovském adresáři /home/kiosk, kam má uživatel kiosku právo zápisu. Pro zaručení čistého a bezpečného prostředí je proto po každém odhlášení uživatele celý adresář smazán a z read-only archivu vytvořen znovu.

Každý kiosek má navíc připojený sdílený filesystém NFS.

Část výpisu připojených filesystémů a souborů vypadá takto:

```
root@k1:~# mount

sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)

proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)

udev on /dev type devtmpfs (rw,relatime,size=10240k,
nr_inodes=247009,mode=755)
```

```
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,
size=199140k,mode=755)

/dev/disk/by-uuid/fadd27f2-309a-48fb-baea-5560547cadc9 on /
type xfs (ro,noatime,attr2,inode64,noquota)

tmpfs on /etc/hostname type tmpfs (rw,nosuid,noexec,relatime,
size=199140k,mode=755)

tmpfs on /etc/hosts type tmpfs (rw,nosuid,noexec,relatime,
size=199140k,mode=755)

tmpfs on /etc/resolv.conf type tmpfs (rw,nosuid,noexec,relatime,
size=199140k,mode=755)

kgate:/usr/local/share/gallery on /usr/local/share/gallery
type nfs (ro,nosuid,nodev,relatime,vers=3,rsize=4096,wsize=4096,
namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,
mountaddr=10.100.1.1,mountvers=3,mountport=56395,
mountproto=udp,local_lock=none,addr=10.100.1.1)
```

## Řídící server kgate

Server plní pro síť kiosků funkci firewallu a poskytovatele všech síťových služeb. Má dvě síťová rozhraní, jedním je připojen do vnějšího internetového segmentu sítě LF, druhým do sítě kiosků. Na kgate běží operační systém Debian amd64. Server poskytuje kioskům služby DHCP pro vyjmenované MAC adresy, DNS, NTP a NFS. Dále pracuje jako centrální syslog server, na který posílají kiosky své logy. Běží na něm domovský web server sítě kiosků s navigací na nejvýznamnější cíle, aplikační proxy server **squid** s ověřováním uživatelů proti centrální ověřovací službě CAS UK a démon IPsec tunelu **racoon** pro propojení se vzdálenou sítí kiosků ve FN.

WWW prohlížeče kiosků přistupují k cílům v Internetu řízeným způsobem přes aplikační proxy server. Ten zpřístupňuje vyjmenované vzdělávací cíle a cíle v sítích LF a FN přímo, bez nutnosti ověření uživatele. Ostatní cíle zpřístupňuje jen po ověření uživatele proti CAS UK v Praze. Používáme ověření jménem a heslem každého uživatele, experimentálně jsme vyzkoušeli i ověřování čipovou kartou ISIC. Všechny přístupy jsou logovány.

Je-li jméno a heslo uživatele přenášeno mezi WWW prohlížečem a proxy serverem sítí, je nezbytné zajistit jejich zabezpečený přenos. Zjištění, že Firefox

nepodporuje HTTPS komunikaci s proxy serverem a nemá k dispozici ani potřebný doplněk, pro mne bylo překvapivé. Otevřený systém však dovoluje řešit problémy různými způsoby. Zabezpečené spojení jsem tedy vytvořil šifrovaným tunelem mezi kioskem a serverem kgate. Na straně kiosku jsem použil aplikaci **stunnel** a v konfiguraci Firefoxu jsem nastavil proxy server na jeho lokální začátek http://localhost:3128. Na straně kgate jsem chtěl tunel zakončit přímo v aplikaci proxy serveru **squid**, ale jeho distribuční verze HTTPS komunikaci s klientem nepodporuje. Proto jsem na straně kgate použil pro zakončení tunelu také aplikaci stunnel, kterou jsem propojil s lokálním vstupním HTTP portem proxy serveru. Tímto způsobem jsou pak veškerá data přenášená mezi WWW prohlížečem kiosku a proxy serverem na kgate transportována šifrovaným tunelem. Toto řešení funguje velmi dobře, ale v logu proxy serveru jsou všechny přístupy zaznamenávány s lokální zdrojovou adresou kgate, protože to je zdrojová adresa klienta po výstupu z stunnelu:

```
24/Mar/2014:07:30:23 +0100 19 127.0.0.1 TCP_MISS/301 949 GET
http://www.wikiskripta.eu/ -
HIER_DIRECT/www.wikiskripta.eu text/html
```

Tím přicházíme o důležitou informaci identifikace klienta. Ve zdrojových kódech proxy serveru squid je HTTPS komunikace s klientem k dispozici. Její přínos mi stál za to, abych squid přeložil s direktivami pro HTTPS přístup klientů a použil ho namísto distribučního balíčku. Tím jsem mohl zakončit šifrovaný tunel od kiosku přímo na vstupním HTTPS portu squidu. To znamená, že na serveru kgate již není potřeba aplikace stunnelu a v logu proxy serveru jsou všechny přístupy zaznamenávány se skutečnou zdrojovou adresou příslušného kiosku :

```
24/Mar/2014:07:35:49 +0100 19 10.100.1.102 TCP_MISS/301 949 GET
http://www.wikiskripta.eu/ -
HIER_DIRECT/www.wikiskripta.eu text/html
```

Kiosky jsou ovládány ze serveru kgate příkazem **kadm**

```
kadm -c {boot|halt|reboot|start|stop|restart|status}
{-n} {kiosek|all} [kiosek kiosek ...]
```

Tento příkaz pak spouští přes službu NRPE odpovídající příkaz na určeném kiosku. Výjimkou je povel **boot**, který přímo vyvolává příkaz wakeonlan vysílající paket umožňující zapnutí vypnutého kiosku.

Výstup povelu **status** vypadá např. takto:

```
# kadm -c status k1
k1 up: 2:56; session: 00:11:52; xscreensaver: 00:00:00;
load: 0.04; temp: +53.1 C; version:
1396456389.1392161302
```

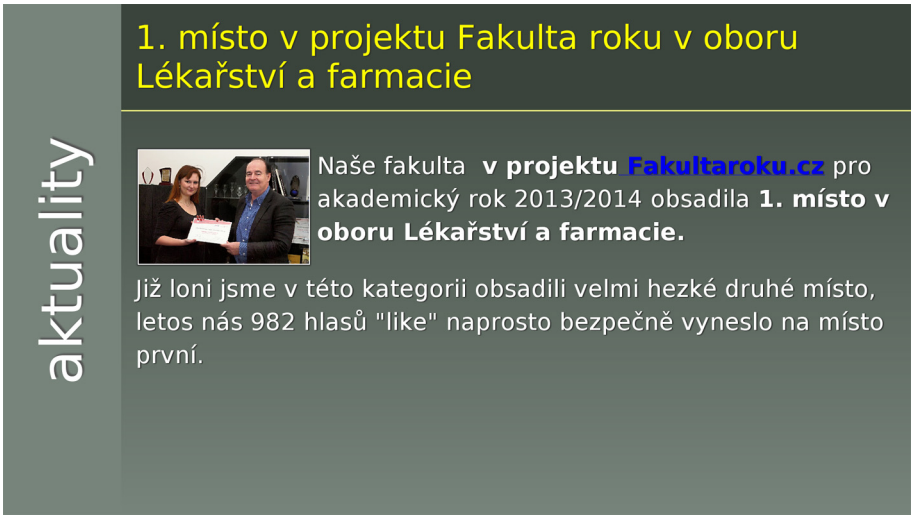
## Zobrazování novinek

Lékařská fakulta provozuje aplikační server, do kterého jsou pro účely prezentace na webu fakulty zadávány aktuální novinky a zprávičky v kategoriích akce, aktuality, informace pro studenty a novinky portálu MEFANET. Rozhodli jsme se využít těchto živých informací pro jejich prezentaci na kioscích v době, kdy na nich nikdo nepracuje.

Na serveru kgate je periodicky spouštěn skript, který z databáze aplikace novinek získává aktuální data a vkládá je do HTML šablony vytvořené pro příslušnou kategorii. Ve virtuálním prostředí X serveru pak spustí aplikaci **cutycapt**, která jádrem WebKitu vykreslí HTML stránku do grafického souboru, zkonvertuje ho do formátu png a ořízne ho na velikost plného rozlišení panelu $1\,920 \times 1\,080$ bodů. Nakonec uloží výsledný obrázek do sdíleného adresáře, který je kioskům dostupný přes NFS.

Na kiosku se po uplynutí doby několika minut bez aktivity spouští xscreensaver, který upraveným modulem **gslideshow** sekvenčně zobrazuje obrázky všech kategorií dostupných na sdíleném NFS filesystému. Namísto obrázků novinek lze použít i obrázky s prezentacemi nebo jiným obsahem.

Tímto plně automatizovaným procesem se nám podařilo smysluplně využít kiosek i v době, kdy s ním uživatel nepracuje. Navíc stále aktuální informace pomáhají udržovat kiosek „živý“.




Obr. 3: Ukázka zobrazování novinek

# Dohled nad infrastrukturou

Pro dohled nad celou infrastrukturou používáme monitorovací systém **Icinga**. Jím sledujeme síťové a interní služby i systémové hodnoty všech kiosků a serveru kgate. Sledování se provádí přímo i přes službu NRPE, používáme standardní pluginy nagiosu i pluginy napsané pro naše potřeby.

| Host | Service | Status | Last Check | Duration | Attempt | Status Information |
|---|---|---|---|---|---|---|
| k1.kiosk.lfp.cuni.cz | DISK / | OK | 25-03-2014 09:47:05 | 0d 3h 18m 6s | 1/4 | DISK OK - free space: / 8141 MB (85% inode=99%): |
| | DISK /opt/rw | OK | 25-03-2014 09:48:24 | 0d 3h 16m 47s | 1/4 | DISK OK - free space: /opt/rw 10144 MB (99% inode=99%): |
| | INTERFACE eth0 | OK | 25-03-2014 09:49:43 | 0d 3h 15m 28s | 1/4 | eth0:UP OK |
| | LOAD | OK | 25-03-2014 09:46:30 | 0d 3h 18m 41s | 1/4 | OK - load average: 0.00, 0.01, 0.05 |
| | MEM | OK | 25-03-2014 09:49:49 | 0d 3h 15m 22s | 1/4 | MEMORY OK - total/used: 1991392/641604 KB |
| | NTP | OK | 25-03-2014 09:49:08 | 0d 3h 16m 3s | 1/4 | NTP OK: Offset -3.218650818e-06 secs |
| | PING | OK | 25-03-2014 09:46:27 | 0d 3h 18m 44s | 1/4 | PING OK - Packet loss = 0%, RTA = 0.30 ms |
| | PROCESS firefox | OK | 25-03-2014 09:49:22 | 0d 3h 15m 49s | 1/4 | PROCS OK: 1 process with command name 'firefox' |
| | PROXY | OK | 25-03-2014 09:48:23 | 0d 3h 16m 48s | 1/4 | HTTP OK: HTTP/1.1 200 OK - 9204 bytes in 0.025 second response time |
| | SESSION | OK | 25-03-2014 09:46:33 | 0d 3h 18m 38s | 1/4 | OK - session time: 03:16:06 |
| | TEMPERATURE | OK | 25-03-2014 09:49:52 | 0d 3h 15m 19s | 1/4 | OK - temperature: +52.9°C |
| | UPTIME | OK | 25-03-2014 09:46:36 | 0d 3h 18m 35s | 1/4 | OK : up 0 days, 03:16:21 |
| | XSCREENSAVER | OK | 25-03-2014 09:46:29 | 0d 3h 18m 42s | 1/4 | OK - xscreensaver time: 03:10:55 |

Obr. 4: Služby sledované na kiosku

# Aktualizace systému kiosku

Pro aktualizaci operačního systému kiosků používám způsob, který se mi osvědčil při vývoji kiosku. Je poměrně jednoduchý a bezpečný.

Na referenčním kiosku přepojím systémový filesystém na sda1 do režimu pro zápis (read-write). Běžným způsobem udělám update systému, zkontroluji, že vše je v pořádku a pak příkazem

```
grub-reboot 1 ; reboot
```

rebootuji kiosek bez potřeby úprav konfiguračního souboru grubu do alternativního servisního systému na sda4. V servisním systému připojím systémový filesystém sda1, vytvořím jeho tar archiv sda1.tgz a pořídím jeho kontrolní součet sda1.tgz.md5sum. Oba tyto soubory pak přenesu na řídící server kgate, kde je uložím do adresáře dostupného kioskům přes NFS.

Při aktualizaci cílového kiosku přepojím jeho systémový filesystém na sda1 do režimu pro zápis (read-write) a příkazem

```
grub-reboot 2 ; reboot
```

ho rebootuji do alternativního servisního systému na sda4 do režimu automatické aktualizace. V ní se porovná lokálně nainstalovaná verze archivu systémového

filesystému sda1.tgz s verzí aktuálně dostupnou na NFS, při zjištění rozdílu se na lokální disk zkopíruje archiv sda1.tgz z NFS, provede se kontrola md5sum, původní obsah filesystému na sda1 se přepíše přeneseným archivem a nakonec se provede reboot do aktualizovaného systému na sda1.

## Zkušenosti s reálným provozem

Deset kiosků máme v provozu 6 měsíců, zapínají se automaticky každý pracovní den v 6.30 a vypínají se ve 20 hodin. Za tuto dobu jsme v rámci celého systému nezaznamenali žádný výpadek.

Uživatelé nemají s orientací v prostředí kiosku ani s jeho ovládáním potíže. Počáteční nezvyk působí o něco silnější sklo panelu zobrazovače, takže prstem se na něm musí tlačit o poznání více, než je zvykem na klasických tabletech nebo dotykových telefonech.

# Moderní Perl

**Lukáš Rampa**

E-mail: lukas.rampa@aimtec.cz

V posledních letech zaznamenal Perl jisté obrození. Vývoj probíhá v pravidelných cyklech a kolem jazyka se děje mnoho zajímavého.

## Historie

Paralelně probíhá vývoj verzí 5 a 6. Perl 6 ještě stále není ve stavu použitelném pro běžného smrtelníka, dále se budeme věnovat jen Perlu 5.

Od roku 2010, kdy byla vydána verze 5.12 probíhá vývoj v ročních cyklech, v květnu 2014 vyjde verze 5.20 (sudá čísla označují produkční verzi). Podporovány jsou vždy dvě poslední verze.

Termín Moderní Perl definoval chromatic v knize Modern Perl: The Book v roce 2010. Změny, které pod pojem Moderní Perl zahrnuje, ale začaly už v roce 2001 s vydáním verze 5.6. Jednalo se o nové vlastnosti, zjednodušující použití Perlu, důraz na čistý styl programování, využívání existujících modulů z CPANu, lepší podpora psaní automatických testů, apod.

## Moderní Perl

### CPAN

CPAN (Comprehensive Perl Archive Network) je systém pro distribuci modulů Perlu. K CPANu existují dvě webová rozhraní – www.cpan.org a novější metacpan.org. CPAN archive má množství zrcadel po celém světě.

Hlavním cílem CPANu je usnadnit vyhledávání existujících modulů (přes 29 000 distribucí v dubnu 2014). Součástí modulů je vždy jejich dokumentace, přístupná přes webové rozhraní CPANu.

Moduly z CPANu se instalují pomocí utility cpan (součást distribuce Perlu) nebo alternativní cpanm Tatsuhiko Miyagawy. cpanm poskytuje zjednodušené rozhraní, nezahlcující uživatele přemírou detailů.

Vedle CPANu vznikla řada souvisejících služeb. Pravděpodobně nejvýznamnější z nich je CPAN Testers (cpantesters.org), který soustřeďuje výsledky testů jednotlivých modulů. Testy se spouštějí automaticky během instalace modulů a také explicitně testery, kteří testují všechny nové verze modulů.

## ORM/DBIx::Class

DBIx::Class (také DBIC) je objektově relační mapper (á la ActiveRecord v ruby nebo JPA v Javě). Pod kapotou používá standardní databázové rozhraní DBI/DBD, podporuje vice než deset typů databází. Snahou DBIx::Class je umožnit práci s databází co nejvíce „Perlím“ způsobem.

Pro každou tabulku se definuje třída typu Result, výsledky dotazu jsou zabaleny v ResultSet. Vše zastřešuje Schema. DBIx::Class podporuje jak generování tříd z existující databáze, tak vytvoření databázových objektů z předem připravených tříd.

Užitečnou vlastností je schopnost překladu hodnot sloupce za letu (například pro de/konstrukci DateTime objektu, formátování hodnot, konverze jednotek atd.).

## Moose

Moose je objektový systém inspirovaný verzí 6 Perlu. Původní způsob práce s objekty v Perlu nebyl příliš pohodlný, Moose je pokusem navrhnout vše „správně“, což se zdá se podařilo.

Podporována je vícenásobná dědičnost, role, modifikátory metod a další.

Role jsou obdobou Java rozhraní, ale na rozdíl od rozhraní mohou obsahovat také implementaci svých metod. Moose třída může implementovat více rolí.

Modifikátory metod dovolují obalit zděděnou metodu nebo metodu role dodatečným kódem a tak modifikovat parametry metody před jejím voláním nebo výsledek metody po jejím provedení.

Moose podporuje také datové typy, atribut třídy může mít přiřazen jeden ze základních typů nebo libovolnou třídu.

Pro Moose existuje velké množství pluginů (MooseX::*). Za zmínku stojí např. MooseX::Types pro pohodlnou definici vlastních datových typů.

## Web frameworky, PSGI/Plack

Základním kamenem aktuálních web frameworků je PSGI (Perl Web Server Gateway Interface). PSGI je specifikace, popisující jednotné rozhraní mezi web serverem a Perl aplikací, díky němu jsou aplikace přenositelné mezi web servery. Implementací PSGI je modul Plack.

Nejznámějšími frameworky jsou Catalyst, Dancer a Mojolicious. Všechny tři jsou dostatečně vyzrálé pro použití v produkci (např. zmiňovaný metacpan.org běží nad Catalystem).

# Komunita

Komunita Perlu měla ne vždy dobrou pověst, nevlídné chování k začátečníkům nebávalo výjimkou. V posledních letech je situace radikálně jiná. Hrubé chování se netoleruje, viník bývá z diskuze vykázán.

Hlavním místem pro virtuální setkání s komunitou je IRC s domovskou stránkou irc.perl.org. Vyhrazené kanály existují pro jednotlivé významné moduly i pro geografické zájmové skupiny. Je zde možné rychle získat potřebné informace nebo jen poklábosit.

Důležitou roli hrají Perl Mongers[1], lokální i virtuální skupiny uživatelů Perlu. Typická Perl Mongers skupina se pravidelně schází k debatám o technologiích a k dalším společenským radovánkám.

V mnoha zemích světa se jednou ročně konají Perl Workshopy – mini-konference na téma Perl. V roce 2014 se uskuteční první český Perl Workshop.[2]

Jednou ročně se v Evropě, Asii a Americe koná YAPC – Yet Another Perl Conference. Zpravidla několikadenní akce hostí stovky účastníků, přednášky probíhají současně v několika bězích. Nejbližší YAPC Europe proběhne v srpnu 2014 v Sofii.

Většina konferencí používá pro svou organizaci systém Act, na jeho stránkách je také k dispozici seznam všech minulých i plánovaných akcí.[3]

[1] http://www.pm.org
[2] http://act.yapc.eu/czpw2014
[3] http://act.mongueurs.net/conferences.html