VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE SLUŽBY PROVISIONING V OPEN SOURCE PBX ASTERISK

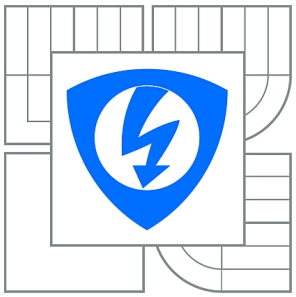
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAN KRÁL
AUTHOR

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE SLUŽBY PROVISIONING V OPEN SOURCE PBX ASTERISK

PROVISIONING IMPLEMENTATION IN ASTERISK OPEN SOURCE PBX

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

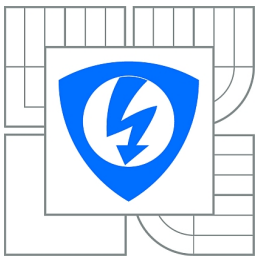
AUTOR PRÁCE — JAN KRÁL

AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL ŠILHAVÝ, Ph.D.

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Jan Král ***ID:*** 146870
Ročník: 3 ***Akademický rok:*** 2013/2014

NÁZEV TÉMATU:

Implementace služby provisioning v Open source PBX Asterisk

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte problematiku automatické konfigurace koncových zařízení pro VoIP telefonii (provisioning) v Open source PBX Asterisk. Podrobně popište obecné možnosti tohoto přístupu. S využitím PBX Asterisk implementujte tuto službu v pro koncová VoIP zařízení dostupná v laboratoři, zejména zařízení výrobců Well a Linksys. Realizujte nástroj pro centralizovanou správu VoIP zařízení dostupných v laboratoři.

DOPORUČENÁ LITERATURA:

[1] Meggelen, J.V, Smith, J., Madsen, L. Asterisk™ The Future of Telephony. Sevastopol: O'Reilly Media, Inc., 2005. ISBN 0-596-00962-3.
[2] Bosse, J.G.. Signaling in telecommunication networks. John Wiley & Sons, Ltd. En-gland 2002 , ISBN 0-471-66288-7.
[3] Collins, D. Carrier grade voice over IP / 2nd ed. New York : McGraw-Hill, 2003. ISBN 0-07-140634-4.

Termín zadání: 10.2.2014 ***Termín odevzdání:*** 4.6.2014

Vedoucí práce: Ing. Pavel Šilhavý, Ph.D.
Konzultanti bakalářské práce:

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

UPOZORNĚNÍ:

ABSTRAKT

Bakalářská práce „Implementace služby provisioning v Open source PBX Asterisk“ se v teoretické části věnuje obecnému popisu služby provisioning a možnostem její implementace do PBX Asterisk z hlediska samotné ústředny a koncových zařízení pro VoIP telefonii k ústředně připojených. Rovněž jsou stručně rozebrány všechny protokoly a služby, které konkrétní scénář implementace popisovaný v práci vyžaduje ke správnému fungování služby. Dále je v práci popsán návrh a požadavky na aplikaci pro centralizovanou správu koncových zařízení. Praktická část práce je zaměřena na instalaci a zprovoznění ústředny a samotné služby provisioning na virtuálním serveru s operačním systémem CentOS. Práce také obsahuje veškeré nastavení samotného serveru, potřebných služeb, ústředny Asterisk a také koncových zařízení pro VoIP telefonii, které je nutné ke správnému fungování této služby v rámci scénáře implementace popsaného v teoretické části práce. Rovněž je zde popsána navržená a vytvořená aplikace pro správu zařízení.

KLÍČOVÁ SLOVA

Asterisk, PBX, provisioning, automatická konfigurace, VoIP, webový nástroj, PHP, MySQL, databáze

ABSTRACT

The bachelor's thesis "Implementation of provisioning service in Open source PBX Asterisk" deals with a general description of the provisioning service and the options of it's implementation in Asterisk PBX based on the capabilities of the PBX and the endpoint devices for VoIP telephony connected to it. The theoretical part of the thesis also contains a brief description of all the protocols and services needed for successful operation of the service in the particular scenario of implementation described in the thesis. A short introduction and requirements for the designed application are also described. The practical portion of the thesis is focused on instalation and setup of the Asterisk PBX and the provisioning service itself on a virtual server running the CentOS operating system. It also contains all the settings of the server, additional services run on the server, the Asterisk PBX and also the settings of the endpoint devices needed for correct operation of the service. A thorough description of the designed and developed web application for management of endpoint devices for VoIP telephony is also contained in this part.

KEYWORDS

Asterisk, PBX, provisioning, automatic configuration, VoIP, web tool, PHP, MySQL, database

KRÁL, Jan *Implementace služby provisioning v Open source PBX Asterisk*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 62 s. Vedoucí práce byl Ing. Pavel Šilhavý, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Implementace služby provisioning v Open source PBX Asterisk“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Toto poděkování chci věnovat vedoucímu mé bakalářské práce panu Ing. Pavlovi Šilhavému, Ph.D. za jeho odborné vedení, konzultaci, rady, připomínky a podnětné návrhy k této práci.

Brno ....

(podpis autora)

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

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

PODĚKOVÁNÍ

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

Brno ....

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM UKÁZEK

ÚVOD

V dnešní době se mnoho organizací s vlastní IT infrastrukturou uchyluje k implementaci konvergovaných sítí, kdy je jedna a ta samá IP síť používána jak pro datové aplikace, tak pro aplikace fungující v reálném čase – VoIP nebo streamování médií. Konvergované sítě umožňují zaměstnancům těchto organizací pomocí jednotné komunikace a různých technologií pro spolupráci pracovat produktivněji a tím zvyšují i produktivitu a výkonnost celé organizace. Zároveň s tím ovšem vyvstává otázka, jakým způsobem lze služby poskytované v těchto sítích spravovat bez zvýšení vyžadovaného úsilí a výdajů.

Odpověďí na tuto otázku může být právě služba provisioning, nebo-li automatická konfigurace koncových zařízení. Tato služba umožňuje centralizovaně spravovat koncová zařízení a šetří tak čas a náklady, které by si jinak ruční konfigurace každého jednotlivého zařízení vyžádala. S touto službou je možné se setkat například v sítích poskytovatelů triple-play [1] služeb, kde bývá využívána pro správu IPTV set-top-boxů nebo modemů.

Tato práce se věnuje obecnému popisu služby provisioning a její implementaci pro koncová zařízení sloužící k VoIP telefonii do Open-source PBX ústředny Asterisk.

Práce je rozdělena na dvě hlavní kapitoly: v první části se věnuje možnostem implementace služby provisioning do PBX Asterisk a nastiňuje konkrétní scénář samotné implementace, včetně požadavků na realizaci nástroje pro automatickou správu koncových zařízení. Jsou zde rovněž obecně popsány možnosti koncových zařízení sloužících pro VoIP telefonii. V závěru kapitoly jsou potom rozebrány jednotlivé protokoly a služby, které jsou k implementaci služby pro popsaný scénář nutné.

Druhá část práce se již věnuje ověření scénáře nastíněného v první části práce v praxi. Postupně je zde rozebráno základní nastavení serveru, na kterém je spuštěna ústředna i všechny popsané služby nutné ke správné funkci provisioningu. Dále je v této části popsáno nastavení samotné softwarové ústředny Asterisk, které je nutné pro zprovoznění hlasové služby na koncových zařízeních a základní nastavení dostupných koncových zařízení, tj. hardwarových VoIP telefonů Linksys SPA921 a Well 3130IF. Poslední část kapitoly je věnována rozboru navržené a vytvořené aplikace (nástroje) pro správu koncových zařízení.

[1] Spojení televizních, internetových a hlasových služeb

1 ŘEŠENÍ STUDENTSKÉ PRÁCE

1.1 Úvod kapitoly

První část této kapitoly se zabývá obecnými možnostmi implementace služby provisioning v PBX Asterisk. Podle dostupné dokumentace je nastíněn možný scénář implementace se zaměřením na dostupná koncová zařízení, tj. zařízení od výrobců Well a Linksys. Dále jsou v první části diskutovány možnosti konfigurace VoIP telefonu Well.

Ve druhé části této kapitoly je nastíněn scénář implementace této služby s ohledem na použití nástroje pro automatickou konfiguraci a správu telefonů, což obnáší zejména propojení nástroje a PBX s databázovým systémem. Rovněž zde budou popsány požadavky na realizovaný nástroj (aplikaci).

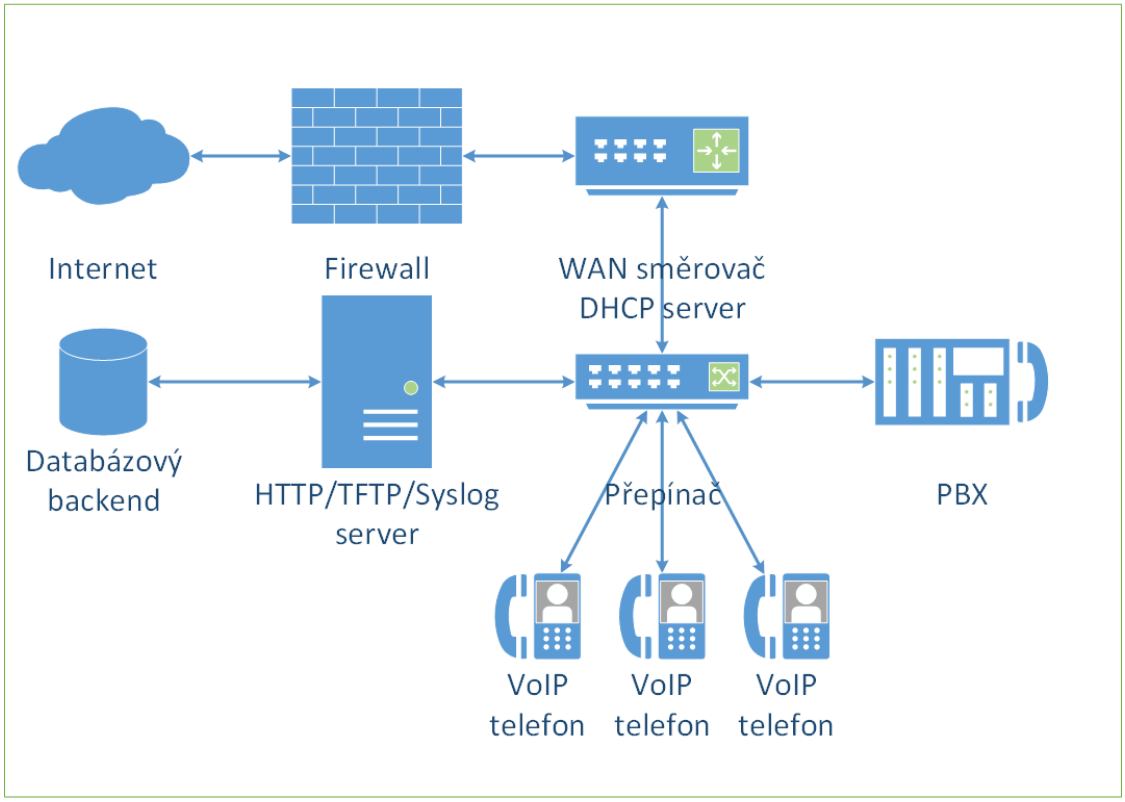
Třetí část této kapitoly stručně rozebírá protokoly, které jsou nutné k implementaci služby provisioning v uvažovaném scénáři. Jedná se zejména o protokoly DHCP, TFTP, HTTP a Syslog a dále také služby PHP, MySQL a ODBC, které byly použity pro realizaci nástroje umožňujícího správu koncových zařízení.

1.2 Možnosti implementace služby provisioning do PBX Asterisk a realizace nástroje pro správu koncových zařízení

Současné verze PBX Asterisk podporují implementaci služby provisioning pomocí modulu *res_phoneprov*. Tato implementace je založena na systému šablon pro jednotlivá koncová zařízení. Z těchto šablon jsou poté nahrazením proměnných v nich obsažených dynamicky vygenerovány konfigurační soubory, které již obsahují hodnoty parametrů doplněné ze souborů obsahujících nastavení jednotlivých uživatelů a zařízení *phoneprov.conf* a *users.conf* vždy pro konkrétní zařízení. Výchozí instalace PBX Asterisk obsahuje předpřipravené šablony pro automatickou konfiguraci VoIP telefonů značky Polycom, šablony pro ostatní výrobce a jednotlivé typy telefonů je tedy nutné dodatečně doplnit. Aktuální implementace provisioningu v PBX Asterisk je omezena na provisioning jednoho uživatele pro jedno zařízení. [1]

Konkrétní způsob implementace služby provisioning je do jisté míry závislý na možnostech koncových zařízení, které je nutné nakonfigurovat. Obecně platí, že pro správnou funkci služby a snížení nutnosti manuálních zásahů do konfigurace koncových zařízení na minimum je zapotřebí v síti nastavit několik dalších služeb, například DHCP, TFTP, HTTP a Syslog servery, opět podle možností jednotlivých

koncových zařízení. Na obrázku 1.1 je nastíněna obecná topologie sítě a všech zmíněných služeb.



Obr. 1.1: Obecná topologie sítě nutná pro správnou funkci služby provisioning

Pokud budeme uvažovat případ, kdy použijeme PBX Asterisk pouze v rámci lokální sítě (např. jedna pobočka organizace), situace se zjednoduší – protože Asterisk je softwarová ústředna, je možné všechny požadované služby i samotnou PBX umístit na jeden fyzický server. Chceme-li automaticky generovat konfigurační soubory pro jednotlivá koncová zařízení podle údajů uživatelů z Asterisku, je nutné využít vestavěný HTTP server, který právě Asterisk poskytuje. Ostatní služby (DHCP, TFTP, Syslog) je nutné zřídit externě. [3]

1.2.1 Implementace služby provisioning s ohledem na externí nástroje

V případě, že chceme provázat PBX Asterisk s externími nástroji, jeví se jako nejjednodušší řešení použití sdíleného databázového backendu, který bude obsahovat všechna potřebná data pro jednotlivé nástroje a jejich vzájemnou interakci.

PBX Asterisk tento scénář umožňuje poměrně jednoduše implementovat pomocí rozhraní nazvaného *Asterisk Realtime Architecture (ARA)*. Pomocí rozhraní *ARA*

lze ukládat konfigurační soubory a nastavení Asterisku do tabulky databáze namísto textových souborů. Ke konfiguraci uložené v databázi lze poté pohodlně přistupovat (případně ji i měnit) různými způsoby, například pomocí PHP skriptů.

ARA rozlišuje dva typy konfiguračních souborů: statické a dynamické. Načítání statických konfiguračních souborů se příliš neliší od načítání klasických textových souborů uložených nejčastěji v adresáři *etc/asterisk/* – pouze s tím rozdílem, že data jsou načítána z databázové tabulky. Při provedení jakékoliv změny ve statické konfiguraci je pro její aplikování nutné opětovné načtení (*reload*) konfigurace. Naproti tomu dynamické konfigurační soubory je možné měnit za provozu, protože PBX z nich načítá data podle potřeby – jedná se například o ekvivalent souboru *sip.conf*, kde je uloženo nastavení účastníků SIP (*users* a *peers*). [3] [4]

1.2.2 Možnosti koncových zařízení

Jak již bylo zmíněno v úvodu této části, implementace služby provisioning je také závislá na možnostech samotných koncových zařízení. To se týká nejenom nastavení automatické konfigurace (některá starší zařízení například nemají možnost stahovat soubory z HTTP serveru), ale zejména různých parametrů, které se na koncových zařízeních dají nastavit. Praktické ukázky, obsažené v této práci, obsahují pouze to nejzákladnější nastavení, které je nutné pro provádění hovorů z koncových zařízení, tj. zejména nastavení protokolu SIP. Možnosti koncových zařízení bývají ovšem značně rozsáhlé – u telefonu Well 3130IF, používaného v ukázkách, lze měnit například:

- globální nastavení (konfigurace sítě, časové zóny)
- nastavení vestavěného LAN portu a DHCP serveru
- nastavení telefonních linek
- nastavení preferovaných VoIP kodeků
- nastavení protokolu SIP
- nastavení protokolu IAX
- nastavení vzdáleného přístupu k telefonu (protokol telnet)
- nastavení kvality služeb (QoS)
- nastavení pro ladění chyb
- uživatelské nastavení (text displeje, tlačítka rychlé volby)

Souhrn nastavitelných parametrů pro telefony používané v praktické části lze najít v [5] (telefon Well 3130IF), resp. [6] (telefon Linksys SPA921).

Obr. 1.2: Použitá koncová zařízení pro VoIP telefonii[1]

1.2.3 Realizace nástroje pro správu koncových zařízení

Jedním z cílů této práce bylo také realizovat nástroj, který by umožňoval centralizovanou správu všech koncových zařízení pro VoIP telefonii dostupných v laboratoři. Nástroj by měl mít především tyto možnosti:

- centralizovaně spravovat všechna dostupná koncová zařízení
- aktivovat/deaktivovat automatickou synchronizaci koncových zařízení podle potřeby
- jednoduše a rychle přidávat, měnit a odebírat registrační údaje pro jednotlivá koncová zařízení připojená k PBX
- přidávat nová koncová zařízení pod správu bez nutnosti ručního zásahu do konfigurace těchto zařízení

Samotný nástroj bude realizován formou webové aplikace ve skriptovacím jazyku PHP, zejména kvůli jeho velké rozšířenosti, dobře dostupné dokumentaci a provázanosti s relačními databázemi, například s databází MySQL – v současnosti se dá většina webových serverů označit za tzv. **LAMP stack**[2], kdy je na serveru (fyzickém či virtuálním) nainstalován balík open-source softwaru, který pro webový server zajišťuje všechny potřebné funkce. Tím bude zajištěna použitelnost nástroje v co největším počtu prostředí. Nastavení potřebné pro správnou funkci nástroje

[1] Obrázky zařízení použity ze stránek distributorů ipmedia.cz a eurosat.cz

je popsáno v kapitole 2.2 a kód realizovaného nástroje je poté podrobně rozebrán v kapitole 2.5.

1.2.4 Scénář implementace

Výstupem této práce by měl být nástroj, umožňující realizovat následující scénář:

Uživatel pomocí webového rozhraní nástroje postupně zadá do databáze informace a parametry všech koncových zařízení, které si přeje pomocí nástroje spravovat. Tyto informace lze kdykoliv zobrazit, upravit nebo odstranit pomocí příslušných akcí provedených ve webovém rozhraní. Stejně tak lze kdykoliv do databáze přidávat další položky.

Po zadání údajů o všech koncových zařízeních může uživatel postupně připojovat jednotlivá koncová zařízení k rozhraní serveru. Nyní mohou nastat dva případy:

a) Koncovým zařízením je telefon Linksys SPA921 (nebo podobný):

Po připojení telefonu v továrním nastavení k rozhraní serveru telefon nejprve získá IP adresu a ostatní nastavení sítě z DHCP serveru. Zároveň s těmito údaji je telefonu pomocí *option 66* předána i adresa TFTP serveru. Telefon se po restartu dotáže TFTP serveru na existenci výchozího konfiguračního souboru. Pokud žádaný soubor na serveru existuje, telefon jej stáhne, uloží do paměti a restartuje se. Po opětovném spuštění telefon posílá žádost o stažení úplného konfiguračního souboru na HTTP server. Pokud jsou pro telefon v databázi uloženy patřičné informace, server na jejich základě vygeneruje pro telefon konfigurační soubor. Po získání a uložení tohoto souboru má již telefon všechny potřebné informace pro registraci k PBX a tedy tak provede. Telefon je nyní nastaven a je možné z něj provádět hovory. Telefon se poté bude v pravidelných intervalech (nebo při selhání registrace k PBX) znovu synchronizovat s úplným konfiguračním souborem.

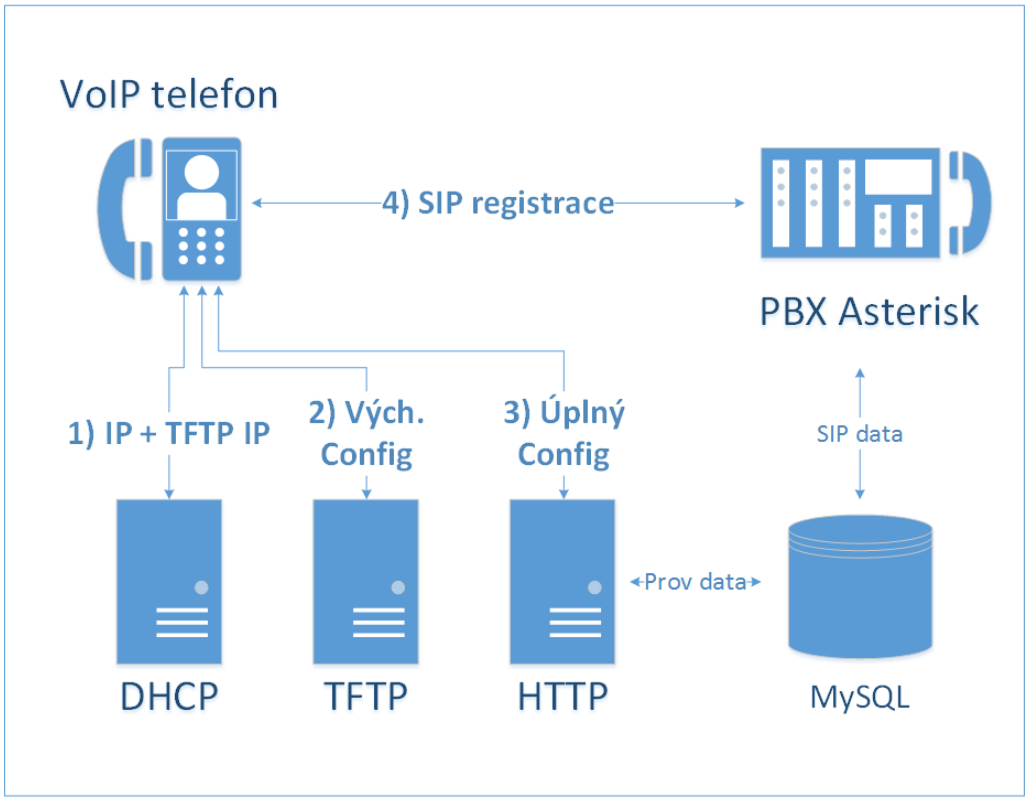
b) Koncovým zařízením je telefon Well 3130IF (nebo podobný):

Po připojení telefonu k rozhraní serveru telefon získá IP adresu a ostatní nastavení sítě z DHCP serveru. Protože je ve výchozí konfiguraci telefonu zakázána služba provisioning, uživatel musí telefon ručně zesynchronizovat s výchozím konfiguračním souborem – to může provést pomocí webového rozhraní telefonu, připojením k telefonu protokolem Telnet, nebo pomocí součásti *Sync Tool* webového nástroje. Po stažení a uložení konfiguračního souboru do paměti telefonu se telefon restartuje a dotazuje se HTTP serveru na úplný konfigurační soubor. Pokud jsou v databázi uvedeny všechny patřičné informace pro tento telefon, server na jejich základě vygeneruje pro telefon konfigurační

[2] Zkratka sestavená z prvních písmen názvů open-source softwaru, nejčastěji operační systém **L**inux, webový server **A**pache, relační databáze **M**ySQL a skriptovací jazyk **P**HP.

soubor. Po získání tohoto souboru a uplatnění nastavení v něm obsažených má již telefon všechny potřebné informace pro registraci k PBX a tedy tak provede. Telefon je nyní nastaven a je možné z něj provádět hovory. Telefon se poté bude v pravidelných intervalech (nebo při selhání registrace k PBX) znovu synchronizovat s úplným konfiguračním souborem.

Všechny telefony připojené k serveru by se měly výše popsaným postupem automaticky nastavit na zadanou konfiguraci. Uživatel dále kdykoliv může pomocí zmíněné součásti webového nástroje *Sync Tool* vyhledávat nové telefony připojené k rozhraní serveru podle IP adresy (nebo rozsahu IP adres) a případně jeden nebo více telefonů synchronizovat s konfiguračními soubory. V nástroji jsou také zahrnuty částečné konfigurační soubory, pomocí kterých je možné u telefonů zakázat či povolit funkci automatické synchronizace. To může být vhodné například v případě, že chce uživatel telefony nastavovat ručně. Telefony je poté možné i v případě jakékoliv změny nastavení zesynchronizovat zpět na výchozí nastavení a tím veškeré ručně provedené změny vymazat. Grafické znázornění scénáře (platí pro automatické nastavení telefonu bez zásahu uživatele) je uvedeno na obrázku 1.3.



Obr. 1.3: Průběh automatické konfigurace koncového zařízení v uvažovaném scénáři

1.3 Použité protokoly a služby

Samotná implementace služby provisioning pro koncová zařízení popisovaná v předchozí části je závislá na několika síťových protokolech a dalších službách serveru, které budou nyní ve stručnosti popsány.

1.3.1 Protokol DHCP

Protokol DHCP (Dynamic Host Configuration Protocol, protokol pro dynamickou konfiguraci zařízení) je protokolem aplikační vrstvy a slouží k poskytování konfiguračních parametrů zařízením v IP sítích. Skládá se z protokolu pro předání konfiguračních parametrů od serveru ke konkrétnímu zařízení a mechanismu pro přidělování síťových adres zařízením. DHCP pracuje na principu klient-server, kdy zařízení určená jako DHCP servery přidělují síťové adresy a poskytují konfigurační parametry dynamicky konfigurovaným zařízením (klientům). Protokol DHCP využívá na transportní vrstvě protokol UDP s porty 67 na straně serveru a 68 na straně klienta. [7]

Kromě základních parametrů nutných pro komunikaci pomocí protokolu IP (IP adresa, maska podsítě, výchozí brána) je možné pomocí protokolu DHCP prostřednictvím tzv. „options“ (možností) zařízením sdělit i další údaje – např. adresu TFTP serveru (option 66), nebo umístění spouštěcího souboru (option 67). [8]

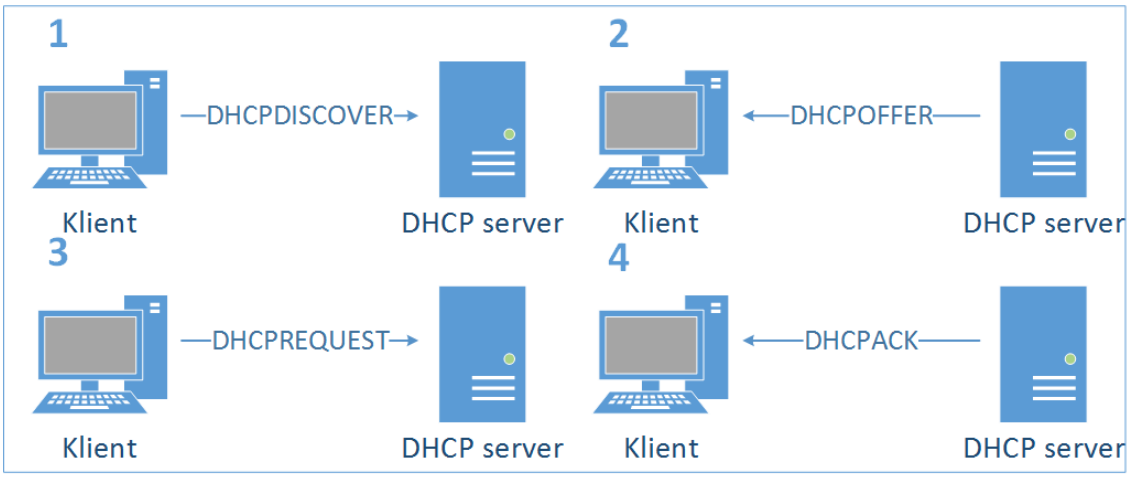
Komunikace DHCP protokolu probíhá obecně ve čtyřech fázích:

1. DHCPDISCOVER – klient v této fázi zasílá na všesměrovou adresu podsítě zprávy, aby objevil dostupné DHCP servery.
2. DHCPOFFER – server odpovídá na žádost klienta zprávou adresovanou na MAC adresu klienta, obsahující IP adresu, masku podsítě, dobu zapůjčení adresy a IP adresu DHCP serveru.
3. DHCPREQUEST – klient vysílá všesměrovou zprávu s žádostí o přidělení nabízené IP adresy.
4. DHCPACK – server odpovídá klientovi, že žádost potvrzuje. Tato zpráva navíc obsahuje i další konfigurační parametry (již zmíněné options), pokud si je klient vyžádal.

Grafické znázornění této komunikace je na obrázku 1.4.

1.3.2 Protokol TFTP

Protokol TFTP (Trivial File Transfer Protocol, protokol pro jednoduchý přenos dat) je jednoduchý protokol aplikační vrstvy sloužící pro přenos dat. Tento protokol byl navržen s cílem jednoduchosti implementace, proto obsahuje pouze velmi omezenou funkcionalitu. Na rozdíl od protokolu FTP neumožňuje autentizaci uživatelů ani výpis seznamu adresářů. Nabízí pouze funkci čtení nebo zápisu dat ze serveru.



Obr. 1.4: Proces výměny zpráv při konfiguraci zařízení pomocí protokolu DHCP

Vzhledem ke své jednoduchosti je často využíván pro přenos konfiguračních nebo spouštěcích souborů v rámci místních sítí. [9]

Protokol TFTP využívá na transportní vrstvě protokol UDP. Přenosy fungují následujícím způsobem:

1. Zařízení zašle paket obsahující žádost o čtení (RRQ) nebo zápis dat (WRQ), název souboru a mód přenosu na UDP port 69 serveru.
2. Server odpoví datovým paketem (na žádost RRQ) nebo potvrzujícím paketem (na žádost WRQ). Tento paket je odeslán z náhodně vygenerovaného portu, na kterém bude probíhat zbytek komunikace.
3. Zařízení, které vlastní zdrojový soubor, zasílá svému partnerovi datové pakety o velikosti 512 bajtů. Partner odpovídá číslovanými pakety ACK s potvrzením.
4. Datový paket, který obsahuje méně než 512 bajtů dat, je považován za finální a přenos je po jeho potvrzení ukončen.

1.3.3 Protokol HTTP

Protokol HTTP (HyperText Transfer Protocol, protokol pro přenos hypertextu) je obecný bezestavový protokol aplikační vrstvy využívaný distribuovanými informačními systémy a slouží primárně pro přenos hypertextových dokumentů. Tento protokol je založený na systému klient-server a využívá systém žádostí a odpovědí.

Komunikace je zahájena HTTP klientem, který sestaví TCP spojení na port serveru (typicky port 80) a odešle serveru žádost (například žádost GET požadující prostředek s určitým URL umístěný na serveru). HTTP server, který na tomto portu naslouchá, po přijetí žádosti odpoví stavovým kódem (např. 200 OK pro úspěšně

splněné HTTP požadavky) a vlastní zprávou, jejíž tělo většinou obsahuje prostředek požadovaný klientem, případně chybovou zprávu nebo jinou informaci. [10]

Číselné stavové kódy jsou rozděleny do následujících skupin:

- 1xx: Informační kódy
- 2xx: Kódy informující o úspěšném zpracování žádosti
- 3xx: Kódy informující o přesměrování, tj. o nutnosti dodatečné akce ze strany klienta
- 4xx: Kódy informující o chybě na straně klienta (např. neplatná žádost)
- 5xx: Kódy informující o chybě na straně serveru (např. interní chyba)

1.3.4 Protokol Syslog

Protokol Syslog slouží ke generování, přenosu a záznamu informačních zpráv o systémových událostech. Je rozdělen na tři vrstvy:

- Syslog content (obsah) – informace obsažená v syslog zprávě.
- Syslog application (aplikace) – na této vrstvě dochází k vytváření, interpretaci a skladování syslog zpráv.
- Syslog transport (přenos) – tato vrstva se stará o předávání a odebírání syslog zpráv do a z transportního protokolu.

Na vrstvě „aplikace“ jsou dále rozlišovány tři druhy zařízení – *originator* (původce), který generuje obsah syslog zpráv; *collector* (sběrač), který zprávy shromažďuje k další interpretaci; *relay* (přenašeč), který přijímá zprávy od původců nebo dalších přenašečů a předává je sběračům nebo opět dalším přenašečům.

Syslog dále rozlišuje, která část systému zprávu vygenerovala (*Facility level*) a závažnost zprávy (*Severity level*). Samotná syslog zpráva obsahuje tři rozdílné části:

1. Priorita (*PRI*) – číslo generované z hodnot *Facility level* a *Severity level*, vyjadřuje prioritu zprávy.
2. Hlavička (*HEADER*) – obsahuje čas, kdy byla zpráva vygenerována (*Timestamp*) a název nebo IP adresu zařízení, které zprávu vygenerovalo.
3. Tělo zprávy (*MSG*) – obsahuje dodatečné informace o události.

Protokol Syslog žádným způsobem nekontroluje, zda byla vygenerovaná zpráva v pořádku doručena. Transport zajišťují protokoly TCP nebo UDP, používá port 514. [11]

1.3.5 MySQL

MySQL je populární Open Source systém pro správu SQL databáze. Je vyvíjen, distribuován a podporován společností Oracle Corporation. Databáze obecně je struk-

turovaná sada dat. Pro přístup k těmto datům a jejich zpracování je nutné použít systém pro správu databáze, kterým právě MySQL server je. MySQL je zástupcem tzv. relační databáze, což znamená, že data jsou ukládána v oddělených databázových tabulkách. Samotné databázové struktury jsou organizovány do logického modelu s objekty jako jsou databáze, tabulky, pohledy, řádky a sloupce, což umožňuje rychlý a flexibilní přístup k požadovaným datům. Relační databáze dále mezi jednotlivými datovými položkami definuje a udržuje různé vztahy, což zamezuje nekonzistentnosti, duplikování, osamocení nebo ztrátě samotných dat.

Pro přístup k datům se používá standardizovaný jazyk SQL (*Structured Query Language*, strukturovaný dotazovací jazyk), který na základě požadavků (dotazů) uživatele provádí manipulaci s daty. [12]

1.3.6 ODBC

ODBC (*Open DataBase Connectivity*) je standard pro přístup k datům v rozličném prostředí systémů pro správu relačních a jiných databází. ODBC poskytuje možnost otevřeného a neutrálního přístupu, nezávislého na výrobci, k datům uloženým v různých databázových systémech. ODBC rozlišuje tři druhy komponentů:

- ODBC Client
- ODBC Driver
- ODBC Server

ODBC klient je front-endová aplikace, která používá jazyk nebo slovník příkazů pro získávání nebo vkládání dat do back-endového systému pro správu databáze. Tento systém ovšem požadavkům aplikace nerozumí, pokud požadavek neprojde přes ODBC Driver (ovladač). ODBC Driver se stará o překlad požadavků a odpovědí mezi ODBC klientem a ODBC serverem tak, aby si tyto dvě komponenty navzájem rozuměly. Tento přístup je výhodný, protože aplikace nemusí mít implementována rozhraní pro přístup k různým druhům databázových systémů, ale stačí jim pouze rozhraní ODBC klienta. Pokud pro konkrétní databázový systém existuje ODBC Driver, jakákoliv aplikace podporující ODBC k němu může bez problémů přistupovat. [13]

1.3.7 PHP

PHP, celým názvem *PHP: Hypertext Preprocessor*, je v dnešní době velmi rozšířeným Open Source skriptovacím jazykem, který je zaměřen zejména na vývoj webových stránek a je úzce propojen s jazykem HTML. Jeho syntaxe má kořeny v jazycích C, Java a Perl. Jedním z hlavních cílů PHP je umožnit vývojářům webových stránek rychlé a pohodlné vytváření dynamických webových stránek.

Původní implementaci PHP vytvořil v roce 1994 Rasmus Lerdorf, kdy se jednalo o jednoduchou sadu binárních souborů CGI napsaných v jazyce C. Po uvolnění zdrojového kódu veřejnosti v roce 1995 si PHP postupně získávalo stále větší množství příznivců, zejména kvůli jeho syntaxi, připomínající jazyk C, která jej činila jednodušším pro použití vývojáři jazyků C, Perl a jim podobných. Během let bylo jádro PHP několikrát přepsáno a od verze 3.0, vydané v roce 1997, je velmi podobné současné verzi. Ta nyní nese číslo 5.0, byla vydána v roce 2004 a jejím jádrem je „*Zend Engine 2.0*“, obsahující nový objektový model a mnoho nových prvků a funkcí. [14] [15]

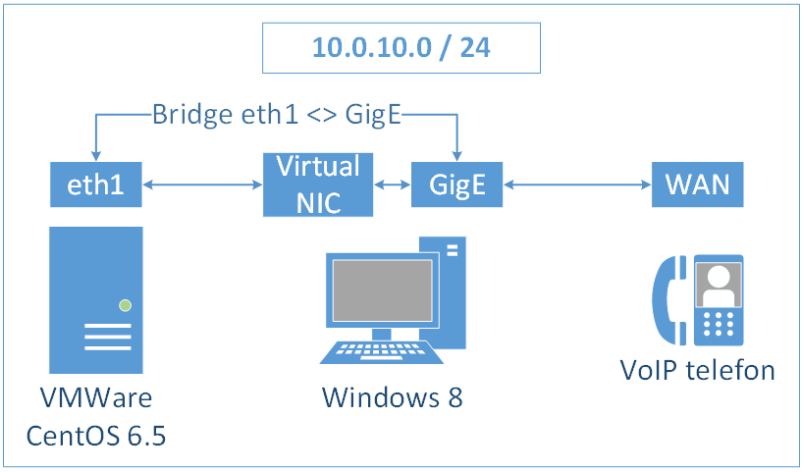
2 VÝSLEDKY STUDENTSKÉ PRÁCE

2.1 Úvod kapitoly

V této části práce je popsán postup instalace a nastavení všech částí serveru, softwarové ústředny Asterisk a parametrů samotných telefonů, který je nutný k automatickému nakonfigurování telefonů tak, aby je bylo možné po připojení k serveru/PBX ihned používat. Dále budou také rozebrány konfigurační soubory pro obě použitá koncová zařízení VoIP (telefony Linksys SPA921 a Well 3130IF) a stejně tak i navržená aplikace pro správu koncových zařízení. Postupně budou popsány následující prvky:

- Server
  - Síťové rozhraní serveru
  - TFTP, DHCP a Syslog server
  - HTTP server Apache
  - MySQL backend a ODBC
- PBX Asterisk
  - Stažení, instalace a kompilace
  - Číslovací plán (soubor extensions.conf)
  - Databázové připojení přes vrstvu ODBC (soubor res_odbc.conf)
  - Rozhraní Asterisk Realtime Architecture (soubor extconfig.conf)
- Konfigurační soubory koncových zařízení pro VoIP telefonii
  - Konfigurační soubory pro telefon Linksys SPA921
  - Konfigurační soubory pro telefon Well 3130IF

V této části práce je uvažována následující topologie (obr. 2.1):



Obr. 2.1: Topologie uvažovaná v příkladech

2.2 Nastavení serveru

2.2.1 Síťové rozhraní

Základním krokem je nastavení síťového rozhraní serveru, ke kterému bude telefon připojen. V operačním systému CentOS, který je v tomto příkladu používán, lze nastavení měnit buď spuštěním nástroje *setup* z terminálu, přes grafické rozhraní nebo přímo ruční úpravou souboru */etc/sysconfig/network-scripts/ifcfg-ethX*, kde *X* je číslo požadovaného rozhraní. Nejdůležitější nastavení je uvedeno v ukázce 2.1

Ukázka 2.1: Nastavení síťového rozhraní *eth0*.

```
DEVICE=eth0
TYPE=Ethernet
ONBOOT=yes
IPADDR=10.0.10.1
NETMASK=255.255.255.0
```

2.2.2 TFTP server

Dalším krokem je instalace a nastavení TFTP serveru, ze kterého si telefony budou moci stahovat prvotní konfigurační soubory. Pokud ještě TFTP server není nainstalovaný, provedeme jeho instalaci zadáním příkazů uvedených v ukázce 2.2 do terminálu. Dále je ještě nutné upravit nastavení TFTP serveru v souboru */etc/xinetd.d/tftp* viz ukázka 2.3

Ukázka 2.2: Instalace TFTP serveru.

```
[root@localhost ~]yum install -y tftp-server
[root@localhost ~]adduser tftpd
[root@localhost ~]chown tftpd:tftpd /var/lib/tftpboot
[root@localhost ~]chkconfig xinetd on
```

Ukázka 2.3: Nastavení TFTP serveru.

```
disable = no
server_args = -c -p -u tftpd -U 117 -s /var/lib/tftpboot
#
```

Poté je již možné TFTP server spustit zadáním příkazu *service xinetd start*. TFTP Server by měl nyní být dostupný na UDP portu 69 a mělo by být možné z něj stahovat soubory umístěné v adresáři */var/lib/tftpboot*.

2.2.3 DHCP server

Dále je nutné nastavit DHCP server tak, aby připojenému telefonu přidělil IP adresu a informoval jej o existenci TFTP serveru v síti, ze kterého je možné stáhnout prvotní konfigurační soubor. DHCP server lze nainstalovat z terminálu zadáním příkazu *yum install -y dhcp*. Před jeho spuštěním je nutné v souboru */etc/sysconfig/dhcpd* nastavit upřednostňované rozhraní, na kterém bude DHCP server naslouchat příchozím žádostem (ukázka 2.4) a dále v souboru */etc/dhcp/dhcpd.conf* určit síť, rozsah přidělovaných adres a další, viz ukázka 2.5.

Ukázka 2.4: Nastavení upřednostňovaného rozhraní v souboru */etc/sysconfig/dhcpd*.

```
DHCPDARGS=eth0
```

Ukázka 2.5: Nastavení parametrů DHCP serveru.

```
default-lease-time 600;
max-lease-time 7200;
log-facility local6;
authoritative;

class "phone-spa"’ {
    match if substring (hardware, 1, 3) = 00:0e:08;
}
class "phone-well" {
    match if substring (hardware, 1, 3) = 00:09:45;
}
subnet 10.0.10.0 netmask 255.255.255.0 {
pool {
    allow members of "phone-spa";
    range 10.0.10.110 10.0.10.119;
  }
  pool {
    allow members of "phone-well";
    range 10.0.10.100 10.0.10.109;
  }
  pool {
    deny members of "phone-spa";
    deny members of "phone-well";
    range 10.0.10.0 10.0.10.99;
    range 10.0.10.120 10.0.10.254;
  }
  option broadcast-address 10.0.10.255;
  option tftp-server-name "10.0.10.1";
}
```

Kromě nastavení výchozí a maximální doby přidělení adres a dalších globálních hodnot pro DHCP server byly ještě vytvořeny dvě třídy, do kterých jsou zařízení žádající o adresu automaticky přiřazena na základě MAC adresy. Podle těchto tříd

poté lze přesně určit, jaké rozsahy adres (v konfiguraci určeny direktivou *pool*) budou těmto zařízením přiřazovány a získat tak lepší přehled a kontrolu.

Poté je již možné server pomocí příkazů v ukázce 2.6 zadaných do terminálu spustit a také zajistit, aby se spouštěl automaticky se startem operačního systému.

Ukázka 2.6: Spuštění DHCP serveru.

```
[root@localhost ~]service dhcpd start
[root@localhost ~]chkconfig dhcpd on
```

2.2.4 Syslog server

Další částí operačního systému, kterou je vhodné nakonfigurovat, je syslog server. Ten umožňuje zaznamenávat informační a případně chybové zprávy z DHCP serveru a také ze samotných telefonů.

Ve výchozí konfiguraci operačního systému CentOS je již syslog server spuštěn, ale není nastaven pro zaznamenávání zpráv vysílaných v síti. To lze napravit odkomentováním následujících řádků v souboru */etc/rsyslog.conf*, viz ukázka 2.7:

Ukázka 2.7: Nastavení syslog serveru pro záznam zpráv vysílaných v síti.

```
# Provides UDP syslog reception
$ModLoad imudp
$UDPServerRun 514

# Provides TCP syslog reception
$ModLoad imtcp
$InputTCPServerRun 514
```

Pro větší přehlednost je vhodné pro syslog server vytvořit šablonu, která zajistí ukládání zpráv pro jednotlivá zařízení do adresářové struktury rozdělené podle data. Toho lze dosáhnout vytvořením souboru */etc/rsyslog.d/remote-hosts.conf* s obsahem podle ukázky 2.8.

Ukázka 2.8: Šablona pro syslog server.

```
$template
DailyPerHostLogs,
"/var/log/syslog/%$YEAR%/%$MONTH%/%$DAY%/%HOSTNAME%_messages.log"
*.info;mail.none;authpriv.none;cron.none -?DailyPerHostLogs
```

Nyní je ještě nutné vytvořit adresář, do kterého budou protokoly ukládány a také syslog server restartovat, jak je možné vidět na ukázce 2.9.

Ukázka 2.9: Poslední krok nastavení syslog serveru.

```
[root@localhost ~]# mkdir /var/log/syslog
[root@localhost ~]# service rsyslog restart
Shutting down system logger:                               [  OK  ]
Starting system logger$:                                   [  OK  ]
[root@localhost ~]#
```

2.2.5 HTTP server

Dalším nutným krokem je instalace a nastavení HTTP serveru a podpory skriptovacího jazyka PHP. V této práci byl použit zřejmě nejrozšířenější HTTP server Apache ve verzi 2.2.15 a PHP ve verzi 5.3.3. Pokud server nebo podpora PHP již není v operačním systému nainstalována, je tak možné učinit následující sadou příkazů (ukázka 2.10):

Ukázka 2.10: Instalace HTTP serveru Apache a PHP v CentOS.

```
[root@localhost ~]# yum -y install httpd
[root@localhost ~]# yum -y install php
[root@localhost ~]# yum -y install php-pdo
[root@localhost ~]# yum -y install php-mysql
[root@localhost ~]# yum -y install php-mbstring
[root@localhost ~]# chkconfig httpd on
```

Poté by již server s podporou PHP měl být nainstalován a spuštěn. Příkaz *chkconfig* poté zajistí, aby se server spouštěl automaticky po startu operačního systému. Nyní je ještě potřeba HTTP server nakonfigurovat, což lze provést editací souboru s nastavením, typicky v adresáři */etc/httpd/conf/httpd.conf*. Na ukázce 2.11 jsou potom všechny direktivy (položky nastavení), které je nutné pro správnou funkci nastavit. Výchozí nastavení PHP nutné měnit není.

Samotný soubor *httpd.conf* obsahuje velké množství řádků (zejména komentářů-dokumentace) a parametrů (direktiv). Většinu jich lze ponechat na výchozích hodnotách, nicméně některé je potřeba nastavit tak, jak je uvedeno na ukázce 2.11. Tyto direktivy jsou ve stručnosti popsány níže.

a) Direktiva *Listen*:

Tato direktiva nastavuje IP adresu a port, na kterém má HTTP server naslouchat. Ve výchozím nastavení je zde uveden pouze port 80.

b) Direktivy *User* a *Group*:

Tato direktiva určuje, pod jakým uživatelem a skupinou se má HTTP server v operačním systému spouštět. Pro HTTP démona byl vytvořen a vyhrazen účet a skupina *apache*.

c) Direktivy *RewriteLog* a *RewriteLogLevel*:

Tyto direktivy nejsou povinné, nastavují úroveň protokolování a protokolovací

Ukázka 2.11: Nastavení HTTP serveru v souboru httpd.conf.

```
Listen 10.0.10.1:80
#<...>
User apache
Group apache
#<...>
RewriteLog "/var/log/apache/rewrite.log"
RewriteLogLevel 4
#<...>
ServerName 10.0.10.1:80
#<...>
DocumentRoot "/var/www/html"
#<...>
<Directory "var/www/html/prov">
  Options Indexes FollowSymLinks
  AllowOverride All
  RewriteEngine on
  RewriteRule ^spa921-([^/]+)\.cfg spa921.php?mac=$1 [L]
  RewriteRule ^([0-9a-f]{12})$ well3130if.php?mac=$1 [L]
#$
</Directory>
```

soubor pro komponent *Rewrite Engine*, který se používá pro přesměrování HTTP žádostí.

d) Direktiva *DocumentRoot*:

Tato direktiva určuje cestu ke kořenovému adresáři *(/)* HTTP serveru na souborovém systému fyzického serveru.

e) Direktiva *<Directory "var/www/html/prov»</Directory>*:

Tato direktiva se týká pouze adresáře */var/www/html/prov*, kde jsou uloženy konfigurační soubory pro jednotlivá koncová zařízení VoIP. Obsahuje několik parametrů, které se starají o přesměrování příchozích HTTP žádostí od těchto zařízení. Kromě zapnutí přesměrování (direktivy *AllowOverride All* a *RewriteEngine On*) jsou zde také definována dvě pravidla pro přesměrování (*RewriteRule*). Obě tato pravidla jsou regulární výrazy, oproti kterým jsou příchozí HTTP žádosti testovány na shodu řetězce. První pravidlo v případě, že je konec řetězce URL příchozí HTTP žádosti ve tvaru „*spa921-X.cfg*“, kde „*X*“ je libovolný řetězec znaků neobsahujících lomítko, přepíše toto URL do tvaru „*spa921.php?mac=X*“, kde „*X*“ je ten stejný řetězec znaků. Jinak řečeno, pokud přijde na server dotaz na konfigurační soubor pro telefon Linksys SPA921 s konkrétní MAC adresou, bude tento dotaz přesměrován na PHP soubor obsahující obecný konfigurační soubor s MAC adresou jako parametrem PHP skriptu. Na základě toho je možné z údajů v databázi vygenerovat konfigurační soubor přesně pro tento telefon.

Druhé pravidlo *RewriteRule* má podobnou funkci, ale stará se o přesměrování

žádostí generovaných telefonem Well 3130IF. Opět se testuje URL příchozí žádosti na řetězec znaků, tentokrát na přítomnost dvanácti alfanumerických znaků *a* až *f* a *0* až *9*, což je formát dotazu telefonu na konfigurační soubor pro jeho MAC adresu. Při nalezení shody je žádost opět přesměrována na obecný konfigurační soubor pro tento telefon, do kterého jsou dynamicky doplněny údaje z MySQL databáze podle odpovídajícího parametru PHP skriptu. Obě pravidla jsou ukončena znaky *[L]*, což značí parametr *Last*, tedy že při shodě s jedním z pravidel je URL přepsáno a s dalšími pravidly se již neporovnává.

Tím je nastavení HTTP serveru hotovo, pro jeho uvedení v platnost je ještě nutné HTTP server restartovat. Poté je možné funkčnost nastavení ověřit otevřením adresy 10.0.10.1 ve webovém prohlížeči, kde by se měla nacházet výchozí stránka HTTP serveru Apache. Po umístění souborů aplikace do kořenového adresáře serveru (*/var/www/html/*) a konfiguračních souborů pro jednotlivé telefony do adresáře */var/www/html/prov/* rovněž lze ověřit správnou funkcí přesměrování URL otevřením např. stránky 10.0.10.1/prov/aabbccddeeff. Měl by se zobrazit textový konfigurační soubor, kde by již některé z údajů měly být vygenerovány na základě testovací MAC adresy (aabbccddeeff) předané v URL jako parametr. Správnou funkci přesměrování lze ověřit také v souboru */var/log/apache/rewrite.log*, kam je zaznamenáván výstup komponenty *Rewrite Engine*. Na ukázce 2.12 můžeme vidět průběh procesu přesměrování.

Ukázka 2.12: Protokolový soubor komponenty Rewrite Engine.

```
[]strip per-dir prefix: /var/www/html/prov/00094560ccf2->00094560ccf2
[]applying pattern '^([0-9a-f]{12})$' to uri '00094560ccf2'
[]rewrite '00094560ccf2' -> 'well3130if.php?mac=00094560ccf2'
[]split uri=well3130if.php?mac=00094560ccf2 -> uri=well3130if.php, args=mac=00094560ccf2
[]add per-dir prefix: well3130if.php->/var/www/html/prov/well3130if.php
[$]
```

2.2.6 MySQL

Dále je potřeba nainstalovat a nastavit MySQL démona, který bude sloužit jako systém pro správu použité MySQL databáze. Zároveň s tím je nutné vytvořit logickou strukturu, tj. databázi a tabulku, ve které budou uchovávána všechna potřebná data pro provoz aplikace a PBX Asterisk.

MySQL démona, server a další potřebné softwarové balíčky je možné nainstalovat pomocí správce balíčků *yum* sadou příkazů uvedených na ukázce 2.13.

Po instalaci a spuštění MySQL serveru by již mělo být možné se k němu připojit zadáním příkazu „`mysqld -u root`“ do terminálu.

Ukázka 2.13: Instalace MySQL serveru.

```
[root@localhost ~]# yum -y install mysql
[root@localhost ~]# yum -y install mysql-server
[root@localhost ~]# yum -y install php-mysql
[root@localhost ~]# service mysqld start
[root@localhost ~]# chkconfig mysqld on
```

Po připojení ke spuštěné instanci MySQL serveru je nutné do příkazové řádky serveru (uvozené řetězcem `mysql>`) zadat příkazy, uvedené na ukázce 2.14. Tyto příkazy mají za úkol následující:

- nastavit heslo uživateli root
- vytvořit uživatele *asterisk* s heslem
- uplatnit změny v uživatelských účtech
- vytvořit databázi *asterisk*
- přiřadit veškerá oprávnění pro databázi *asterisk* uživateli *asterisk*
- vytvořit tabulku *ast_sipfriends*, její sloupce s datovými typy a indexy.

Ukázka 2.14: Konfigurace MySQL.

```
mysql> UPDATE mysql.user SET Password = PASSWORD('testpwd')
       WHERE User = 'root';
mysql> CREATE USER 'asterisk'@'localhost' IDENTIFIED BY 'asterisk';
mysql> FLUSH PRIVILEGES;
mysql> CREATE DATABASE 'asterisk';
mysql> GRANT ALL ON asterisk.* TO 'asterisk'@'localhost';
mysql>
CREATE TABLE IF NOT EXISTS `ast_sipfriends` (
      `id` int(11) NOT NULL AUTO_INCREMENT,
      `name` varchar(40) NOT NULL,
      `ipaddr` varchar(40) DEFAULT NULL,
      `port` int(5) DEFAULT NULL,
      `regseconds` int(11) DEFAULT NULL,
      `defaultuser` varchar(10) DEFAULT NULL,
      `secret` varchar(40) DEFAULT NULL,
      `fullcontact` varchar(35) DEFAULT NULL,
      `regserver` varchar(20) DEFAULT NULL,
      `useragent` varchar(20) DEFAULT NULL,
      `lastms` int(11) DEFAULT NULL,
      `host` varchar(40) DEFAULT NULL,
      `type` enum('friend','user','peer') DEFAULT NULL,
      `context` varchar(40) DEFAULT NULL,
      `callerid` varchar(40) DEFAULT NULL,
      `macaddr` varchar(40) DEFAULT NULL,
      `phonetype` varchar(40) DEFAULT NULL,
      PRIMARY KEY (`id`),
      UNIQUE KEY `name` (`name`),
      UNIQUE KEY `macaddr` (`macaddr`),
      KEY `ipaddr` (`ipaddr`,`port`),
      KEY `host` (`host`,`port`)
) ENGINE=InnoDB;
```

Tabulka *ast_sipfriends* obsahuje sloupce pro uložení informací o účastníkovi SIP v rámci PBX Asterisk. Všechny tyto sloupce jsou pro správnou funkci rozhraní Asterisk Realtime Architecture nutné. Vyjímkou jsou pouze poslední dva sloupce, *macaddr* a *phonetype*, které slouží pro uložení informací o samotném koncovém zařízení, konkrétně o jeho MAC adrese a typu (SPA921 / Well 3130IF).

2.2.7 ODBC

Posledním krokem je instalace a nastavení vrstvy ODBC, kterou bude PBX Asterisk používat pro připojení k MySQL databázi. Ačkoliv lze PBX Asterisk nastavit pro přímé připojení k MySQL databázi, je výhodnější pro toto připojení použít vrstvu ODBC a připojení provést přes ni. V případě, že bude nutné z nějakého důvodu změnit databázový systém, stačí pouze upravit parametry ODBC připojení bez nutnosti změny nastavení PBX Asterisk.

Podporu pro ODBC lze do systému nainstalovat zadáním sady příkazů uvedených na ukázce 2.15 do terminálu.

Ukázka 2.15: Instalace podpory pro vrstvu ODBC.

```
yum install -y unixODBC unixODBC-devel libtool-ltdl libtool-ltdl-devel
yum install -y mysql-connector-odbc
```

Po instalaci je nutné ještě vrstsvu ODBC nastavit, resp. vytvořit identifikátor připojení, který bude později PBX Asterisk používat. Toho lze docílit vložením řádků uvedených na ukázce 2.16 do souboru */etc/odbc.ini*.

Ukázka 2.16: Vytvoření identifikátoru připojení v ODBC.

```
[asterisk-connector]
Description   = MySQL <> Asterisk
Driver        = MySQL
Database      = asterisk
Servername    = localhost
UserName      = asterisk
Password      = asterisk
```

Dále ověříme přítomnost ovladače pro databázový systém MySQL ve vrstvě ODBC. V souboru */etc/odbcinst.ini* by měla být obdobná definice ovladače jako na ukázce 2.17.

Zbytek konfigurace potřebný pro zprovoznění připojení mezi PBX Asterisk a databází MySQL přes vrstvu ODBC je nutné nastavit v konfiguračních souborech PBX Asterisk a bude tedy popsán v následující kapitole (2.3.3).

Tím je veškeré nastavení samotného serveru hotovo a je možné přejít k instalaci a konfiguraci PBX Asterisk.

Ukázka 2.17: Definice ODBC ovladače pro MySQL.

```
# Driver from the mysql-connector-odbc package
# Setup from the unixODBC package
[MySQL]
Description = ODBC for MySQL
Driver      = /usr/lib/libmyodbc5.so
Setup       = /usr/lib/libodbcmyS.so
Driver64    = /usr/lib64/libmyodbc5.so
Setup64     = /usr/lib64/libodbcmyS.so
FileUsage   = 1
```

2.3 Instalace a nastavení PBX Asterisk

V této části práce bude popsána instalace samotné softwarové ústředny Asterisk a základní nastavení jejích součástí (číslovací plán, nastavení ODBC připojení, nastavení rozhraní ARA). Uvažována bude pouze základní verze Asterisku, bez doplňků, jako jsou grafická webová rozhraní pro správu nebo dodatečné ovladače pro hardwarové prvky ústředny. Dále je předpokládáno, že jsou v operačním systému již nainstalovány všechny balíčky, které jsou vyžadovány pro kompilaci a spuštění Asterisku.

Jak již bylo zmíněno v úvodu, PBX Asterisk v sobě obsahuje nástroj pro provisioning koncových zařízení – modul *res_phoneprov*. Tento nástroj bohužel nepodporuje dynamické načítání dat z databáze, tudíž ho pro uvažovanou implementaci provisioningu nelze použít. Samotné nastavení tohoto nástroje je popsáno formou komentářů v souborech *phoneprov.conf*, *users.conf* a *http.conf*, v této práci ovšem z výše zmíněných důvodů rozebíráno nebude.

2.3.1 Instalace PBX Asterisk

Patrně nejjednodušším a nejrychlejším způsobem instalace softwarové ústředny Asterisk je stažení a následná kompilace jejích zdrojových souborů. Aktuální verze jsou dostupné např. na oficiálních stránkách společnosti Digium, `http://downloads.asterisk.org/pub/telephony/asterisk/` [1]. V této práci byla použita verze 1.8.24.

Prvním krokem je stáhnutí zdrojových kódů v komprimované podobě, které budou následně rozbaleny a zkompilovány. Ukázka 2.18 obsahuje odpovídající sadu příkazů, po jejichž zadání do terminálu budou do složky */usr/src* staženy a poté rozbaleny komprimované zdrojové kódy Asterisku.

Po stažení a rozbalení zdrojových souborů je již možné zdrojové kódy Asterisku zkompilovat, což lze provést sadou příkazů uvedených v ukázce 2.19.

[1] Dostupné k 20.12.2013

Ukázka 2.18: Stažení a rozbalení zdrojových kódů PBX Asterisk.

```
[root@localhost ~]# cd /usr/src
[root@localhost src]# wget http://downloads.asterisk.org/pub/telephony/
                      asterisk/asterisk-1.8-current.tar.gz
[root@localhost src]# tar zxvf asterisk-1.8-current.tar.gz
```

Ukázka 2.19: Kompilace zdrojových kódů Asterisku.

```
[root@localhost src]# cd /usr/src/asterisk-1.8.24.0
[root@localhost asterisk-1.8.24.0]# ./configure
[root@localhost asterisk-1.8.24.0]# make install
[root@localhost asterisk-1.8.24.0]# make samples
[root@localhost asterisk-1.8.24.0]# make config
```

Tyto příkazy nainstalují samotnou PBX Asterisk, dále vzorové konfigurační soubory (příkaz *make samples*) a spustí konfigurační skripty, které zajistí automatické spouštění Asterisku po startu systému (příkaz *make config*). Po restartu systému by měl nyní být Asterisk automaticky spuštěn a mělo by být možné se k běžící instanci procesu přes terminál pomocí příkazu *asterisk -rvvv* připojit, jak je vidět na ukázce 2.20.

Ukázka 2.20: Připojení k procesu Asterisku pomocí terminálu.

```
[root@localhost /]# asterisk -rvvv
Asterisk 1.8.24.0, Copyright (C) 1999 - 2013 Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
<zkráceno>
==========================================================================
Connected to Asterisk 1.8.24.0 currently running on localhost
(pid = 2099)
Verbosity is at least 3
localhost*CLI>
```

Nyní je již možné přistoupit k úpravě konfiguračních souborů Asterisku.

2.3.2 Číslovací plán

Aby bylo možné ověřit funkčnost připojených telefonů, je nutné vytvořit alespoň základní číslovací plán. Číslovací plán v PBX Asterisk určuje, jakým způsobem budou ošetřeny všechny příchozí a odchozí hovory. Jedná se vlastně o seznam jednotlivých instrukcí, které Asterisk postupně plní.

Pro účely testování bude nyní úpravou souboru */etc/asterisk/extensions.conf* vytvořen jednoduchý číslovací plán. Obsah souboru je vidět na ukázce 2.21.

V tomto jednoduchém číslovacím plánu byly nadefinovány dvě části, v Asterisku označované jako *kontexty.* Kontext *[internal]* pro místní hovory v rámci ústředny

Ukázka 2.21: Obsah souboru *extensions.conf*.

```
[general]
; pokud se Asterisk dostane na konec seznamu příkazů pro volané číslo,
; hovor bude automaticky ukončen
autofallthrough=yes

; místní hovory v rámci ústředny
[internal]
; Testovací číslo - po vytočení bude hovor přijat, přehraje se záznam
; a po 10 vteřinách bude hovor ukončen
exten => 9999,1,Answer()
exten => 9999,n,Playback(hello)
exten => 9999,n,Wait(10)
exten => 9999,n,Hangup()

exten => _XXXX,1,Verbose(Dialing SIP peer ${EXTEN})
exten => _XXXX,n,Dial(SIP/${EXTEN},30)
exten => _XXXX,n,Playback(vm-nobodyavail)
exten => _XXXX,n,Hangup()

include => nomatch
; po vytočení neplatného čísla se přehraje záznam a hovor bude poté
; ukončen
[nomatch]
exten => _X.,1,Answer
exten => _X.,n,Wait(1)
exten => _X.,n,Playback(access-denied)
exten => _X.,n,Hangup
```

a kontext *[nomatch]*, který bude uplatněn při vytočení neplatného (neexistujícího) čísla. V kontextu *[internal]* jsou definovány dvě „klapky“ – klapka 9999, po jejímž vytočení Asterisk hovor přijme, přehraje předem nahranou zprávu *hello*, poté počká 10 sekund a hovor ukončí.

Druhá definovaná klapka je dynamická – v případě, že je navolena libovolná čtyrčíselná klapka, Asterisk se pokusí o sestavení SIP spojení na účastníka s touto klapkou. Pokud účastník do 30 vteřin hovor nepřijme, je přehrán záznam o nedostupnosti účastníka *vm-nobodyavail* a spojení se ukončí. Pokud tedy všechny telefony budou používat čísla o čtyřech cifrách, není nutné definovat pro každý telefon položku v číslovacím plánu zvlášť.

2.3.3 Připojení k databázi přes vrstvu ODBC

Nastavení PBX Asterisk pro připojení k databázi přes vrstvu ODBC je poměrně jednoduché – za předpokladu, že byly při instalaci Asterisku zkompilovány všechny potřebné moduly, stačí pouze do souboru */etc/asterisk/res_odbc.conf* vložit následující řádky (ukázka 2.22):

Po uložení změn provedených v souboru a restartování PBX Asterisk můžeme

Ukázka 2.22: Nastavení ODBC připojení v souboru res_odbc.ini.

```
[asterisk]
enabled => yes
dsn => asterisk-connector
username => asterisk
password => asterisk
pooling => no
pre-connect => yes
```

ověřit funkčnost spojení zadáním příkazu `odbc show` do konzole Asterisku. Výstup příkazu by měl být identický s výstupem na ukázce 2.23.

Ukázka 2.23: Ověření funkčnosti nastavení ODBC.

```
localhost*CLI> odbc show

ODBC DSN Settings
-----------------

  Name:   asterisk
  DSN:    asterisk-connector
    Last connection attempt: 2014-04-20 16:00:00
  Pooled: No
  Connected: Yes
```

2.3.4 Nastavení rozhraní Asterisk Realtime Architecture

Posledním krokem konfigurace je nastavení rozhraní *ARA*. Jak již bylo řečeno v úvodu práce, toto rozhraní umožňuje načítat konfigurační soubory z tabulek databáze místo z textových souborů. Samotné nastavení se provádí v souboru */etc/asterisk/extconfig.conf*. V tomto souboru je možné určit, které z dynamických nebo statických konfiguračních souborů se budou načítat z databáze místo z textových souborů. Pro tuto konkrétní implementaci scénáře postačí aktivovat načítání konfigurace z databáze pro soubor *sip.conf*, přičemž jednotlivé položky souboru (účastníci SIP a jejich nastavení) budou do databáze přidávány pomocí webového nástroje. Obsah souboru s relevantním nastavením je zobrazen na ukázce 2.24.

Ukázka 2.24: Obsah souboru *extconfig.conf*.

```
[settings]
sippeers => odbc,asterisk,ast_sipfriends
sipregs => odbc,asterisk,ast_sipfriends
```

Toto nastavení Asterisku říká, že má pro získání konfigurace účastníků SIP použít ovladač ODBC a dříve vytvořenou databázi *asterisk* a tabulku *ast_sipfriends*.

2.4 Konfigurační soubory koncových zařízení

V této části práce budou popsány konfigurační soubory jednotlivých koncových zařízení nutné pro jejich automatické nastavení tak, aby je bylo možné po připojení k ústředně bez dalších zásahů uživatele ihned používat.

2.4.1 Konfigurace VoIP telefonu Linksys SPA921

Protože je po uvedení do továrního nastavení VoIP telefon Linksys SPA921 nastaven tak, aby se po startu dotazoval na konfigurační soubor umístěný na TFTP serveru (v případě, že je tento server v síti dostupný), je nutné pro tento telefon připravit konfigurační soubory dva – základní, který bude sloužit pouze jako odkaz na další soubor a finální, který bude dostupný pomocí HTTP serveru a bude obsahovat na základě MAC adresy zařízení dynamicky vygenerovanou finální konfiguraci pro tento konkrétní telefon.

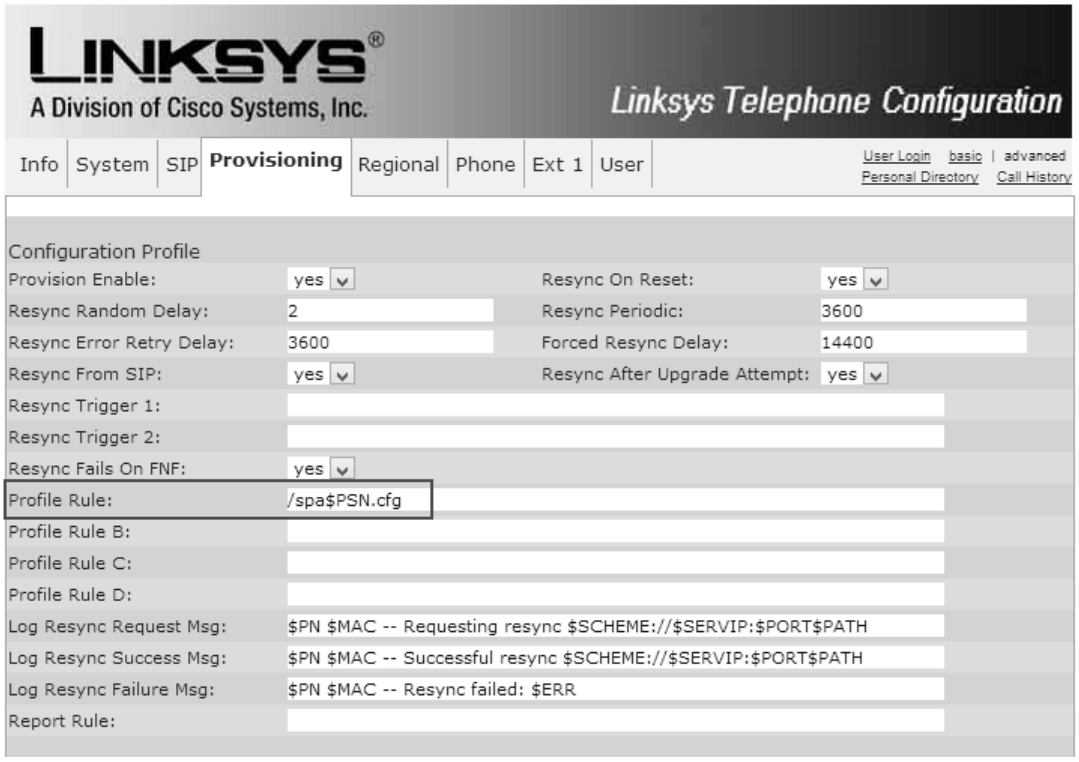
Obsah souboru pro prvotní konfiguraci telefonu je vidět na ukázce 2.25.

Ukázka 2.25: Počáteční konfigurační soubor telefonu SPA921.

```
<flat-profile>
 <Profile_Rule ua="na">
   http://10.0.10.1/prov/spa921-$MAC.cfg
 </Profile_Rule>
 <Syslog_Server ua="na">
   10.0.10.1
 </Syslog_Server>
 <Provision_Enable ua="na">
   Yes
 </Provision_Enable>
 <Resync_On_Reset ua="na">
   Yes
 </Resync_On_Reset>
 <Resync_Trigger_1 ua="na">
   $REGTMR1 gt 60
 </Resync_Trigger_1>
</flat-profile>
```

Ze struktury souboru je možné vidět, že obsahuje XML značky a přiřazuje parametru *Profile_Rule* hodnotu *http://10.0.10.1/prov/spa921-$MAC.cfg*. Parametru *Profile_Rule* je ve výchozím (továrním) nastavení telefonu přiřazena hodnota */spa$PSN.cfg*, jak je vidět na obrázku 2.2.

Tato hodnota způsobí, že telefon se po startu pokusí o stažení konfiguračního souboru s názvem *spa921.cfg* (proměnná $PSN je nahrazena hodnotou podle typu telefonu, v tomto případě hodnotou 921) z TFTP serveru, jehož adresa byla telefonu oznámena při automatickému získávání IP adresy z DHCP serveru.

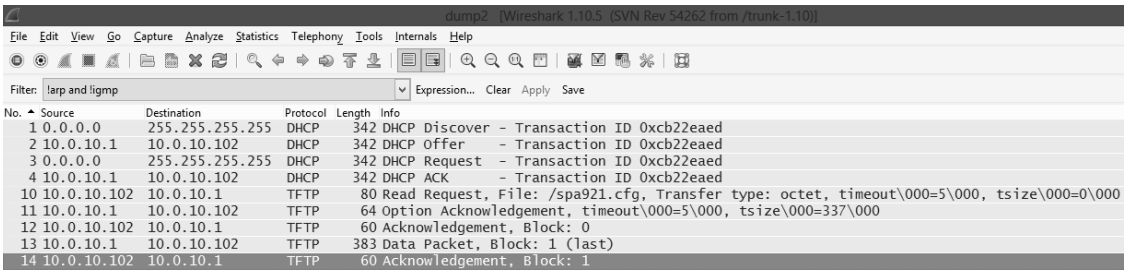


Obr. 2.2: Sekce Provisioning webového rozhraní telefonu SPA921 ve výchozím nastavení

Po stažení telefon upraví svou konfiguraci na nové hodnoty parametrů uvedených v tomto souboru a poté se restartuje. Protože původní hodnota parametru *Profile_Rule* byla nahrazena hodnotou novou, telefon se po restartu již nebude dotazovat na soubor *spa921.cfg* umístěný na TFTP serveru, ale pošle žádost na stažení souboru *spa921-00:0e:08:dd:89:11.cfg* (proměnná $MAC v názvu souboru je nahrazena skutečnou MAC adresou telefonu) který by měl být dostupný na HTTP serveru s URL 10.0.10.1/prov/.

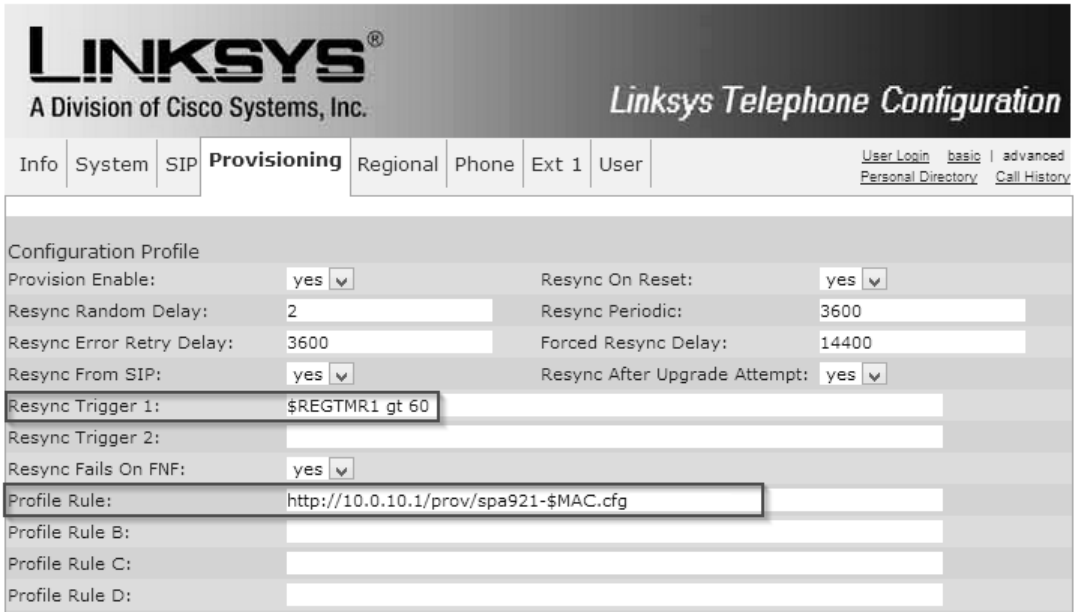
Ostatní parametry uvedené v tomto souboru zajišťují periodický restart a stažení nové konfigurace v případě, že telefon není po dobu delší než 60 sekund registrován k ústředně a určují adresu dříve nastaveného syslog serveru, na který bude telefon zasílat informační zprávy o průběhu konfigurace.

Pokud tedy vložíme obsah ukázky 2.25 do textového souboru *spa921.cfg*, umístíme tento soubor do kořenového adresáře TFTP serveru (*/var/lib/tftpboot*), připojíme k rozhraní serveru VoIP telefon SPA921 uvedený do továrního nastavení a pomocí nástroje na zachytávání síťového provozu (zde byl použit program Wireshark) odposlechneme komunikaci na rozhraní serveru *eth0*, výsledek by měl být stejný, jako na obrázku 2.3.



Obr. 2.3: Provoz zachycený na rozhraní *eth0* serveru při přidělování adres a stahování souboru z TFTP serveru

Na obrázku 2.3 je možné vidět, že po připojení telefonu k rozhraní serveru si telefon nejprve vyžádá od DHCP serveru IP adresu a ostatní nastavení a poté, co mu je adresa přidělena, kontaktuje TFTP server s žádostí o stažení souboru *spa921.cfg* a tento soubor si také stáhne. Pomocí webového rozhraní telefonu je možné ověřit, že stažení a nahrání konfiguračního souboru do paměti v telefonu proběhlo úspěšně. Na obrázku 2.4 jsou zvýrazněny ohraničením parametry, jejichž hodnoty byly po nahrání souboru z TFTP serveru do paměti telefonu změněny.



Obr. 2.4: Sekce Provisioning webového rozhraní telefonu SPA921 po aplikování nových hodnot parametrů

Nyní zbývá pouze připravit finální konfigurační soubor, který bude obsahovat dynamicky vygenerované údaje uživatele přiřazeného ke konkrétnímu zařízení. Samotný obsah konfiguračního souboru je na ukázce 2.26.

Z ukázky je vidět, že na začátku souboru je uveden PHP kód. Tento kód slouží k vygenerování dynamického konfiguračního souboru, respektive k doplnění hodnot parametrů charakteristických pro každý telefon.

Po připojení k MySQL serveru a vybrání databáze *asterisk* tento PHP skript načte z URL, se kterým byl zavolán, parametr *macaddr*, jehož hodnotou je MAC adresa telefonu. MAC adresa telefonu je získána přesměrováním HTTP žádosti při dotazu telefonu na svůj konfigurační soubor, kdy je URL původního dotazu přepsáno tak, aby obsahovalo parametr *macaddr* s MAC adresou telefonu. Po získání MAC adresy skript pošle do MySQL databáze SQL dotaz, jehož návratovou hodnotou by mělo být asociativní pole, obsahující všechny informace z databáze pro položku se zvolenou MAC adresou. Tyto informace jsou poté postupně pomocí PHP funkce *echo()* doplněny do konfiguračního souboru.

Samotný konfigurační soubor má stejnou strukturu, jako soubor počáteční, pouze obsahuje více parametrů. Soubor je řetězcem znaků „<!-- -->“ rozdělen na několik částí. První část je identická s počátečním konfiguračním souborem, tj. jedná se pouze o nastavení parametrů provisioningu.

Ve druhé části jsou obecné parametry *GPP_X*, které při konfiguraci telefonu nehrají roli, ale jsou zde pouze pro účely testování. Obsahují totiž proměnné, které budou při dynamickém vygenerování souboru pomocí PHP skriptu nahrazeny skutečnými hodnotami – např. hodnotou parametru *GPP_C* bude po nahrání konfigurace do paměti telefonu IP adresa serveru, ze kterého si telefon konfigurační soubor stáhnul.

V třetí části šablony se nacházejí parametry pro registraci telefonu k ústředně pomocí protokolu SIP. Kromě obecných parametrů, jako je například povolení telefonní linky 1 (*Line_Enable_1 = Yes*) nebo vynucení registrace (*Register = Yes*) jsou zde uvedeny i parametry, jejichž hodnoty jsou opět dynamicky generovány PHP skriptem – jedná se zejména o identifikaci účastníka SIP, heslo pro registraci k ústředně aj.

Poslední část konfiguračního souboru obsahuje pouze doplňkové nastavení, jako je např. preferovaný audio kodek.

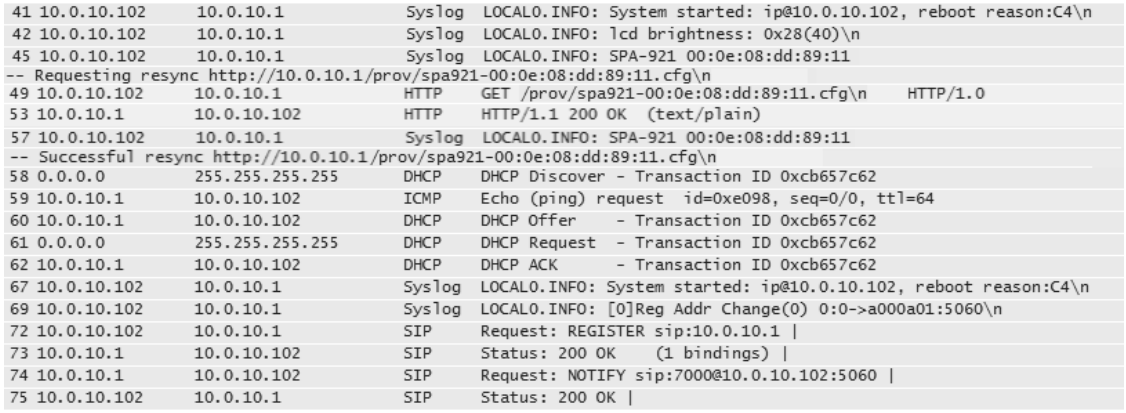
Posledním krokem je uložení konfigurace uvedené v ukázce 2.26 jako textového souboru s názvem *spa921.php* do adresáře HTTP serveru určeného pro dynamické konfigurační soubory, tj. do adresáře */var/www/html/prov*.

VoIP telefon SPA921 nyní bude uveden do továrního nastavení. Zároveň bude opět na rozhraní serveru *eth0* zachytáván provoz, aby bylo možné sledovat průběh konfigurace telefonu.

Ukázka 2.26: Finální konfigurační soubor pro VoIP telefon SPA921.

```
<?php

mysql_connect("localhost", "root", "testpwd") or die(mysql_error());
mysql_select_db("asterisk") or die(mysql_error());

  $macaddress = $_GET['mac'];
  $data = mysql_query("SELECT * FROM ast_sipfriends WHERE macaddr='$macaddress'") or die
    (mysql_error());
  $rows = array();
  while($row = mysql_fetch_array($data))
    $rows[] = $row;

  foreach ($rows as $row){
    $id = $row['id'];
    $name = $row['name'];
    $ipaddr = $row['ipaddr'];
    $secret = $row['secret'];
    $callerid = $row['callerid'];
  }
?>
<flat-profile>
  <Profile_Rule ua="na">http://<?php echo $_SERVER['SERVER_ADDR'] ?>/prov/spa921-$MAC.
    cfg</Profile_Rule>
  <Syslog_Server ua="na"><?php echo $_SERVER['SERVER_ADDR'] ?></Syslog_Server>

  <Provision_Enable ua="na">Yes</Provision_Enable>
  <Resync_On_Reset ua="na">Yes</Resync_On_Reset>
  <Resync_Trigger_1 ua="na">$REGTMR1 gt 60</Resync_Trigger_1>

  <GPP_A ua="na">Auto resync on</GPP_A>
  <GPP_B ua="na">Config file created for <?php echo $ipaddr ?></GPP_B>
  <GPP_C ua="na">Provisioned via HTTP at <?php echo $_SERVER['SERVER_ADDR'] ?></GPP_C>
  <GPP_D ua="na">User is <?php echo $callerid ?></GPP_D>
  <GPP_E ua="na">MAC address is <?php echo $_GET['mac'] ?></GPP_E>

  <Line_Enable_1_ ua="na">Yes</Line_Enable_1_>
  <Proxy_1_ ua="na"><?php echo $_SERVER['SERVER_ADDR'] ?></Proxy_1_>
  <Register_1_ ua="na">Yes</Register_1_>
  <Make_Call_Without_Reg_1_ ua="na">No</Make_Call_Without_Reg_1_>
  <Ans_Call_Without_Reg_1_ ua="na">No</Ans_Call_Without_Reg_1_>

  <Display_Name_1_ ua="na"><?php echo $callerid ?></Display_Name_1_>
  <User_ID_1_ ua="na"><?php echo $name ?></User_ID_1_>
  <Password_1_ ua="na"><?php echo $secret ?></Password_1_>
  <Use_Auth_ID_1_ ua="na">Yes</Use_Auth_ID_1_>
  <Auth_ID_1_ ua="na"><?php echo $name ?></Auth_ID_1_>

  <Preferred_Codec_1_ ua="na">G711a</Preferred_Codec_1_>
  <Silence_Supp_Enable_1_ ua="na">No</Silence_Supp_Enable_1_>
  <Use_Pref_Codec_Only_1_ ua="na">Yes</Use_Pref_Codec_Only_1_>
  <DTMF_Tx_Method_1_ ua="na">Auto</DTMF_Tx_Method_1_>
</flat-profile>
```

Na obrázku 2.5 je možné sledovat průběh automatické konfigurace telefonu. První



Obr. 2.5: Zachycený provoz na portu *eth0* serveru po připojení VoIP telefonu SPA921

fáze konfigurace (stažení souboru z TFTP) je již zachycena na obrázku 2.3. V zaznamenaném provozu jsou nejprve vidět zprávy syslog, které telefon zaslal na syslog server. První z nich informuje o restartu telefonu, druhá potom oznamuje, že si telefon vyžádal konfigurační soubor z HTTP serveru. Následuje HTTP žádost GET na konfigurační soubor umístěný na serveru a odpověď 200 (OK) od serveru. Samotné TCP pakety obsahující konfigurační soubor na tomto obrázku vidět nejsou.

Dále je zachycena syslog zpráva informující o úspěšné synchronizaci se staženým konfiguračním souborem, následuje restart telefonu, po kterém je ve výpisu vidět opět žádost o IP adresu pomocí protokolu DHCP a zprávy syslog informující o restartu. V poslední části souboru je možné vidět úspěšnou registraci telefonu k ústředně pomocí protokolu SIP.

Ověříme nyní funkčnost volání vytočením testovací klapky 9999, která byla definována dříve v souboru *extensions.conf*. Na ukázce 2.27 je výpis z konzole (CLI) Asterisku, který po vytočení klapky provádí definované příkazy, v tomto případě odpoví na volání (Answer), vypíše do konzole hlášku „Playback test“ (Verbose), přehraje určený záznam (Playback), vyčká 10 sekund (Wait) a konečně zavěsí (Hangup).

Ukázka 2.27: Průběh hovoru v konzoli Asterisku.

```
-- Executing [9999@internal:1] Answer("SIP/5000-00000004","") in new stack
-- Executing [9999@internal:2] Verbose("SIP/5000-00000004","1,Playback test") in new
  stack Playback test
-- Executing [9999@internal:3] Playback("SIP/5000-00000004","abandon-all-hope") in new
   stack
-- <SIP/5000-00000004> Playing 'hello.alaw'(language 'en')
-- Executing [9999@internal:4] Wait("SIP/5000-00000004","10") in new stack
-- Executing [9999@internal:5] Hangup("SIP/5000-00000004","") in new stack
== Spawn extension (internal, 9999, 5) exited non-zero on'SIP/5000-00000004'
```

VoIP telefon SPA921 je nyní nastaven a je možné z něj okamžitě volat a to bez jediného zásahu uživatele.

Aby bylo možné pomocí webového nástroje pro konfiguraci koncových zařízení u tohoto telefonu zakázat, resp. povolit automatickou synchronizaci s konfiguračními soubory, je nutné připravit ještě dva další konfigurační soubory. Tyto soubory nemusí obsahovat stejné množství parametrů, jako finální konfigurační soubor, uvedený výše – postačí pouze upravit hodnoty příslušných parametrů. Na ukázce 2.28 je uveden konfigurační soubor, ve kterém jsou parametry pro provisioning záměrně nastaveny tak, aby telefon samovolně neprováděl synchronizaci. Je tedy možné s nastavením telefonu (např. při výuce v laboratoři) libovolně pracovat, bez toho, aby se telefon neustále restartoval a synchronizoval.

Ukázka 2.28: Konfigurační soubor pro vypnutí funkce AutoSync.

```
<flat-profile>
  <Profile_Rule ua="na">http://<?php echo $_SERVER['SERVER_ADDR'] ?>/prov/spa921-$MAC.
    cfg</Profile_Rule>
  <Syslog_Server ua="na"><?php echo $_SERVER['SERVER_ADDR'] ?></Syslog_Server>

  <Provision_Enable ua="na">Yes</Provision_Enable>
  <Resync_Periodic ua="na">36000</Resync_Periodic>
  <Resync_On_Reset ua="na">No</Resync_On_Reset>
  <Resync_From_SIP ua="na">No</Resync_From_SIP>
  <Resync_Trigger_1 ua="na"></Resync_Trigger_1>

  <GPP_A ua="na">Auto resync off$</GPP_A>
</flat-profile>
```

Konfigurační soubor pro povolení automatické synchronizace je identický, pouze má jako hodnoty parametrů *ResyncOnReset* a *ResyncFromSIP* nastaveny na hodnotu *Yes*, a hodnotou parametru *ResyncPeriodic* je 3600 sekund, tj. automatická synchronizace proběhne každou hodinu.

2.4.2 Konfigurace VoIP telefonu Well 3130IF

Možnosti automatické konfigurace telefonu Well 3130IF jsou bohužel o něco více omezené, než u telefonu Linksys SPA921. Na rozdíl od telefonu značky Linksys je u telefonu Well uvedeného do továrního nastavení automatické konfigurace implicitně zakázána, je tedy nutné ji povolit. To lze provést ručním nastavením patřičných parametrů pomocí webového rozhraní telefonu nebo připojením k telefonu protokolem Telnet. Nejprve bude popsána první možnost, tj. ruční nastavení.

Telefon je nutné nejprve uvést do továrního nastavení. Podržíme tlačítko # na klávesnici telefonu, dokud se nezobrazí výzva na zadání hesla (123). Potvrdíme tlačítkem *Local IP/OK*. V menu se lze pohybovat tlačítky *Up*, *Down* a právě *Local*

IP/OK. Postupně vybereme položky *System > Tovarni nastav. > Smaz vse* a opět potvrdíme tlačítkem *Local IP/OK*. Po restartu telefonu odpojením od napájení je telefon uveden do výchozího nastavení.

Telefon je po uvedení továrního nastavení nastaven pro automatické získávání adresy z DHCP serveru. Přidělenou adresu je možné zjistit přímo na telefonu, kdy se po stisknutí tlačítka *Local IP/OK* na displeji zobrazí text `--DHCP-- <IP adresa>`. Webový server telefonu naslouchá na portu 9999. Do prohlížeče tedy zadáme adresu i s portem, v tomto případě adresu 10.0.10.105:9999. Zobrazí se stránka vyžadující přihlášení. Výchozí uživatelské jméno je *root*, heslo nastaveno není.

Obr. 2.6: Nastavení služby provisioning u telefonu Well 3130IF pomocí webového rozhraní

Na obrázku 2.6 je možné vidět oddíl „Správa“, včetně nastavení, která je nutná provést ke zprovoznění služby provisioning. jedná se především o adresu serveru, na kterém je umístěný konfigurační soubor (10.0.10.1/prov), název konfiguračního souboru (nevyplněno), protokol, pomocí kterého bude soubor stažen (HTTP) a mód aktualizace (po startu). Pokud není uveden název konfiguračního souboru, telefon pošle HTTP serveru žádost o stažení souboru, jehož názvem je MAC adresa telefonu – tedy např. soubor „00094560ccf“.

Nyní stačí pouze nastavení aplikovat pomocí tlačítka *Použít*, uložit jej do paměti telefonu (záložka *Nastavení > Uložit*) a telefon restartovat (záložka *Restartovat > Restartovat*).

Druhou variantou je povolení služby provisioning pomocí protokolu Telnet. K telefonu je možné se připojit libovolným telnetovým klientem na TCP portu 23. Po připojení je nutné zadat uživatelské jméno a heslo, které je stejné jako při ruční konfiguraci – uživatel *root*, heslo nastaveno není. Po úspěšném přihlášení je nejjednodušší variantou, jak telefon správně nastavit, stáhnutí konfiguračního souboru, kde je provisioning povolen. Soubor lze do telefonu stáhnout zadáním příkazu *download tftp*

-ip <ip adresa> -file <název souboru>. Telefon po zadání příkazu stáhne z TFTP serveru se zadanou adresou konfigurační soubor se zadaným názvem a nahraje jej do flash paměti. Po restartu telefonu příkazem *reload* je nastavení hotovo.

Poslední variantou prvnotního nastavení telefonu je provedení synchronizace pomocí nástroje **Sync Tool**, viz kapitola 2.5.10.

Konfigurační soubor nutný pro povolení provisioningu je uveden na ukázce 2.29.

Ukázka 2.29: Prvotní konfigurační soubor telefonu Well.

```
<<VOIP CONFIG FILE>>Digests:2.0002

<AUTOUPDATE CONFIG MODULE>
Download Server IP :10.0.10.1/prov
Config File Name   :
Config File Key    :
Download Protocol  :4
Download Mode      :1
Download Interval  :1

<<END OF FILE>>
```

Stejně jako u telefonu SPA921 má tento konfigurační soubor textovou formu, rozdílem ovšem je, že hodnoty parametrů již nejsou uzavřeny do XML značek, ale jsou od názvů parametrů odděleny dvojtečkou – každý parametr je umístěn na samostatném řádku. Soubor je rovněž členěn do oddílů, tentokrát oddělených řetězcem `<název oddílu>`.

Soubor je uvozen řetězcem «`VOIP CONFIG FILE`»`Digests:2.0002`, který vyjadřuje verzi konfiguračního souboru. Protože telefon po stažení souboru porovnává verzi nového souboru s verzí starého (nahraného v paměti telefonu), je nutné po každé změně hodnotu verze inkrementovat. Pokud totiž telefon zjistí, že je číslo verze souboru nižší nebo stejné jako souboru, který má aktuálně v paměti, nově stažený soubor zahodí. Telefon ovšem nabízí možnost, kdy automaticky porovnává kontrolní součet obou konfiguračních souborů a zjišťuje tak, zda došlo mezi verzemi k nějaké změně. Tuto funkcnionalitu zajišťuje klíčové slovo `Digests`.

Oddíl `<AUTOUPDATE CONFIG MODULE>` slouží právě pro nastavování parametrů služby provisioning. Kromě IP adresy serveru, na kterém jsou umístěny konfigurační soubory, je možné také nastavit, jaký protokol bude použit pro přenos (TFTP, FTP, HTTP). Parametr *Download Mode* určuje, zda je provisioning povolen (1), či naopak zakázán (0), a parametr *Download Interval* udává počet hodin, po uplynutí kterých má proběhnout automatická synchronizace.

Obdobně jako u telefonu SPA921, i konfigurační soubor pro telefon Well 3130IF obsahuje PHP skript, který do konfiguračního souboru dynamicky doplní všechny

potřebné údaje podle informací z databáze. Úplný konfigurační soubor, obsahující všechno potřebné nastavení, je uveden v ukázce 2.30.

Ukázka 2.30: Konfigurační soubor pro telefon Well 3130IF.

```
<?php
mysql_connect("localhost", "root", "testpwd") or die(mysql_error());
mysql_select_db("asterisk") or die(mysql_error());
  $macaddress = $_GET['mac'];
  $data = mysql_query("SELECT * FROM ast_sipfriends WHERE macaddr='$macaddress'") or die
    (mysql_error());
  #if (mysql_num_rows($data)==0) {};
  #mysql_data_seek($data, 2);
  $rows = array();
  while($row = mysql_fetch_array($data))
    $rows[] = $row;
  foreach ($rows as $row){
    $id = $row['id'];
    $name = $row['name'];
    $ipaddr = $row['ipaddr'];
    $secret = $row['secret'];
    $callerid = $row['callerid'];
  }
?>
<<VOIP CONFIG FILE>>Digests:2.0002
<SIP CONFIG MODULE>
SIP  Port          :5060
Reg Retry Time     :30
--SIP Line List--  :
SIP1 Phone Number  :<?php echo $name ?>
SIP1 Display Name  :<?php echo $callerid ?>
SIP1 Sip Name      :<?php echo $name ?>
SIP1 Register Addr :<?php echo $_SERVER['SERVER_ADDR'] ?>
SIP1 Register Port :5060
SIP1 Register User :<?php echo $name ?>
SIP1 Register Pwd  :<?php echo $secret ?>
SIP1 Register TTL  :60
SIP1 Enable Reg    :1
SIP1 Proxy Addr    :<?php echo $_SERVER['SERVER_ADDR'] ?>
SIP1 Proxy Port    :5060
SIP1 Proxy User    :<?php echo $name ?>
SIP1 Proxy Pwd     :<?php echo $secret ?>
SIP1 User Agent    :WELL 3130IF
<AAA CONFIG MODULE>
Enable Syslog      :1
Syslog address     :<?php echo $_SERVER['SERVER_ADDR'] ?>
Syslog port        :514
<PHONE CONFIG MODULE>
Keypad Password    :123
LCD Logo           :User <?php echo $name ?>
<AUTOUPDATE CONFIG MODULE>
Download Server IP :<?php echo $_SERVER['SERVER_ADDR'] ?>/prov
Config File Name   :
Download Protocol  :4
Download Mode      :1
Download Interval  :1
<<END OF FILE$>>
```

Na obrázku 2.7 je zachycen provoz na rozhraní *eth0* po připojení telefonu Well 3130IF k serveru, resp. po jeho restartování. Jako první můžeme vidět syslog zprávu informující o restartu telefonu. Následuje již známá výměna DHCP zpráv, po které je telefonu přidělena IP adresa. Poté je již možné ve výpisu vidět samotnou HTTP GET žádost o stažení konfiguračního souboru. Po stažení souboru (zpráva HTTP 200 OK) telefon zkontroluje stažený soubor, zapíše jej do flash paměti, informuje o úspěšném zápisu a poté se restartuje, což je vidět ze zachycených syslog zpráv. Po dokončení restartu telefonu se opakuje výměna DHCP zpráv a dále můžeme vidět samotnou registraci telefonu k ústředně pomocí protokolu SIP.

```
3032 10.0.10.105  10.0.10.1      Syslog  USER.NOTICE: [MGR]  | NOTICE | System is rebooting....\n
3033 10.0.10.105  10.0.10.1      DHCP    DHCP Release  - Transaction ID 0x714adf5
3034 0.0.0.0      255.255.255.255 DHCP   DHCP Discover - Transaction ID 0x716ae948
3036 10.0.10.1    10.0.10.105    DHCP    DHCP Offer    - Transaction ID 0x716ae948
3037 0.0.0.0      255.255.255.255 DHCP   DHCP Request  - Transaction ID 0x716ae948
3038 10.0.10.1    10.0.10.105    DHCP    DHCP ACK      - Transaction ID 0x716ae948
3050 10.0.10.105  10.0.10.1      HTTP    GET /phoneprov/well3130if.txt HTTP/1.0
3052 10.0.10.105  10.0.10.1      Syslog  USER.INFO: [MGR]  | INFO   | DHCP server start successfully\n
3053 10.0.10.105  10.0.10.1      Syslog  USER.NOTICE: [MGR]  | NOTICE | NAT module start successfully\n
3060 10.0.10.1    10.0.10.105    HTTP    HTTP/1.1 200 OK  (text/plain)
3064 10.0.10.105  10.0.10.1      Syslog  USER.INFO: [MGR]  | INFO   | Start to match config \n
3065 10.0.10.105  10.0.10.1      Syslog  USER.CRIT: [MGR]  | BUG    | flashWrite addr:0xb01e0000  offset :8,len:208f\n
3066 10.0.10.105  10.0.10.1      Syslog  USER.INFO: [MGR]  | INFO   | save conf write ok\n
3067 10.0.10.105  10.0.10.1      Syslog  USER.NOTICE: [MGR]  | NOTICE | System is rebooting....\n
3068 10.0.10.105  10.0.10.1      DHCP    DHCP Release  - Transaction ID 0x714adf5
3074 0.0.0.0      255.255.255.255 DHCP   DHCP Discover - Transaction ID 0x716ae948
3075 10.0.10.1    10.0.10.105    DHCP    DHCP Offer    - Transaction ID 0x716ae948
3076 0.0.0.0      255.255.255.255 DHCP   DHCP Request  - Transaction ID 0x716ae948
3077 10.0.10.1    10.0.10.105    DHCP    DHCP ACK      - Transaction ID 0x716ae948
3082 10.0.10.105  10.0.10.1      SIP     Request: REGISTER sip:10.0.10.1:5060 |
3083 10.0.10.1    10.0.10.105    SIP     Status: 401 Unauthorized    (0 bindings) |
3084 10.0.10.105  10.0.10.1      SIP     Request: REGISTER sip:10.0.10.1:5060 |
3085 10.0.10.1    10.0.10.105    SIP     Status: 200 OK    (1 bindings) |
3086 10.0.10.1    10.0.10.105    SIP     Request: NOTIFY sip:5000@10.0.10.105:5060 |
3087 10.0.10.105  10.0.10.1      SIP     Status: 200 OK |
3088 10.0.10.105  10.0.10.1      Syslog  USER.INFO: [MGR]  | INFO   | DHCP server start successfully\n
3089 10.0.10.105  10.0.10.1      Syslog  USER.NOTICE: [MGR]  | NOTICE | NAT module start successfully\n
3090 10.0.10.105  10.0.10.1      Syslog  USER.INFO: [MGR]  | INFO   | informative_cb!\n
```

Obr. 2.7: Provoz na rozhraní *eth0* serveru po připojení telefonu 3130IF

Telefon je nyní nakonfigurován pro volání, což lze ověřit vytočením testovací klapky 9999. V konzoli Asterisku by měl být vidět obdobný výpis, jako na ukázce 2.27.

2.5 Aplikace Provisioning Tool

V této části bude popsána funkcionalita navržené a vytvořené webové aplikace a také podrobně rozebrány všechny části, které ji tvoří.

Samotná aplikace by se dala rozdělit na dvě pomyslné části – část **Provisioning Tool**, která má za úkol práci s MySQL databází běžící na pozadí (často označována také jako *backend*, „zadní část“ aplikace skrytá uživateli), resp. zobrazování, přidávání, úpravu a mazání záznámů v databázi, na základě kterých jsou potom generovány konfigurační soubory pro jednotlivé telefony a část **Sync Tool**, umožňující ručně vyhledávat dostupné telefony v sítí a poté je synchronizovat s výše zmíněnými konfiguračními soubory.

Nejprve budou popsány soubory *ProvTool.php*, *SyncTool.php* a *PHPTelnet.php* obsahující stejnojmenné PHP třídy s obslužnými funkcemi (metodami), které jsou dále volány aplikací na pokyn uživatele a následně ostatní PHP soubory, které již tvoří samotné webové rozhraní aplikace.

2.5.1 Soubor _class/ProvTool.php

V souboru *_class/ProvTool.php* se nachází deklarace třídy **ProvTool**, obsahující metody zejména pro práci s MySQL databází. Tato třída je používána nástrojem **Provisioning Tool**.

Třída **ProvTool** má deklarovány 4 vlastnosti: *$host, $username, $password, $database* – všechny tyto vlastnosti slouží pro připojení k MySQL databázi.

Metoda db_connect()

Tato metoda se po zavolání pokusí připojit k MySQL databázi podle výše zmíněných vlastností, se kterými je vytvořena instance třídy **ProvTool**. V metodě jsou využívány funkce *mysql_connect()* a *mysql_select_db()* pro připojení k MySQL serveru a zvolení databáze, se kterou se bude pracovat. V případě neúspěchu jedné z těchto dvou operací se metoda zastaví a vrátí pomocí funkce *mysql_error()* chybu, která nastala.

Metoda db_display_all()

Metoda *db_display_all()* se po jejím zavolání nejprve pokusí připojit k MySQL databází pomocí výše zmíněné metody *db_connect()* a pokud je připojení úspěšné, vybere pomocí funkce *mysql_query* z tabulky databáze (zde tabulka *ast_sipfriends*) všechny položky (SQL dotaz *"SELECT * FROM ast_sipfriends"*).

Metoda dále vytiskne pomocí funkce *echo* do HTML stránky hlavičku tabulky a poté postupným iterováním funkce *mysql_fetch_array()* přes asociativní pole,

které vrací funkce *mysql_query*, naplní jednotlivé řádky tabulky údaji z databáze. Výsledná tabulka obsahuje údaje o pozici položky v databázi (id), názvu/klapce telefonu (name), heslu pro registraci (secret), identifikaci volajícího (callerid), typu zařízení (useragent a phonetype) a konečně adrese IP a MAC (macaddr, ipaddr).

Metoda db_display_id($id)

Tato metoda funguje téměř stejně jako metoda *db_display_all()*, ovšem s tím rozdílem, že obsahuje vstupní parametr *$id* – číslo záznamu v databázi. Tato metoda tedy z databáze získává pouze informace o jednom vybraném záznamu. Oproti výše zmíněné metodě navíc po položení SQL dotazu do databáze testuje, zda dotaz vrátil alespoň jeden výsledek. Pokud ano, metoda vytiskne do HTML stránky veškeré informace o položce z databáze, např. informace o kontextu, ve kterém se telefon nachází v rámci číslovacího plánu Asterisku nebo údaje o SIP registraci. V opačném případě, tj. výsledkem dotazu je nulový počet řádků (testováno pomocí funkce *mysql_num_rows()*), metoda do HTML stránky vytiskne informaci o tom, že položka se zadaným ID v databázi nebyla nalezena.

Metoda db_add_id($name, $secret, $callerid, $phonetype, $macaddr)

Tato metoda po připojení k databázi pomocí metody *db_connect()* vloží do databáze novou položku (telefon) s parametry zadanými uživatelem na HTML stránce **phone_add.php**. Pokud je vložení úspešné, informuje o tom uživatele vytisknutím informace do HTML stránky, v opačném případě vypíše opět pomocí funkce *mysql_error()* chybu, která nastala.

Metoda db_edit_id($id)

Metoda *db_edit_id()* se stará o výpis informací o zvolené položce (id) do HTML stránky **phone_edit.php**, aby je poté bylo možné upravit. Po svém zavolání se nejprve připojí k databázi, pomocí SQL dotazu na konkrétní záznam získá z databáze všechny potřebné informace a ty poté vytiskne do HTML stránky v HTML formuláři, pomocí kterého lze získané informace následně upravit. Stejně jako metoda *db_display_id*, i tato metoda v případě, že položka v databázi nebyla nalezena (počet získaných výsledků je 0), vytiskne HTML stránku informující o této skutečnosti.

Metoda db_update_id($id, $name, $secret, $callerid, $phonetype, $macaddr)

Tato metoda je volána akcí uživatele poté, co upravil v HTML formuláři údaje o položce a změny potvrdil tlačítkem na stránce **phone_edit.php**. Podobně jako ostatní metody pracující s daty v databázi, i tato metoda se nejprve k databázi připojí a poté odešle SQL dotaz *"UPDATE"*, který aktualizuje parametry položky v databázi podle změn v HTML formuláři provedených uživatelem. Pokud je zápis do databáze úspěšný, je o tom uživatel opět informován vytisknutím informace do HTML stránky. V opačném případě, tj. pokud je počet změněných řádků v databázi nulový (testováno funkcí *mysql_affected_rows()*, např. pokud uživatel žádné změny neprovedl a formulář odeslal, se do HTML stránky vytiskne patřičná informace. Tato metoda je podobná metodě *db_add_id()*, ale navíc pracuje s parametrem *$id*, neboli s identifikátorem položky.

Metoda db_remove_id($id)

Poslední metodou ve třídě **ProvTool** je metoda *db_remove_id()*. Vstupním parametrem této metody je proměnná *$id*, identifikátor položky v databázi, na základě kterého metoda po připojení k databázi provede SQL dotaz *"SELECT"* na zadané ID. Pokud se položka s tímto identifikátorem v databázi nachází, provede metoda další SQL dotaz *"DELETE FROM"*, který položku z databáze vymaže. V opačném případě (výsledek dotazu na položku je prázdný) metoda do HTML stránky vytiskne informaci o tom, že položka v databázi nebyla nalezena.

2.5.2 Soubor _class/PHPTelnet.php

Tento soubor obsahuje deklaraci třídy *PHPTelnet*, kterou využívá stejnojmenný nástroj umožňující realizovat spojení pomocí protokolu Telnet v rámci PHP kódu. Autorem tohoto nástroje s licencí volného díla (public domain) je Antone Roundy. [16] Tato třída je používána některými metodami třídy **SyncTool**.

2.5.3 Soubor _class/SyncTool.php

Soubor *_class/SyncTool.php* obsahuje deklaraci stejnojmenné třídy, obsahující metody, které využívá nástroj **Sync Tool**. Jedná se zejména o metody pracující s IP adresami a metody sloužící k testování spojení a synchronizaci VoIP telefonů ke konfiguračním souborům. Jak již bylo řečeno, některé metody této třídy využívají třídu **PHPTelnet**, proto je tato třída v deklaraci třídy **SyncTool** zahrnuta direktivou *require_once*.

Metoda ip_validate($ipaddr)

Tato metoda slouží k ověření, zda je IP adresa, která je jejím vstupním parametrem, platná. Pro ověření adresy je volána funkce *inet_pton*, která převádí zadaný řetězec (IP adresu) na 32bitovou (v případě IPv4 adresy) binární strukturu. Pokud adresa není platná, funkce vrátí hodnotu *„FALSE“*, čehož je využito v této metodě – pokud je IP adresa neplatná, návratovou hodnotou metody je „0“, v opačném případě metoda vrátí hodnotu „1“.

Metoda ip_range($rangestart, $rangeend)

Metoda *ip_range()* generuje ze vstupních parametrů, kterými jsou počáteční a koncová IP adresa, pole IP adres (rozsah). Po zavolání této metody jsou nejprve vstupní parametry (IP adresy) ověřeny metodou *ip_validate()*. Pokud není některá ze zadaných IP adres v platném formátu, metoda do HTML stránky vypíše o této skutečnosti informaci. V opačném případě metoda pomocí funkce *ip2long()* převede obě zadané adresy do celočíselného formátu. Po převodu metoda testuje, zda je celočíselná hodnota počateční adresy (*$rangestart*) menší, než celočíselná hodnota koncové adresy (*$rangeend*). Pokud ne, je o tom uživatel informován vytisknutím patřičné informace do HTML stránky, pokud ano, metoda dále otestuje pomocí rozdílu hodnoty koncové adresy a počáteční adresy, zda není zadaný rozsah příliš velký, tj. větší než podsíť třídy C (254 IP adres). Pokud zadané parametry vyhovují všem těmto podmínkám, metoda pomocí funkce *range()* vytvoří z počáteční a koncové hodnoty pole (rozsah) celočíselných adres, které posléze postupně převede funkcí *long2ip()* zpět na čitelné IP adresy (tečkovaný formát). Návratovou hodnotou metody je tedy pole (*$iparray*), obsahující všechny IP adresy v rozsahu určeném počáteční a koncovou IP adresou.

Metoda check_host($ipaddr)

Metoda *check_host()* testuje, zda hostitel určený pomocí vstupního parametru (IP adresy) přijímá spojení na TCP portu 80 a 9999. Na těchto portech jsou dostupná webová rozhraní pro konfiguraci telefonů (port 80 u telefonu Linksys SPA921, port 9999 u telefonu Well 3130IF) a tato metoda je tedy poměrně jednoduchý způsob, jak určit, zda je na testované IP adrese konkrétní typ telefonu. Předpokládá se, že budou testovány pouze rozsahy IP adres vyhrazené pro telefony, protože v opačném případě metoda chybně označí např. webový server naslouchající na portu TCP 80 za telefon SPA921. Spojení je postupně testováno na obou portech pomocí funkce *fsockopen()* s časovým limitem 200ms na jedno spojení, aby bylo možné relativně rychle prohledat zadaný rozsah. Návratovou hodnotou metody je asociativní pole

$result, které obsahuje klíč – testovanou IP adresu – s hodnotou „spa921“, pokud se podařilo navázat spojení na TCP portu 80, hodnotou „well3130if“, pokud se podařilo navázat spojení na TCP portu 9999 nebo hodnotou „unreachable“, pokud se nepodařilo navázat spojení ani na jednom z testovaných portů.

Metoda sync ($ipaddr, $prov)

Další metodou třídy **SyncTool** je metoda *sync()*. Tato metoda slouží k zasílání příkazů k synchronizaci s konfiguračním souborem jedntlivým VoIP telefonům. Vstupními parametry je IP adresa zařízení (*$ipaddr*) a parametr *$prov*, který určuje konfigurační soubor, se kterým se telefon má synchronizovat – buď úplný (výchozí) konfigurační soubor nebo soubor, kde je pouze zakázána/povolena automatická synchronizace.

Metoda nejprve zavolá metodu *check_host*, která jí umožní získat typ zařízení nacházející se na zadané IP adrese. Metoda poté v případě, že se jedná o telefon Linksys SPA921, otevře pomocí funkce *file_get_contents()* URL, ve kterém je zakomponovaná IP adresa telefonu, IP adresa serveru a cesta ke zvolenému konfiguračnímu souboru, čímž telefonu dá příkaz k synchronizaci.

Pokud se jedná o telefon Well 3130IF, metoda vytvoří novou instanci třídy **PHPTelnet** a po sestavení Telnet spojení telefonu zašle postupně příkaz pro stažení vybraného konfiguračního souboru z TFTP serveru a příkaz k restartování telefonu, poté je Telnet spojení ukončeno. Uživatel je o výsledku synchronizace informován vytištěním výsledku synchronizace do HTML stránky.

Metoda sync_all ($iprangestart, $iprangeend, $prov)

Poslední metoda třídy **SyncTool** slouží k hromadné synchronizaci všech telefonů v určeném rozsahu IP adres. Vstupními parametry metody je počáteční (*$iprangestart*) a koncová (*$iprangeend*) adresa zmíněného rozsahu a opět parametr *$prov*, který stejně jako u metody *sync()* určuje typ konfiguračního souboru.

Metoda nejprve pomocí metody *ip_range()* získá pole všech IP adres nacházejících se v zadaném rozsahu a v případě, že toto pole není prázdné, postupně pro každou IP adresu v poli volá metodu *sync()*, čímž zesynchronizuje všechny telefony v rozsahu se zadaným konfiguračním souborem.

2.5.4 Soubor header.php

V souboru *header.php* je obsažena hlavička, která se pomocí direktiv *include* připojí ke všem ostatním PHP souborům, tvořícím uživatelské rozhraní aplikace. Je

zde definována pouze „hlava“ (head) a „tělo“ (body) HTML souboru s navigačním panelem obsahujícím odkazy na jednotlivé části aplikace.

2.5.5 Soubor index.php

Soubor *index.php*, jak název napovídá, obsahuje kód pro vytvoření titulní stránky aplikace. Na titulní stránce se kromě připojené hlavičky nachází také tabulka, vypisující základní informace o všech záznamech (telefonech) vytvořených v databázi. Toho je dosaženo vytvořením nové instance třídy **ProvTool** a zavoláním funkce *db_display_all()*. Pokud je stránka *index.php* zavolána metodou GET a v URL se nachází parametry *function* a *id*, přesměruje stránka uživatele do jiné části aplikace podle toho, jakou funkci (zobrazení, přidání nebo úpravu záznamu v databázi) uživatel vybral pomocí lišty s dostupnými funkcemi umístěné ve spodní části HTML stránky.

2.5.6 Soubor phone_add.php

V souboru *phone_add.php* je obsažen kód, který generuje HTML stránku pro přidání nového záznamu do databáze. Uživatel na tuto stránku může přistupovat pomocí odkazu v hlavičce. Po připojení hlavičky je vytvořena nová instance třídy **ProvTool** a do HTML stránky je vytištěn formulář, pomocí kterého může uživatel zadat parametry přidávaného telefonu – jméno/klapku, heslo pro registraci k ústředně, identifikaci volajícího, typ telefonu a jeho MAC adresu. Po odeslání formuláře je stránka znovu načtena, tentokrát i se zadanými parametry obsaženými v URL. Pokud uživatel správně vyplnil všechny potřebné údaje, je zavolána funkce *db_phone_add()*, která do databáze přidá novou položku. V opačném případě, tj. pokud nejsou zadány všechny potřebné parametry, je uživateli zobrazena informace, vyzývající ho k zadání všech parametrů.

2.5.7 Soubor phone_edit.php

Tato část aplikace slouží pro úpravu položek v databázi, resp. pro vypsání údajů z databáze do formuláře v HTML stránce, aby je uživatel mohl jednoduše upravit a poté odeslat zpět do databáze. Po připojení hlavičky a vytvoření nové instance třídy **ProvTool** je v případě, že je v URL uveden správný parametr s identifikací záznamu v databázi *(id)*, zavolána funkce *db_edit_id()*, která se stará o již zmíněné vytištění formuláře i s údaji získanými z databáze.

2.5.8 Soubor phone_update.php

Na tuto stránku je uživatel přesměrován po změně parametrů upravované položky a odeslání formuláře. Po připojení hlavičky a vytvoření instance třídy **ProvTool** PHP kód obsažený v souboru načte všechny potřebné parametry z URL a ty poté předá funkci *db_update_id()*, která aktualizuje parametry upravované položky v databázi na nové, uživatelem zadané, hodnoty. O úspěchu, či naopak případném neúspěchu této operace je uživatel informován vytištěním patřičné zprávy do HTML stránky.

2.5.9 Soubor phone_remove.php

Stránka *phone_remove.php* slouží k odstranění vybraného záznamu z databáze. Na tuto stránku je uživatel přesměrován z titulní stránky aplikace (*index.php*), s URL obsahujícím identifikátor položky, která má být odstraněna. Uživateli se po načtení stránky zobrazí dotaz, zda chce opravdu zvolenou položku odstranit. Pokud uživatel výzvu potvrdí, stránka je načtena znovu s URL obsahujícím parametr *confirm* s hodnotou *„Yes“* a je volána funkce *db_phone_remove()*, která zajistí odstranění položky z databáze. Pokud uživatel výzvu nepotvrdí, tj. hodnotou parametru *confirm* je řetězec *„No“*, je uživatel přesměrován zpět na výchozí stránku aplikace, *index.php*,ž a položka v databázi zůstává nezměněna.

2.5.10 Soubor sync_tool.php

Stránka *sync_tool.php* představuje uživatelské rozhraní k vytvořenému nástroji **SyncTool**. Po otevření stránky se testuje její URL na přítomnost parametru *function*, který slouží k provedení funkcí dostupných v tomto nástroji. Může nastat několik případů:

a) Parametr *function* se v URL nenachází:

Toto je výchozí stav po přesměrování uživatele na stránku *sync_tool.php* z titulní stránky. Pokud PHP kód nezjistí přítomnost parametru v URL, vypíše do HTML stránky výzvu pro zadání adresy nebo rozsahu adres, kterou má nástroj zkontrolovat. Zároveň s tím je do HTML stránky vytištěn odpovídající formulář, kde lze IP adresy zadat a pomocí tlačítka *Check Host* zkontrolovat pouze jednu IP adresu, případně pomocí tlačítka *Check All In Range* prověřit všechny adresy v zadaném rozsahu. Stránka *sync_tool.php* je poté načtena znovu, tentokrát již s patřičnými parametry v URL.

b) Hodnotou parametru *function* je *CheckHost*:

Pokud je při kontrole URL nalezen parametr *function* s hodnotou *CheckHost*, při načtení stránky je nejprve vytvořena instance třídy **SyncTool**, dále je

z URL načten parametr *iprangestart*, který je v tomto případě roven požadované IP adrese a tento parametr je dále předán do metody *ip_validate()*, která ověří, zda je zadaná IP adresa platná. Pokud je zjištěno, že zadaná IP adresa je neplatná (špatný formát, adresa nepatří do adresního prostoru IPv4, ...), je o tom uživatel informován vytištěním zprávy do HTML stránky a musí zadání opakovat. V případě, že se jedná o platnou IPv4 adresu, je volána funkce *check_host()* a výsledkem je potom tabulka, kde je v jejím řádku uvedena IP adresa, typ zařízení a pokud se jedná o jeden z používaných telefonů, je přítomno také tlačítko umožňující výběr této položky. Dále je vytištěn nový HTML formulář, kde si uživatel může zvolit, zda chce telefon synchronizovat s jedním z konfiguračních souborů.

c) Hodnotou parametru *function* je *CheckRange*:

Při nalezení parametru s touto hodnotou v URL nástroj postupuje obdobně jako v předchozím případě – nejprve je testován zadaný rozsah IP adres na jejich platnost a v případě, že se jedná o platné IP adresy, nástroj vygeneruje pole obsahující všechny IP adresy v rozsahu. Toto pole je dále adresu po adrese opět předáváno metodě *check_host()*. Výsledkem operace je tabulka obsahující IP adresy, typy zařízení a možnosti výběru jednotlivých zařízení. Narozdíl od funkce *CheckHost* je poté ovšem možné zesynchronizovat buď pouze jeden, nebo všechny nalezené telefony – k tomu slouží HTML formulář rozšířený o patřičnou možnost.

d) Hodnotou parametru *function* je *Sync*:

Přítomnost parametru s touto hodnotou URL nástroji signalizuje, že uživatel již prověřil zadaný rozsah adres (případně pouze jednu adresu) a přeje si zařízení s touto IP adresou zesynchronizovat s konfiguračním souborem. Po načtení potřebných parametrů z URL (IP adresa, konfigurační soubor) jsou tyto parametry předány metodě *sync* třídy **SyncTool**, která provede samotnou synchronizaci. Uživatel je o výsledku tohoto procesu informován zprávou vytištěnou do HTML stránky.

e) Hodnotou parametru *function* je *SyncAll*:

Nástroj po zjištění této hodnoty parametru provádí stejnou funkci, jako v případě výše zmíněné hodnoty *Sync*, ale pro všechny IP adresy v zadaném rozsahu. Místo metody *sync()* je volána metoda *sync_all()* s patřičnými parametry načtenými z URL.

3 ZÁVĚR

Tato práce se věnovala obecnému popisu služby provisioning, možnostem její implementace do open-source PBX ústředny Asterisk včetně stručného popisu protokolů, které jsou pro implementaci a správnou funkci této služby potřebné. Byly v ní rozebrány možnosti implementace z hlediska dostupných koncových zařízení pro VoIP telefonii a také navržena a vytvořena aplikace pro správu těchto koncových zařízení.

Navrhnutý scénář implementace byl poté ověřen na virtuálním serveru s operačním systémem CentOS, do kterého byla nainstalována samotná ústředna a také ostatní zmiňované služby. Práce rovněž obsahuje popis obsahu veškerých konfiguračních souborů serveru, softwarové ústředny i hardwarových koncových zařízení pro VoIP telefonii, které jsou nutné pro základní provoz této služby i samotných zařízení. Společně s tím byla i ověřena funkčnost realizované aplikace pro správu koncových zařízení. Tím byly splněny všechny cíle této práce.

Práci by případně bylo možné rozvinout rozšířením podporovaných koncových zařízení o nové telefony nebo vytvořením a přidáním více druhů konfiguračních souborů pro jednotlivá koncová zařízení umožňujících konfiguraci dalších různých parametrů zařízení. Samotná aplikace by mohla být vylepšena například o funkci automatického generování telefonního seznamu obsahujícího položky pro všechna koncová zařízení vedená v databázi.

LITERATURA

[1] Davenport, M. *Phone Provisioning in Asterisk*. Asterisk Project. [online]. 2010, poslední aktualizace 31. 8. 2010 [cit. 25. 12. 2013]. Dostupné z URL: <https://wiki.asterisk.org/wiki/display/AST/Phone+Provisioning+in+Asterisk>.

[2] Davenport, M.; Newton, R. *Realtime Database Configuration*. Asterisk Project. [online]. 2010, poslední aktualizace 23. 5. 2013 [cit. 27. 5. 2014]. Dostupné z URL: <https://wiki.asterisk.org/wiki/display/AST/Realtime+Database+Configuration>.

[3] Meggelen, J. V.; Madsen, L.; Smith, J. *Asterisk: The Future of Telephony, Second Edition*. O'Reilly Media, Inc., 2007

[4] Anonym. *The Asterisk RealTime Architecture*. VOIP-Info.org LLC [cit. 27. 5. 2014]. Dostupné z URL: <http://www.voip-info.org/wiki/view/Asterisk+RealTime>.

[5] Anonym. *Auto Provision Manual for WELL 3195IF and 3130IF*. [cit. 20. 12. 2013]. Dostupné z URL: <http://www.joyce.cz/files/technicka-podpora/prirucky/voip-zarizeni/provisioning-manual-3195if-3130if.pdf>.

[6] Anonym. *Linksys SPA Provisioning Guide: Version 3.01* Cisco Systems, Inc., 2007

[7] Droms, R. *RFC 2131 - Dynamic Host Configuration Protocol*. Technická zpráva, Internet Engineering Task Force - Network Working Group, 1997

[8] Alexander, S.; Droms, R. *RFC 2132 - DHCP Options and BOOTP Vendor Extensions*. Technická zpráva, Internet Engineering Task Force - Network Working Group, 1997

[9] Sollins, K. *RFC 1350 - The TFTP Protocol (Revision 2)*. Technická zpráva, Internet Engineering Task Force - Network Working Group, 1992

[10] Fielding, R.; Gettys, J.; Mogul, J.; aj. *RFC 2616 - Hypertext Transfer Protocol –– HTTP/1.1*. Technická zpráva, Internet Engineering Task Force - Network Working Group, 1999

[11] Gerhards, R. *RFC 5424 - The Syslog Protocol*. Technická zpráva, Internet Engineering Task Force - Network Working Group, 2009

[12] Axmark, D.; Widenius, M.; MySQL Documentation Team *MySQL 5.6 Reference Manual*. Oracle Corporation [cit. 27. 5. 2014]. Dostupné z URL: <http://downloads.mysql.com/docs/refman-5.6-en.a4.pdf>.

[13] Anonym. *ODBC–Open Database Connectivity Overview*. Microsoft Knowledge Base [cit. 27. 5. 2014]. Dostupné z URL: <http://support.microsoft.com/kb/110093/en-us>.

[14] Anonym. *PHP Manual Preface*. The PHP Group [cit. 27. 5. 2014]. Dostupné z URL: <http://www.php.net/manual/en/preface.php>.

[15] Anonym. *History of PHP*. The PHP Group [cit. 27. 5. 2014]. Dostupné z URL: <http://www.php.net/manual/en/history.php.php>.

[16] Roundy, A. *PHP Telnet 1.1*. Antone Roundy's Gecko Tribe. [online]. [cit. 20. 5. 2013]. Dostupné z URL: <http://www.geckotribe.com/php-telnet/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

IP Internet Protocol

TCP Transmission Control Protocol

UDP User Datagram Protocol

DHCP Dynamic Host Configuration Protocol

TFTP Trivial File Transfer Protocol

HTTP HyperText Transfer Protocol

HTML HyperText Markup Language

SIP Session Initiation Protocol

IAX Inter-Asterisk eXchange

PBX Private Branch Exchange

MAC Media Access Control

XML eXtended Markup Language

VoIP Voice over Internet Protocol

QoS Quality of Service

CentOS Community enterprise Operating System

NIC Network Interface Card

URL Uniform Resource Locator

WAN Wide Area Network

LAN Local Area Network

SQL Structured Query Language

ODBC Open DataBase Connection

PHP PHP: Hypertext Preprocessor

SEZNAM PŘÍLOH

A OBSAH CD

Přiložený nosič obsahuje následující:

- Elektronickou verzi této práce (soubor xkralj08_bp.pdf)
- Konfigurační soubory jednotlivých koncových zařízení (soubor configs.zip)
- Realizovaný webový nástroj (soubor provtool.zip)
- Ukázky uživatelského rozhraní webového nástroje (soubor provtool_obr.zip)

B POKYNY K INSTALACI

Pro správnou funkci všech prvků popsaných v této práci musí být v operačním systému (uvažován CentOS verze 6.5) přítomny následující balíčky a jejich prerekvizity (ukázka B.1):

Ukázka B.1: Seznam softwarových balíčků.

```
gcc
gcc-c++
ncurses-devel
libtermcap-devel
libtool-ltdl
libtool-ltdl-devel
libuuid

unixODBC
unixODBC-devel

php
php-pdo
php-mysql
php-mbstring

mysql
mysql-server
mysql-connector-odbc

dhcp
httpd
telnet
```

Instalace a nastavení všech potřebných softwarových komponentů je podrobně rozebrána v částech 2.2 a 2.3. Některé parametry (zejména rozsahy IP adres, uživatelská jména a hesla) může být nutné nastavit jiným způsobem, než je popsáno v práci, dle potřeby. Po instalaci a nastavení těchto komponentů je ještě potřeba provést následující:

- Rozbalit obsah souboru **provtool.zip** (umístěn na DVD nosiči) do kořenového adresáře webového serveru (typicky /var/www/html).
- Rozbalit obsah souboru **configs.zip** (umístěn na DVD nosiči) a složku **prov** umístit do kořenového adresáře webového serveru, obsah složky **tftpboot** umístit do kořenového adresáře TFTP serveru (typicky /var/lib/tftpboot).

Po provedení těchto kroků by již měl být nástroj pro správu koncových zařízení a PBX Asterisk dostupný a funkční.

V případě problémů je doporučeno zkontrolovat protokolovací soubory všech komponentů na výskyt případných chyb, viz ukázka B.2.

Ukázka B.2: Umístění jednotlivých protokolovacích souborů.

```
PBX Asterisk:
/var/log/asterisk/messages

DHCP server:
/var/log/dhcpd.log

Syslog:
/var/log/syslog

HTTP server a PHP:
/var/log/httpd/logs
/var/log/apache/
```