Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalářská práce

Extrakce VOIP dat ze síťového provozu

Filip Šuster

Vedoucí práce: Mgr. Jan Starý, Ph.D.

12. května 2015

Poděkování

Poděkovat bych chtěl především vedoucímu bakalářské práce Mgr. Janu Starému, Ph.D. za pravidelné konzultace, odborné vedení a připomínky během tvorby práce. Dále děkuji Ing. Ivo Fišerovi za odborné rady o VoIP telefonii a v neposlední řadě bych rád poděkoval mé rodině a přítelkyni za podporu při tvorbě práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 12. května 2015 .....

Odkaz na tuto práci

Šuster, Filip. *Extrakce VOIP dat ze síťového provozu*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato bakalářská práce se zabývá implementací aplikace, která v lokální počítačové síti odposlouchává hlasové hovory přenášené IP protokolem (VoIP telefonie). Aplikace naslouchá na síťovém rozhraní a detekuje hovory signalizované SIP protokolem podle zadaných kritérií. Následně extrahuje samotné audio pakety do souborů a ukládá metainformace o hovorech v databázi. Audio pakety jsou poté zpracovány do zvukového formátu, metainformace jsou uživateli přístupné skrze webové rozhraní. Konverze hrubých audio dat do zvukových formátů a tvorba webového rozhraní není cílem této práce, spolu s pracemi kolegů však tvoří komplexní celek. Součástí práce je programátorská a uživatelská dokumentace.

Klíčová slova jazyk C, UNIX, extrakce hlasových dat, telefonní hovory, monitoring síťového provozu, VoIP telefonie, protokol SIP

Abstract

This Bachelor's Thesis deals with the implementation of an application for monitoring voice communication transferred by IP Protocol (VoIP telephony). The application monitors at the network interface and detects calls signalled by SIP protocol, according to given criteria. As a result, audio packets are extracted to files and metainformation about calls is stored in a database. Audio files are then processed into a sound format, metainformation is accessible to users through a web interface. The conversion of the raw audio data into the sound format and the web interface is beyond the aim of this thesis, however, it makes an aggregate together with theses of my coleagues. The thesis includes the programmer's and user's documentation.

Keywords C language, UNIX, voice data extraction, phone calls, network monitoring, VoIP telephony, SIP

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Komunikace patří k základním pilířům lidské společnosti. Od vynálezu telefonu v roce 1876 uplynula již řada let a telekomunikace je od té doby stále nezbytnější součástí každodenního života. Forma v podstatě zůstává stejná, volající zvedne sluchátko a vytočí číslo volaného. Co se ale rapidně mění, jsou technologie. Stále se zvyšující rychlost Internetu a důraz na pohodlí uživatele činí hlasové spojení rychlejší a jednodušší. Telekomunikace v počítačových sítích (VoIP) patří dnes již ke standardním technologiím digitální telefonie a její význam stále roste. Tradiční analogový přenos hlasu je na ústupu před moderními technologiemi, které umožňují přenášet hlas prostřednictvím vysokorychlostního Internetu. To platí pro pevné i mobilní sítě. Například technologie Voice over LTE (VoLTE), která byla nasazena v roce 2012, umožňuje přenášet hlasové hovory vysokorychlostním Internetem v mobilních sítích.

Téměř každá společnost, která využívá technologii VoIP, potřebuje zajistit dohled nad vnitřní i vnější komunikací. S tímto monitorováním provozu se můžeme běžně setkat například v call centru, kde před začátkem hovoru zazní známé: „V rámci zkvalitnění našich služeb může být váš hovor monitorován.“ Existuje více důvodů, proč odposlechy provádět. Z právního hlediska, například dojde-li k serióznímu sporu mezi zákazníkem a zaměstnancem o věci dohodnuté po telefonu. Některé firmy (především banky a pojišťovny) mají dokonce povinnost hovory archivovat po několik let. Odposlechy jsou prováděny také kvůli bezpečnosti, kdy nikdo není anonymní a potenciální prohřešek je dohledatelný. A nakonec kvůli dohledu nad kvalitou hovorů, kdy například síťový administrátor může analyzovat důvody výpadku hovorů.

Práce má tři implementačně oddělené části, které tvoří komplexní celek. Mou úlohou je zachytávat pakety ze sítě a ukládat metainformace o hovorech. Kolega Josef Kučera pracuje na extrakci samotného zvuku z paketů[1] a kolega Viktor Robejšek vytváří webové uživatelské a administrátorské rozhraní[2].

Kapitola 1

Cíl práce

Cílem práce je vytvořit nástroj, který odposlouchává a monitoruje VoIP hovory. Umožňuje také filtrování hovorů podle zvolených kritérií a poskytuje statistiky. Aplikace je součástí komplexního řešení, které umožní například centralizovanou správu pomocí webového rozhraní a databáze nebo konverzi samotných audio paketů do zvukových formátů. Toto řešení si klade za cíl pomoci firmám, např. call centrům, ke zkvalitnění jejich služeb či ke zvýšení bezpečnosti. Práce je zaměřena na univerzálnost a vysokou přenositelnost kódu. Jednotlivé cíle této práce jsou následující:

- Implementovat aplikaci v jazyce C, která na síťovém rozhraní rozpozná, získá, uloží a monitoruje VoIP hovory s využitím existujících knihoven a nástrojů (libpcap, libosip).
- Umožnit spolupráci s databází, aby mohla být aplikace součástí komplexního řešení.
- Rozeznávat nahrávací pravidla z databáze pro možnost filtrování hovorů.
- Zařídit vysokou přenositelnost kódu, aby mohla být aplikace nasazena na co nejvíce UNIXových systémech.
- Vytvořit programátorskou dokumentaci a manuálovou stránku.
- Program řádně otestovat.

Kapitola 2

VoIP telefonie

2.1 Úvod do technologie VoIP

Kapitola čerpá ze zdrojů [3, 4, 5, 6].

Voice over Internet Protocol (VoIP), nazýváno také IP telefonie, je technologie umožňující přenos hlasové komunikace a multimediálních relací pomocí datové sítě.

Technologie VoIP vznikla v Izraeli v malé firmě Vocaltec v roce 1995. Produkt InternetPhone této firmy umožňoval komunikaci mezi dvěma účastníky pomocí počítače a mikrofonu. O tři roky později se již dalo volat z počítače na telefon i z telefonu na telefon. Zpočátku byly hovory zdarma, avšak volající si museli vyslechnout několik reklam, než došlo ke spojení. V roce 1998 bylo zastoupení VoIP hovorů pouze 1 %, avšak o pět let později již čtvrtina všech hovorů.

V dnešní době se IP telefonie hojně používá ve firmách a vysokých školách, rozšiřuje se ale také do malých firem a domácností. Díky stále se zvyšující rychlosti internetového připojení vznikly také sesterské technologie jako VoWLAN[1] nebo novější VoLTE[2].

VoIP technologii lze provozovat v jakékoli síti s podporou IP protokolu, tedy nejen v Internetu, ale i v privátních či firemních sítích (Intranet). Je to druh digitální telefonie, neboť pro přenos převádí analogový hlas do digitální formy. Provozování IP telefonie vyžaduje kvalitní síťové připojení, protože je, stejně jako ostatní služby reálného času, citlivá na rychlost a ztrátovost spojení. Hlasové pakety by neměly být zahazované, příliš zpožděné nebo mít velké rozdíly ve zpoždění (jitter). Na síťových prvcích je často nutné zajistit dostatečnou kvalitu služeb (QoS), aby VoIP pakety byly upřednostňovány před jinými, méně časově náročnými službami.

[1] Technologie umožňující přenos hlasu pomocí Wi-Fi.

[2] Technologie, která přenáší hlas prostřednictvím vysokorychlostního Internetu v mobilních sítích.

Uživatel, který chce využívat službu VoIP, potřebuje IP telefon, počítač s klientským software nebo klasický analogový telefon s převodníkem. Může se dovolat na jakékoli telefonní číslo, ať už pevné, mobilní či zahraniční. To zajišťují tzv. VoIP brány, které převádějí IP telefonii na analogovou, digitální nebo GSM (mobilní) telefonii. Záleží ovšem na službách, které jsou uživateli VoIP poskytovatelem umožněny.

Jednoznačnou výhodou VoIP oproti tradičním telefonním sítím je úspora finančních prostředků (nižší poplatky za hovorné, žádné poplatky za vedení pevné linky). VoIP také umožňuje více, než jen volání a posílání textových zpráv. V dnešní době bohaté nabídky komunikačních médií nároky uživatelů rostou. Lidé si chtějí posílat fotografie, provádět videohovory apod. VoIP tyto služby s integrací dalších protokolů a aplikací umožňuje. Další výhodou je přenositelnost telefonu (nebo jiného média). Tradiční telefonní systém přiděluje číslo fyzickému telefonu a místu, tudíž není možné přenést telefon na jiné místo a bez změny čísla ho používat. IP telefon může být přenášen libovolně, stačí zajistit příslušnou IP konektivitu. Přenositelnost samotného telefonního čísla je také velkou výhodou. Uživatel může použít stejné číslo ve více telefonech (nebo jiných médiích), stačí vždy daný telefon před použitím zaregistrovat.

2.2 Právní pozadí odposlechu hovorů

Implicite platí, že odposlechy hovorů nejsou přípustné kvůli právu na soukromí. Zákon o ochraně osobních údajů 101/2000 Sb. upravuje právo člověka být informován o zpracování svých osobních údajů, včetně osobních údajů vázaných na odposlech a záznam telekomunikačního provozu.

Na druhou stranu, § 97 zákona č. 127/2005 Sb. o elektronických komunikacích ukládá povinnost provádět odposlechy hovorů právnickým nebo fyzickým osobám, které zajišťují veřejnou komunikační síť nebo poskytují veřejně dostupnou službu elektronických komunikací, a to zejména v případech, kdy se řeší odposlechy prováděné v rámci probíhajících trestních řízení.

Podle § 86 Nového občanského zákoníku č. 89/2012 Sb. nelze bez svolení člověka pořizovat o jeho soukromí zvukový záznam. V obchodních vztazích to pak znamená, že o nahrávání hovorů musí být účastníci hovorů předem informováni a musí k němu dát svolení. To se vztahuje např. na smlouvy uzavírané po telefonu.

Kapitola 3

Signalizační protokoly

V Internetu existuje mnoho aplikací, které potřebují vytvořit a spravovat spojení mezi účastníky. Spojením se myslí výměna multimediálních dat jako video, zvuk či textové zprávy. Obecně signalizační protokoly slouží k předávání informací o uskutečnění a ukončení spojení, dále se však budu zaměřovat na signalizaci VoIP hovorů, především na protokol SIP.

3.1 Protokol SIP

Kapitola čerpá především z RFC 3261 [7].

3.1.1 Popis protokolu

Dominantním protokolem pro signalizaci hovorů ve VoIP sítích je protokol SIP (Session Initiation Protocol). Je to textově orientovaný protokol aplikační vrstvy podobný protokolu HTTP. Ve VoIP se typicky používá pro signalizaci hovorového spojení. Samotné hlasové pakety jsou poté přenášeny pomocí dohodnutého transportního protokolu, např. RTP (Real-time Transfer Protocol [8]). Protokol SIP umožňuje následující signalizační funkce [9]:

- Nalezení koncového bodu
- Kontaktování koncového bodu a zjištění jeho zájmu o spojení
- Výměna metadat, která jsou potřebná k uskutečnění spojení
- Modifikace existujícího spojení
- Ukončení existujícího spojení

SIP byl také rozšířen o výměnu informací o stavu účastníka, je-li dostupný (online) nebo nedostupný (offline). Toto rozšíření zahrnuje funkce:

- Vysílání informace o stavu účastníka
- Žádost o informaci o momentálním stavu účastníka

Ve VoIP je protokol SIP používán v kombinací s protokolem SDP (Session Description Protocol), který zajišťuje vyjednání komunikačních možností. Díky tomu si účastníci spojení dohodnou parametry multimediálního přenosu, jako například kodek hovoru či IP adresu. Více o SDP v sekci 3.1.4.

SIP používá pro identifikaci zdroje SIP-URI (Uniform Resource Identifier), který používá schéma „sip:“ nebo „sips:“, pokud se jedná o šifrované spojení. Schémata jsou popsána v `RFC 2396`. SIP-URI je podobné e-mailové adrese a v obecné rovině vypadá takto:

```
sip:uživatel:heslo@hostitel:port;uri-parametry?hlavičky
```

Ukázkový příklad:

```
sip:alice@atlanta.com
```

Popis jednotlivých částí SIP-URI je následující:

- *uživatel* – Identifikátor zdroje (klienta). Typicky telefonní číslo, může ovšem obsahovat jakýkoliv řetězec znaků. Například v pobočkových sítích se podle rozsahu používá kratší číselný identifikátor nebo jméno (10, 123, Alice apod.).
- *heslo* – Heslo klienta. Nedoporučuje se takto používat, protože je přenášeno v otevřené formě.
- *hostitel* – Cílový server (poskytovatel SIP služeb), kam je požadavek zaslán. Jedná se o doménu nebo IP adresu hostitele. Toto je jediné povinné pole SIP-URI (vyjma schématu).
- *port* – Číslo portu, kam se požadavek zasílá.
- *uri-parametry* - Mohou ovlivnit zpracování požadavku.
- *hlavičky* – Mohou obsahovat některé další informace, zasílají se ve formátu `název=hodnota`.

3.1.2 SIP zprávy

SIP zprávou se myslí buď požadavek (Request) klienta na server, nebo odpověď (Response) serveru klientovi. Důležitý pojem je User Agent (UA), což je

jakékoliv hardwarové zařízení nebo software, který je schopný přijímat požadavky SIP protokolu. V SIP dialogu[3] se rozlišují pojmy UAC (UA Client) a UAS (UA Server), zkráceně klient a server. V průběhu dialogu se tyto role mění (více v sekci 3.1.3 Typická relace). Zprávy jsou rozeznatelné podle první řádky, kde má každá svou syntax.

3.1.2.1 Hlavičky zpráv

Hlavičky (Header Fields) protokolu SIP nesou podrobné informace o zprávách. Jsou podobné hlavičkám protokolu HTTP jak syntakticky, tak sémanticky. Mají formát `název: hodnota`, přičemž je dovoleno libovolné množství bílých znaků. Na pořadí hlaviček nezáleží. Některé jsou povinné, jiné volitelné. Zde uvádím seznam těch nejdůležitějších:

- `Call-ID` – Jednoznačný identifikátor dialogu (série zpráv). Měl by být globálně unikátní, každý UA musí zaručit, že žádný jiný UA nevygeneruje stejné `Call-ID`. Ke generování je doporučeno využít kryptografické funkce. Často mají formát „lokální-ID@hostitel“. `Call-ID` je case-sensitive (rozlišuje malá a velká písmena), porovnává se bajt po bajtu.

- `To` – Specifikuje požadovaného adresáta žádosti (volaného účastníka). Většinou obsahuje SIP-URI volaného, může ale obsahovat i jiná schémata (např. schéma „tel“ popsané v `RFC 2806`).

- `From` – Označuje odesílatele požadavku. Obsahuje SIP-URI a volitelné zobrazovací jméno. Hlavička musí obsahovat parametr `tag` zvolený UAC. Ten spolu s `Call-ID` identifikuje daný SIP dialog.

- `Via` – Určuje cestu, kterou požadavek prošel. Každý UAC musí přidat `Via` hlavičku k odeslané žádosti. Pokud již žádost `Via` hlavičku obsahuje (např. když proxy server[4] přesměrovává požadavek), přidá UAC na konec seznamu novou hlavičku `Via`.

- `Route` – Seznam proxy serverů, přes které se má směrovat daný požadavek.

- `CSeq` – Slouží k identifikaci pořadí příchozích zpráv. Skládá se ze sekvenčního čísla a z názvu metody. Metoda se musí shodovat s metodou některého z požadavků.

[3] Dialog reprezentuje konkrétní peer-to-peer spojení mezi dvěma UA.

[4] Proxy server nebo také SIP server směruje požadavky, provádí autentizaci uživatelů a spravuje spojení mezi koncovými UA nebo dalšími proxy servery. Je tedy prostředníkem pro komunikaci.

- `Max-Forwards` – Omezuje počet směrování požadavku. Obsahuje číslo, které se při každém „skoku“ zmenší o jedničku. Pokud hodnota dosáhne hodnoty 0 a nebyla ještě doručena adresátovi, bude žádost odmítnuta kódem 483 (Too Many Hops).
- `Contact` – Obsahuje SIP-URI specifického UA, je povinná v INVITE žádosti a specifikuje, kde se UA nachází (určuje tedy „kde“ je uživatel, narozdíl od hlavičky `From`, která určuje „kdo“ je uživatel).
- `Content-Type` – Určuje typ dat zasílaných v těle požadavku (tělo typicky obsahuje SDP zprávu).
- `Content-Length` – Délka dat v těle požadavku.

3.1.2.2 Žádosti

Typická INVITE žádost vypadá takto:

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(SDP obsah je skrytý)
```

Žádosti jsou zasílány klientem (UAC) a jejich první řádka vypadá následovně:

`Metoda␣Cílová-SIP-URI␣Verze-SIPu␣\r\n`

Cílová SIP-URI je ta, která je uvedena v hlavičce `To`. Výjimku tvoří metoda REGISTER (jejíž cílová SIP-URI například nemusí obsahovat část *uživatel*) nebo některé speciální případy. Verze SIPu může nabývat hodnot „SIP/2.0“ nebo „SIP“ (verze 1.0). Verze 1.0 je zastaralá a `RFC 3261` specifikace ji nedovoluje používat.

`RFC 3261` definuje šest základních metod:

- REGISTER slouží k registraci klienta na registračním serveru (registrar). Je prováděna periodicky, např. každou minutu a informuje SIP

server o pozici klienta. Pokud například server zachytí INVITE žádost pro tohoto klienta, ví, kam ji má zaslat.

- INVITE specifikuje klientův požadavek o navázání spojení. Obsahuje informace o volajícím, zpravidla včetně přiložené SDP zprávy, kterou klient udává své komunikační možnosti. Toto pozvání může být přijato nebo odmítnuto volajícím nebo proxy serverem. Pokud volaný přijme pozvání, dostane volající odpověď s kódem 2xx[5]. Pokud volaný nebo proxy server odmítne, volající obdrží zprávu s kódem jiným než 2xx, záleží na důvodu odmítnutí. Po zaslání INVITE se relaci říká také dialog.

- ACK slouží jako reakce na odpovědi s kódem 200-699, poté, co vznikl dialog pomocí INVITE. SIP totiž, stejně jako např. TCP, implementuje tzv. three-way-handshake. Minimální sekvence zpráv potřebná k započetí hovoru je metoda INVITE, odpověď 200 OK a metoda ACK.

- CANCEL slouží ke zrušení požadavku INVITE, kdy ještě nedošlo k zahájení hovoru. Může být zaslán volajícím nebo proxy serverem. Typicky vzniká, pokud volající položí sluchátko, protože mu to volaný nezvedl.

- BYE metoda ukončuje probíhající hovor. Typicky vzniká položením sluchátka. Může ho zaslat volající nebo volaný (případně proxy server).

- OPTIONS slouží k dotazování serveru o jeho možnostech.

Další metody jsou definovány v jiných RFC, například NOTIFY, INFO nebo SUBSCRIBE. Jejich podrobnější popis zde neuvádím, funkcionalita mé aplikace na nich není závislá.

[5]Označení pro kódy 200-299.

3.1.2.3 Odpovědi

Typická OK odpověď vypadá takto:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.biloxi.com
   ;branch=z9hG4bKnashds8;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
   ;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

(SDP obsah je skrytý)
```

Odpovědi jsou zasílány serverem (UAS) jako reakce na klientskou žádost. Jejich první řádka vypadá následovně:

`Verze-SIPu␣Kód-odpovědi␣Fráze-odpovědi␣\r\n`

Verze SIPu je stejná jako u žádosti. Kód odpovědi je trojciferné číslo, které udává reakci na žádost. Fráze odpovědi je krátký textový popis, který specifikuje číselnou odpověď. Kód je určen k automatickému zpracování, kdežto fráze je spíše popisem pro uživatele a může být odlišná od standardní fráze. Například typická odpověď „SIP/2.0 200 OK“ je stejně validní jako „SIP/2.0 200 Everything's fine“. První cifra kódu udává kategorii odpovědi. Další dvě cifry nemají kategorizační funkci. Odpovědi s kódem 100-199 jsou označeny `1xx`, 200-299 jsou `2xx` atp. `RFC 3261` specifikuje šest kategorií odpovědí:

- `1xx` (průběh) – požadavek přijat a provádí se
- `2xx` (úspěch) – žádost úspěšně přijata
- `3xx` (přesměrování) – požadavek probíhá, ale pro splnění je třeba provést další akci
- `4xx` (chyba klienta) – požadavek je nevalidní nebo nemůže být serverem zpracován
- `5xx` (chyba serveru) – serveru se nepodařilo splnit klientův validní požadavek

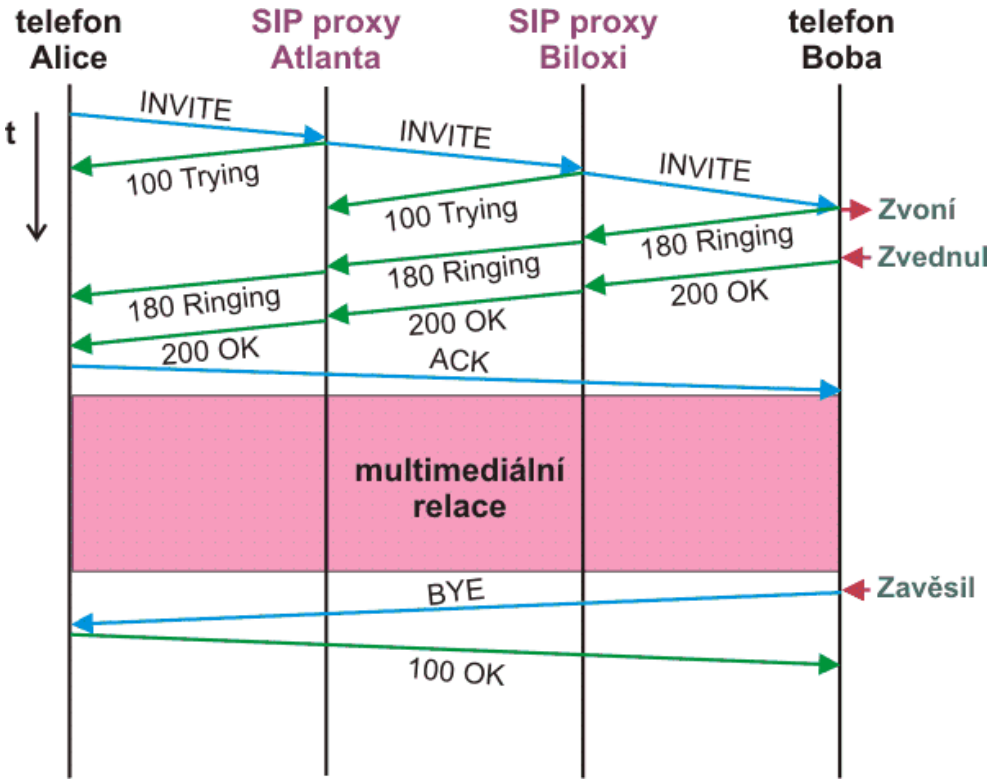
- `6xx` (obecná chyba) – žádost nemůže být splněna žádným serverem

Zde uvádím seznam nejtypičtějších odpovědí:

- 100 (Trying) – žádost byla přijata prvním serverem na cestě a další nespecifikovaná akce je vyžadována pro splnění požadavku
- 180 (Ringing) – cílový UA přijal INVITE a začal vyzvánět
- 183 (Session In Progress) – tato odpověď je použita pro zaslání dalších informací o hovoru, který je právě spojován (může například obsahovat SDP parametry, pomocí kterých je poté klientovi přehrána automatizovaná hláska „Vyčkejte prosím, volaný účastník právě hovoří.“)
- 200 (OK) – požadavek úspěšně splněn, může být reakcí na více žádostí (BYE, INVITE, REGISTER, ..)
- 401 (Unauthorized) – UAS nebo registrační server vyžaduje uživatelovu autentizaci
- 404 (Not Found) – server odmítl žádost, protože uživatel s cílovou SIP-URI neexistuje
- 407 (Proxy Authentication Required) – podobný kódu 401, ale je zasílán proxy serverem
- 408 (Request Timeout) – server nemohl odpovědět do určitého časového kvanta (například nemohl zastihnout adresáta žádosti)
- 480 (Temporarily Unavailable) – volaný byl úspěšně kontaktován, ale v současné době není dostupný (například není přihlášený nebo má na svém zařízení nastaveno „nerušit“)
- 486 (Busy Here) – volaný byl úspěšně kontaktován, ale není ochotný nebo schopný přijmout hovor (typická odpověď pro odmítnutý hovor)
- 487 (Request Terminated) – požadavek byl ukončen žádostí BYE nebo CANCEL
- 603 (Declined) – volaný byl úspěšně kontaktován, ale explicitně nechce nebo nemůže přijmout hovor; tento kód je použit, pokud volaný ví, že žádné koncové zařízení nepřijme hovor (to se projeví například při službě call-forking[6])

[6]Technika, která umožňuje zastihnout více UA téhož uživatele.

3.1.3 Typická relace



Obrázek 3.1: Typická SIP relace [10]

Obrázek 3.1.3 ukazuje typický příklad SIP komunikace mezi dvěma účastníky, Alicí a Bobem. V příkladu jsou také použity dva proxy servery, které žádosti předávají. Časová souslednost plyne shora dolů.

Alice zavolá Bobovi pomocí jeho SIP identity (SIP–URI), v tomto případě „sip:bob@biloxi.com“. `biloxi.com` je doména Bobova SIP poskytovatele. Alice má obdobné SIP-URI „sip:alice@atlanta.com“. Alice mohla takto zadat Bobovo SIP-URI nebo mohla například kliknout na odkaz ve svém adresáři.

Aličin UA tedy vygeneroval INVITE žádost spolu s přiloženou SDP zprávou. Jakmile žádost dorazila k první proxy Atlanta, proxy zjistila, že požadavek musí předat dál. Předala tedy žádost dalšímu proxy serveru a Alici poslala odpověď 100 Trying. Tento postup se znovu opakuje s proxy Biloxi. Poté, co žádost INVITE dorazila k Bobovu UA (v tomto případě telefonu), začal telefon vyzvánět a poslal zpět odpověď 180 Ringing. Tato odpověď byla proxy servery postupně doručena Alici. Jakmile Bob zvedl sluchátko, poslal jeho telefon odpověď 200 OK s přiloženou SDP zprávou s dohodnutými parametry

hovoru. Když dorazila tato odpověď k telefonu Alice, telefon zaslal potvrzení pomocí ACK, aby dokončil three-way-handshake. Jakmile Bob obdržel ACK, začal probíhat hovor (multimediální relace). Nakonec Bob telefon zavěsil, poslal žádost BYE a telefon Alice ukončil dialog odpovědí 200 OK (na obrázku je chyba, kód OK odpovědi je 200). Na obrázku je také vidět, že zprávy po ukončení hovoru již neputují přes proxy servery. To je teoreticky možné, v praxi se to však nepoužívá, protože proxy servery takto ztrácejí kontrolu nad hovory, což je nežádoucí například kvůli tarifikaci hovorů.

Všimněme si, že role UA Alice a Boba se v průběhu dialogu mění. Alice začala žádostí INVITE jako UAC a Bob odpovědí 180 Ringing jako UAS. V momentě, kdy Bobův telefon zaslal BYE, se jeho telefon stal UAC a telefon Alice při zasílání 200 OK změnil roli na UAS.

3.1.4 Protokol SDP

Kapitola čerpá především z RFC 4566 [11].

SIP protokol se zabývá vytvořením, upravováním a ukončováním relací. Samotné parametry relace jsou popsány protokolem SDP (Session Description Protocol), který v SIP zprávách slouží jako příloha. SDP protokol je také textový a obsahuje řádky `znak=hodnota`, kde znak je písmeno abecedy, které je case-sensitive a hodnota je textový řetězec. SDP specifikace umožňuje ustanovit více relací najednou, v SIPu však musí obsahovat popis pouze jedné relace (media description) [12]. SDP funguje na principu nabídka/odpověď (offer/answer). Nedochází tedy k vyjednávání parametrů v pravém slova smyslu. Nabízející pošle nabídku svých parametrů (např. audio kodeky), přijímající si jeden vybere podle svých možností a priorit [13]. Zde uvádím příklad a popis parametrů SDP zprávy, která je typicky přiložena k INVITE žádosti:

```
v=0
o=carol 28908764872 28908764872 IN IP4 100.3.6.6
s=-
t=0 0
c=IN IP4 192.0.2.4
m=audio 49170 RTP/AVP 0 1 3
a=rtpmap:0 PCMU/8000
a=rtpmap:1 1016/8000
a=rtpmap:3 GSM/8000
```

- `v` – verze protokolu, v současné době nabývá pouze hodnoty 0
- `o` – původce (originator) relace, obsahuje jméno, ID, číslo verze a IP adresu
- `s` – jméno relace (session), musí obsahovat alespoň jeden znak

- `t` – udává počáteční a koncový čas, jak dlouho je relace aktivní (na začátku jsou tyto hodnoty typicky nulové)
- `c` – obsahuje data pro spojení (connection) - typ spojení (IN jako Internet), verze adresy (IP4 nebo IP6), IP adresa spojení
- `m` – obsahuje popis relace (media description) – jméno relace (obecně jich může být více: audio, video, text, aj., pro telefonní hovory se typicky používá pouze audio, poté číslo portu, použité pro přenos dat, protokol pro přenos dat (typicky RTP/AVP[7]) a samotné podporované kodeky číslované podle RTP/AVP tabulky [14]
- `a` – specifikuje podrobněji atributy z `m=`

Popis relace pomocí protokolu SDP je typicky přiložen k INVITE žádosti, kdy volající nabídl své možnosti, a ke 200 OK odpovědi, kdy volaný zvolil jednu z možností. `RFC 3261` však dovoluje případ, kdy SDP přílohu volající přiloží až těsně začátkem relace pomocí ACK potvrzení [9]. Tento případ program nepodporuje, protože v praxi se používá jen zřídka. SDP může být taky přiložena v odpovědi 183 (Session In Progress), touto cestou se uživateli přehrávají netarifikované hlásky, například „Počkejte prosím, volaný účastník právě hovoří.“ Program takové hlásky nenahrává.

3.2 Ostatní protokoly

SIP je ve VoIP nejpoužívanější pro jeho otevřenost, jednoduchost a rychlost navazování relací. Kromě SIPu existují i jiné signalizační protokoly. Zde uvádím ty nejvýznamnější.

3.2.1 SCCP

SCCP (Skinny Call Control Protocol), někdy také nazývaný pouze Skinny, je proprietární protokol od společnosti Cisco, může být tedy použit pouze pro produkty Cisco. Je určen k signalizaci mezi Call Managerem a Cisco VoIP telefony. Narozdíl od SIPu používá pro signalizaci nativně TCP protokol. SCCP má velmi jednoduchou strukturu zpráv. Jelikož je ale proprietární, není zcela komplexně otevřeně popsán.

3.2.2 H.323

Signalizační protokol H.323 je starší než SIP protokol, byl navržen v roce 1996. Je také robustnější a složitější, H.323 zastřešuje celou rodinu menších protokolů pro multimediální přenos. Dnes je ve VoIP na ústupu před SIP

[7] Audio video profil.

protokolem, ale někteří poskytovatelé ho stále používají. H.323 se používá například pro videokonference.

Kapitola 4

Existující řešení

IP telefonie je dnes velice rozšířená a existuje již celá řada monitorovacích a nahrávacích programů. Některé se zabývají monitorováním VoIP provozu a sledují kvalitu hovorů, jiné například sledují SIP signalizaci a snaží se odhalit útoky (fraud). Zde uvádím příklady dostupných aplikací, které se zabývají VoIP hovory.

4.1 Podobné programy

4.1.1 Wireshark

Program Wireshark je populární síťový analyzátor s grafickým rozhraním. Dokáže dekódovat VoIP hovory živě i ze souboru a umožňuje i jejich přehrávání. Neumožňuje však spolupracovat s databází nebo filtrovat hovory na úrovni SIP signalizace. Wireshark působil spíše jako pomocník při vývoji programu.

4.1.2 Voipmonitor

Aplikace Voipmonitor poskytuje velmi komplexní řešení. Je to open-source síťový VoIP analyzátor s komerčním webovým uživatelským rozhraním (WEB GUI). Podporuje signalizační protokoly SIP a SCCP od společnosti Cisco. Poskytuje nejrůznější statistiky, které ukládá do MySQL databáze. Umožňuje také ukládat hovory do souborů formátu Pcap, dekódovat hovor a uložit ho ve formátu WAV. Má přívětivé uživatelské rozhraní, kde lze živě poslouchat hovory, zobrazovat nejrůznější statistiky, grafy apod. Toto WEB GUI je pouze licencované.

4.1.3 Homer

Homer je profesionální, robustní open-source nástroj pro monitorování VoIP provozu se SIP signalizací. Dokáže získávat data přímo ze SIP serverů (např.

Kamilio), může být tedy vhodným nástrojem pro správce VoIP ústředen. Vyvinul vlastní protokol HEP (Homer Encapsulation Protocol), který používá jako nadstavbu pro SIP a ostatní protokoly.

4.1.4 Oreka

Oreka je open-source program pro podniky s podporou SIP i SCCP protokolu. Nabízí webové rozhraní a podporuje mnoho ústředen (Asterisk, Cisco Call Manager, FreeSwitch a další). Je používán v call centrech pro monitorovací účely.

Aplikací k monitorování VoIP je celá řada, licencovaná drahá řešení i open-source programy. Mezi open-source řešeními jsme našli jen pár řešení, které byly podobné našemu návrhu.

Kapitola 5

Návrh řešení

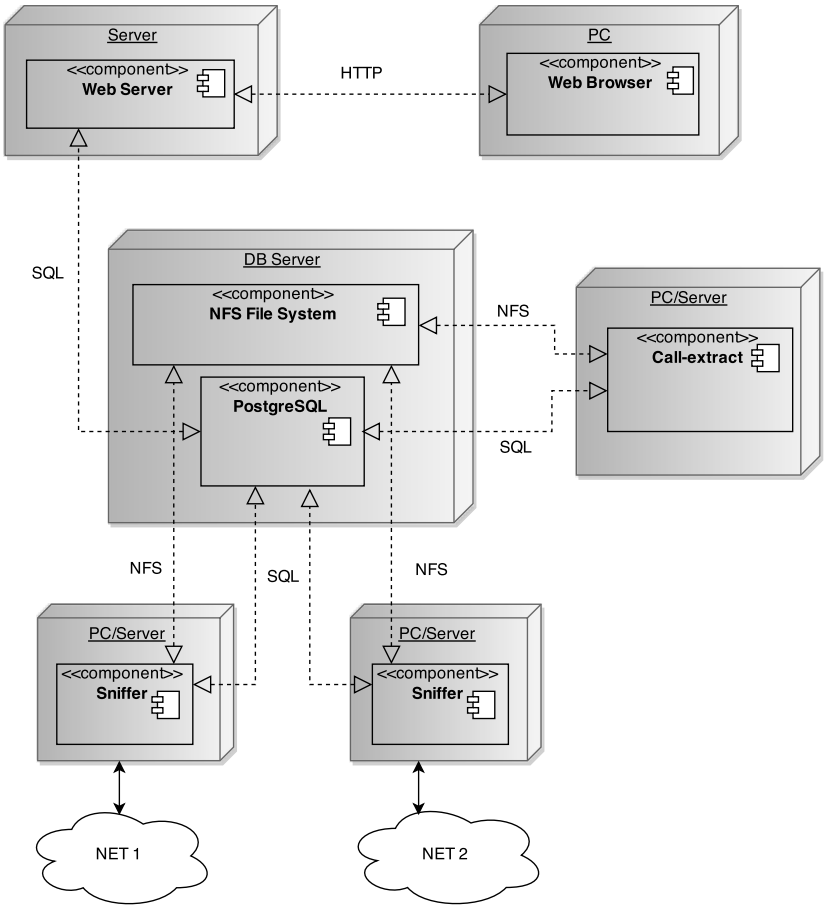
5.1 Komplexní návrh

Komplexní návrh je složen ze tří zmiňovaných bakalářských prací. Návrh sestává ze čtyř komponent:

- Sniffer – aplikace monitorující VoIP hovory na síti (předmět této práce)
- Call-extract – aplikace, která převádí zachycené VoIP pakety do audio formátu
- Database – sdílená databáze, kde jsou ukládány hovory a metainformace o nich
- Web interface – webové rozhraní pro uživatele a administrátora

Každá komponenta může běžet nezávisle na jiném stroji. Návrh je koncipován tak, že aplikací Sniffer může běžet více v různých sítích, například ve více pobočkách firmy. Každý Sniffer poslouchá ve své síti VoIP hovory, ukládá zachycené hovory do souborů a posílá metainformace do společné databáze. Aplikace Call-extract periodicky kontroluje, zda v databázi nepřibyly hovory pro konverzi do audio formátu (např. WAV). Pokud ano, vyzvedne si dávku hovorů a provede konverzi. Ve webovém rozhraní je skrze databázi možné sledovat různé stavy probíhajících hovorů. Například pokud bylo iniciováno volání (Sniffer zachytil INVITE), na webu jsou hned vidět některé informace (kdo komu volá, čas INVITE zprávy apod.). Sniffer tedy zasílá informace o hovorech tak, aby stav databáze co nejvíce odpovídal aktuálnímu stavu hovorů.

Diagram nasazení všech komponent vypadá následovně:



Obrázek 5.1: Komplexní návrh

Programy komunikují s databází pomocí jazyka SQL, pomocí protokolu NFS se soubory exportují do vzdáleného souborového systému. Prakticky by aplikace Sniffer měla zapisovat spíše na lokální systém souborů (aby stíhala zapisovat při větším množství hovorů) a exportovat soubory do vzdáleného systému souborů.

Kapitola 6

Instalace a konfigurace

6.1 Požadavky na systém

Aplikace byla úspěšně zkompilována a otestována na následujících operačních systémech a architekturách:

- OpenBSD 5.7 (i386/amd64/PowerPC/ARM)
- FreeBSD 10.1 (i386)
- Debian Wheezy 7.8 (amd64)

Dále je třeba zajistit dostupnost potřebných knihoven pro běh aplikace. Těmi jsou: libpcap (zachytávání paketů), libpq (spolupráce s PostgreSQL databází), libosip (parsování SIP/SDP zpráv).

6.2 Síťové požadavky

Aplikace Sniffer poslouchá na síťovém rozhraní (Network Interface Controller, zkráceně NIC). Je nutné, aby NIC podporoval promiskuitní mód, protože Sniffer potřebuje slyšet veškerý síťový provoz na LAN síti. NIC v základním módu totiž zahazuje pakety, které mu nejsou adresovány.

Dále je potřeba zajistit, aby byl kýžený provoz vůbec směrován do tohoto NIC. To je možné zajistit více způsoby. Jednak je možné připojit NIC do přepínače, který podporuje zrcadlení portů (port mirroring). Přepínač se nastaví tak, aby kýžený provoz přeposílal na port, kam je NIC zapojen. Druhá možnost je použít rozbočovač (hub) místo přepínače. Rozbočovač totiž veškerý provoz zasílá na všechny aktivní porty (nepodporuje přepínání podle MAC adresy). Rozbočovač se však v dnešních sítích již téměř nepoužívá. Další možností je umístit aplikaci Sniffer tam, kudy přirozeně prochází všechen provoz

(například do zařízení, které v lokální síti slouží jako brána do Internetu). Možností může být více, záleží na konkrétní topologii sítě.

6.3 Konfigurační soubor

Takřka veškerá konfigurace programu je popsána v konfiguračním souboru. Soubor obsahuje řádky `direktiva␣hodnota`. Prázdné řádky, stejně jako všechny nadbytečné bílé znaky jsou ignorovány. Řádek začínající znakem `#` je považován za komentář. Každé direktivě náleží právě jedna hodnota. Pokud je v souboru více stejných direktiv, je použita ta poslední. Zde uvádím malou ukázku:

```
# What interface to listen on.
# If not specified, pcap_lookupdev() finds one.
#iface eth0

# What port to listen to.
#port 5060

# How to name the individual call dumps.
name    %f_%t-%H%M%S.pcap

# Where call dumps will be stored.
# Remember that sniffer runs chrooted in user's home directory.
path    /calls/%Y-%m-%d-%H
```

Seznam všech direktiv je následující:

- `user` – dedikovaný systémový uživatel, pod kterým Sniffer poběží, až se vzdá práv superuživatele (více v sekci 6.4 Režimy spuštění), jeho domovský adresář je použit pro chroot[8] prostředí programu
- `group` – dedikovaná skupina, pod kterou Sniffer poběží, až se vzdá práv superuživatele
- `iface` – síťové rozhraní, na kterém Sniffer poslouchá provoz
- `port` – port, na kterém Sniffer poslouchá SIP signalizaci
- `snifferid` – ID instance Snifferu v databázi; Sniffer se dle tohoto ID registruje v databázi, stahuje si pravidla určená jemu a hovory jím nahrané

[8] Change root, prostředí s posunutým kořenovým adresářem.

jsou poté rozeznatelné od jiných Snifferů; ID Snifferu je třeba předem znát (první musí být záznam v databázi[9])

- `dbhost` – doménové jméno nebo IP adresa databázového spojení
- `dbport` – port, na kterém databáze naslouchá připojením
- `dbname` – jméno databáze
- `dbuser` – databázový uživatel
- `dbpass` – heslo databázového uživatele
- `name` – jméno nahraných hovorových souborů, může obsahovat symboly rozeznatelné funkcí `strftime(3)` a další vlastní symboly (podrobněji popsáno v manuálové stránce)
- `path` – do jakého adresáře budou ukládány hovorové soubory, může také obsahovat symboly rozeznatelné pomocí `strftime(3)`
- `mode` – jaká práva budou přiřazována hovorovým souborům
- `len` – časový údaj v sekundách, který udává, jak dlouho se má maximálně nahrávat jeden hovor
- `max` – kolik hovorů současně se může nahrávat
- `logto` – kam bude Sniffer logovat (více o dalších bodech v sekci 7.8 Logování)
- `logfacl` – facility pro logování používaná nástrojem Syslog
- `loglvl` – úroveň logování

Podrobnější popis jednotlivých hodnot direktiv včetně výchozích hodnot lze nalézt v přiložené manuálové stránce. Některé další konfigurační možnosti jsou předány programu z příkazové řádky.

6.4 Režimy spuštění

Program nemá žádné grafické rozhraní a spouští se z příkazové řádky. Typicky bude spuštěn jako daemon[10], poběží tedy na pozadí jako systémová služba. Program nabízí více režimů spuštění, přičemž režim Interakce s databází je označen jako primární, protože takto byl původně navržen pro spolupráci

[9] Sniffer se v databázi přidává pomocí funkce `newRecorder()`.

[10] Program, který je spuštěn dlouhodobě a není v přímém kontaktu s uživatelem.

s ostatními programy v komplexním návrhu. V režimech Interakce s databází a Standalone mód musí být program spouštěn pod superuživatelem (root). Otevírá totiž síťové rozhraní k odposlouchávání, což je privilegovaná operace.

6.4.1 Interakce s databází

Pro tento režim je program primárně určen, protože je zde využita jeho plná funkcionalita. Před spuštěním je třeba znát databázové ID Snifferu, které musí být předem v databázi zavedeno. Pokud nebylo uživatelem (z příkazové řádky) specifikováno, aby se nahrávaly všechny hovory, program si po spuštění z databáze stáhne nahrávací pravidla.

6.4.1.1 Nahrávací pravidla

Pravidla specifikují některé atributy hovoru a časové údaje, podle kterých se hovory budou, nebo nebudou nahrávat. U vyhodnocování pravidel záleží na pořadí a vyhodnocují se podle strategie first-match[11]. Shoda hovoru s pravidlem je definována konjunkcí všech atributů v tabulce kromě „pořadí“ a „nahrávat“, přičemž prázdné místo v tabulce znamená libovolnou hodnotu. Nenajde-li program u hovoru shodu s žádným pravidlem, hovor se nenahraje. V tabulce 6.1 uvádím ukázku pravidel a jejich slovní popis.

[11] První shoda hovoru s pravidlem je vyhodnocena.

pořadí	nahrávat	číslo volajícího	číslo volaného	minuta	hodina	den	měsíc	den v týdnu
1.	ano	-	906500900	-	8-16	-	-	-
2.	ano	15	10	-	-	-	-	1-3
3.	ne	-	800700600	-	-	-	-	-
4.	ano	123456789	987654321	00-45	12	-	-	-
5.	ne	-	-	-	-	8	5	-
6.	ano	-	-	-	-	-	-	-

Tabulka 6.1: Ukázka nahrávacích pravidel

Slovní popis pravidel je následující:

1. Nahrávat hovory každý den od 8 do 16 hod, kde číslo volaného je 906500900
2. Nahrávat hovory každé pondělí, úterý a středu, kde číslo volajícího je 15 a číslo volaného je 10 (takováto čísla jsou používána v pobočkových sítích)
3. Nenahrávat hovory, kde číslo volaného je 800700600
4. Nahrávat hovory každý den od 12.00 do 12.45, kde číslo volajícího je 123456789 a číslo volaného 987654321
5. Nenahrávat žádné hovory 8. května
6. Nahrávat všechny hovory

Poslední pravidlo zní „nahrávat všechno“. To je typické užití pravidel, kdy nejdříve jsou uvedena ta specifická a nakonec ta obecná, právě kvůli strategii first-match.

Následující ukázka kódu demonstruje, jak program pravidla vyhodnocuje.

```
for (i = 0; i < rules->count; ++i){
    /* do the call characteristics (dstnum, srcnum,..)
     * match the rule? */
    if (!match_call(call, rules->rule_array[i]))
        continue;

    /* does the invited time of the call match the time
     * part of the rule */
    if (!match_time(&(call->invited),
        rules->rule_array[i]))
            continue;

    /* everything matched, do we want the call? */
    if (rules->rule_array[i]->yes_no == 1)
        return 1;
    else
        return 0;
}
```

6.4.2 Offline mód

V tomto režimu program neprovádí živý odposlech sítě. Místo toho má na vstupu soubor, často nazývaný jako „dump“, který obsahuje záznam síťového provozu. Záznam může být z jakékoli sítě, kde proběhly nějaké VoIP hovory. V tomto režimu program nepotřebuje žádná privilegia. Program zpracuje dump soubor a nahraje metainformace o hovorech do databáze, jako by se odehrály na síti, kterou Sniffer poslouchá. Poskytuje tedy stejnou funkcionalitu jako v režimu Interakce s databází.

6.4.3 Standalone mód

V tomto režimu se program nepřipojuje k databázi a živě zachytává všechna volání. Z konfiguračního souboru jsou využity všechny direktivy kromě těch, které se týkají databáze. Toto je alternativní režim, neboť nevyužije plnou funkcionalitu programu (nikam neukládá metainformace o hovorech).

Kapitola 7

Implementace

7.1 Zvolené nástroje/knihovny

7.1.1 knihovna Pcap

Knihovna libpcap, zkráceně nazývaná Pcap, je standardní open-source nástroj pro zachytávání paketů ze sítě. Byla vytvořena již v roce 1994 ve výzkumném ústavu Univerzity v Berkeley [15]. Záměrem vývojářů bylo vytvořit přenositelné API (Application Programming Interface) nezávislé na operačním systému. API je určeno pro použití v jazyce C a C++, ale existuje mnoho rozšíření do ostatních jazyků (např. Perlu). Dnes je knihovna Pcap spravována komunitou Tcpdump Group [16], kde je dostupná dokumentace.

Důležitou součástí každého takového nástroje je paketový filtr. Síťové monitorovací programy jsou spuštěny v uživatelském módu (nikoliv kernel módu[12]), pakety tedy musí být kopírovány z kernelového prostoru do uživatelského. Efektivita tohoto kopírování je závislá právě na zvoleném paketovém filtru. Pcap využívá efektivní The BSD Packet Filter (BPF), který pracuje až stokrát rychleji než ostatní filtry (např. Ultrix Packet Filter). BPF například nespotřebuje takřka žádný výkon CPU na pakety, které neodpovídají nastavenému filtru [17].

Existují jiné nástroje, například NetFlow od společnosti Cisco, který poskytuje detailní pohled na síťovou komunikaci pomocí IP toků. Neposkytuje však celé pakety, které pro samotné nahrávání hovorů program potřebuje.

Pcap je možné využívat i v rychlých sítích (10 Gbps a více), výkon a ztrátovost paketů ovšem může být omezena hardwarem. Ztrátovost paketů je možné sledovat pomocí funkce `pcap_stats()`. Pokud je ztrátovost nenulová, je možné například použít kernelový modul `PF_RING`™ (dostupný pro Linuxové jádro 2.6.32 a vyšší), který používá efektivní kruhový buffer a poradí si s vysokými rychlostmi a velkým objemem dat na síti [18].

[12] Mód CPU, kdy má aplikace přístup ke všem jeho instrukcím.

Knihovnu Pcap jsem použil, protože je to standardní open-source nástroj pro zachytávání paketů a v podstatě k němu neexistuje vyhovující alternativa.

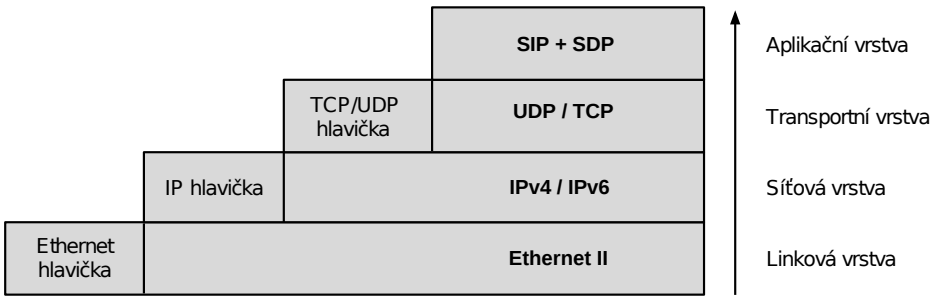
7.1.2 knihovna oSIP

Knihovna oSIP je stabilní a přenositelný nástroj, který implementuje SIP podle `RFC 3261`. Poskytuje jednoduché C API a potřebuje ke svému běhu pouze standardní C knihovnu. Z knihovny oSIP jsem použil její modul osip-parser2 pro rozebrání obsahu SIP a SDP zpráv, které je potřeba analyzovat pro ukládání a nahrávání hovorů. Knihovna neposkytuje příliš kvalitní programátorskou dokumentaci, což ale nepředstavuje příliš velký problém, neboť samotný kód je jednoduchý, dobře čitelný a komentovaný.

Existuje více různých nástrojů, které implementují SIP. Například PJSIP jsem zavrhl pro jeho nadbytečnou rozsáhlost. Sofia-SIP je velmi podobná oSIP. Mezi těmito dvěma nástroji jsem se rozhodoval, oba jsou velmi univerzální a mají malý objem kódu (cca 400 kB), nakonec jsem Sofia-SIP zavrhl, protože neposkytuje tak intuitivní API jako oSIP.

7.2 Rozebrání paketů

Knihovna Pcap poskytuje uživatelskému programu hrubá data. Program sám musí rozebrat síťové pakety a zjistit, jaký obsah skýtají. Obrázek 7.1 ilustruje, které protokoly na kterých vrstvách modelu TCP/IP jsou programem v rámci SIP signalizace podporovány.



Obrázek 7.1: Rozebrání paketů

7.2.1 Fyzická vrstva

Fyzická vrstva je nejnižší vrstva síťového modelu. Určuje, jak jsou spolu fyzicky propojená síťová zařízení. Tato vrstva je určena hardwarem (např. síťovou kartou) a programu je skryta. Přímo však ovlivňuje sousední linkovou vrstvu.

7.2.2 Linková/datová vrstva

Na linkové vrstvě program podporuje rámce standardu Ethernet II, který mezi standardy Ethernet v Internetu dominuje. Program nepodporuje zapouzdření standardu 802.3, který narozdíl od Ethernet II nenese informaci o protokolu vyšší vrstvy. 802.3 se používá spíše k servisním účelům v lokálních sítích, např. mezi směrovači. Program také podporuje standard 802.11 (Wi-Fi). Je však nutné, aby síťová karta resp. její systémový ovladač podporoval funkcionalitu FullMAC resp. SoftMAC, která zajistí, že programu jsou na linkové vrstvě poskytnuty rámce Ethernet II místo složitého zapouzdření standardu 802.11. Toto použití je určeno spíše pro testování, neboť Wi-Fi karty často v praxi nepodporují promiskuitní mód [19]. Program také pro testovací účely podporuje rozhraní loopback.

7.2.3 Síťová vrstva

Na síťové vrstvě program podporuje IPv4 a IPv6. Zde jsou prováděny základní validace paketu (CRC, kontrola hlaviček). Nutno říci, že program nepodporuje fragmentaci paketů. Paketový filtr knihovny Pcap neumožňuje efektivně takové pakety zachytávat. Jelikož SIP zprávy nemají omezenou velikost, může teoreticky k fragmentaci docházet, typicky se to však nestává. Obvyklá hodnota maximální přenosové jednotky (MTU) 1500 bajtů je dostačující, SIP zpráva pro VoIP signalizaci málokdy překročí 1300 bajtů. Pouze INVITE zpráva s přiloženým SDP obsahem a autentizačními údaji mívá více než kilobajt.

7.2.4 Transportní vrstva

Na transportní vrstvě program podporuje protokoly UDP a TCP. V SIP signalizaci jednoznačně dominuje UDP (User Datagram Protocol), který je nespojovaný a nezajišťuje spolehlivost doručení. Jeho jednoduchost a malá režie je nezbytnou výhodou v systémech proudového přenosu (stream) jako je VoIP. UDP se běžně používá jak v SIP signalizaci hovorů, tak v samotném přenosu audio paketů. Protokol TCP (Transmission Control Protocol) je v SIP signalizaci používán pro již zmíněné fragmentované pakety a pro šifrované hovory. Šifrované hovory program nepodporuje, jelikož se nedají jednoduše odposlouchávat, a navíc se v praxi příliš nepoužívají. Například v ČR je nepodporuje žádný veřejný VoIP poskytovatel. Šifrování má však podporu v akademickém prostředí. TCP může být použit i z jiných důvodů.

7.2.5 Aplikační vrstva

Konečně na aplikační vrstvě program rozpoznává SIP zprávy. Některé SIP zprávy nesou data SDP protokolu. K analyzování všech těchto dat program používá knihovnu oSIP a její modul osipparser2.

7.3 Zpracování SIP signalizace

Poté co program v paketu rozpozná SIP zprávu, analyzuje její obsah a rozhodne, jakou operaci provede. Z široké škály SIP žádostí a odpovědí program rozeznává pouze pro něj relevantní podmnožinu. Zde uvedu seznam těchto zpráv a přidružené reakce programu.

7.3.1 Žádosti a odpovědi

V této sekci ukazuji, jak program reaguje na SIP zprávy. Uvádím je v pořadí, jak vznikají při pokusu o spojení.

7.3.1.1 INVITE

INVITE označuje žádost o navázání spojení a obsahuje SDP informace o volajícím. Klient, který posílá INVITE, vygeneruje globálně unikátní `Call-ID`, jednoznačný identifikátor dialogu. Pomocí tohoto identifikátoru se rozliší, ke kterému dialogu zachycené SIP žádosti a odpovědi patří. Jakmile program zachytí INVITE žádost, nejprve zjistí, vyhovuje-li tento hovor zadaným nahrávacím pravidlům. Pokud ano, vytvoří strukturu `TCALL` (více o struktuře v sekci 7.4), naplní ji daty, která jsou ze zprávy k dispozici, a uloží ji do globálního seznamu hovorů.

7.3.2 RINGING

Odpověď s kódem 180 označována jako RINGING indikuje takový stav, kdy byl zastižen volaný a jeho UA začal vyzvánět. V tomto okamžiku program pouze změní stav volání, což může později sloužit k rozlišení mezi nepřijatým a neuskutečněným hovorem.

7.3.3 OK

Odpověď OK s kódem 200 může být zaslána na různé žádosti, zpravidla na REGISTER, INVITE a BYE. Pomocí pole `CSeq` v těle zprávy je možné zjistit, ke které žádosti odpověď patří. V mém programu rozpoznávám pouze OK odpověď na žádost INVITE, která indikuje úspěšně navázaný hovor. Jakmile ji program zachytí, provede následující kontrolní mechanismus:

1. Zkontroluje, jestli odpověď patří k žádosti INVITE
2. Zkontroluje, zda již eviduje volání s patřičným `Call-ID`
3. Zjistí, jestli se nejedná o duplikovanou odpověď

Pokud byly všechny podmínky splněny, bude se provádět nahrávání hovoru. Jak je vidět v typické relaci, v momentě, kdy je volajícím zachycena

OK odpověď, zašle ještě ACK potvrzení, a multimediální relace, tedy hlasový hovor, může začít. Zde bych rád zmínil, že program potvrzovací ACK žádosti nedetekuje, protože je to v tomto případě nadbytečná režie. Nahrávání samotných hovorů provádí programem vytvořené podprocesy, o tom je kapitola 7.5 Vytváření obslužných procesů.

7.3.4 CANCEL

Žádost CANCEL slouží ke zrušení předchozí INVITE žádosti klienta. CANCEL může zaslat jednak volající, pokud ukončí volání ještě před započatým hovorem, a jednak proxy ústředna. Jakmile program zachytí žádost CANCEL, nalezne přidružené volání v globálním seznamu (opět podle `Call-ID`) a hovor smaže.

7.3.5 BYE

Žádost BYE ukončuje probíhající hovor a může ji zaslat volající nebo volaný. Zachytí-li program BYE žádost, označí hovor jako úspěšně nahraný, ukončí obslužný hovorový proces, který hovor nahrával, a smaže hovor z globálního seznamu.

7.4 Struktura TCALL

Hlavní datovou strukturou v programu je `TCALL`, která udržuje všechny potřebné informace o volání. Je popsána v tabulce 7.1.

Řetězce, které mají velikost 256 nebo 64 bajtů, reprezentují takové atributy SIP zprávy, které nemají podle `RFC 3261` omezenou velikost. Statická alokace celé struktury je však nutná, aby bylo možné přeposílat strukturu databázovému procesu (pomocí roury). 256 bajtů je vyhovující kompromis pro `call_id`, `dst_sip_uri` a `src_sip_uri`. Atributy `src_num` a `dst_num` jsou získány ze SIP-URI (řetězec před znakem `@`), které typicky obsahují telefonní čísla, mohou ale obsahovat i řetězce (např. alice123). 64 bajtů je vyhovující maximální délka. Časové atributy datového typu `timeval` nesou informace v Unix time[13] s přesností na mikrosekundy. Poté se převádějí do čitelné formy (například `2015-04-25 13:55:02.468665`). Neprázdný atribut `rejected` indikuje, že bylo volání nebo hovor odmítnut (nerozlišuje se, jakým způsobem). Všechny časové údaje jsou ukládány (a poskytovány databázi) v UTC[14], neboť jsou takto poskytovány od knihovny Pcap. Databázová pravidla taktéž musí být v UTC pro správné zpracování Snifferem. Všechny časové atributy `TCALL` označují časy, kdy Sniffer danou SIP zprávu zachytil, nikoliv kdy tato zpráva vznikla. Atribut `sip_status` je výčtového typu a nabývá hodnot UNKNOWN, INVITE, RINGING, OK, CANCEL a BYE.

[13] Počet sekund, který uplynul od 1970-01-01 00:00:00 UTC.
[14] Coordinated Universal Time - koordinovaný světový čas.

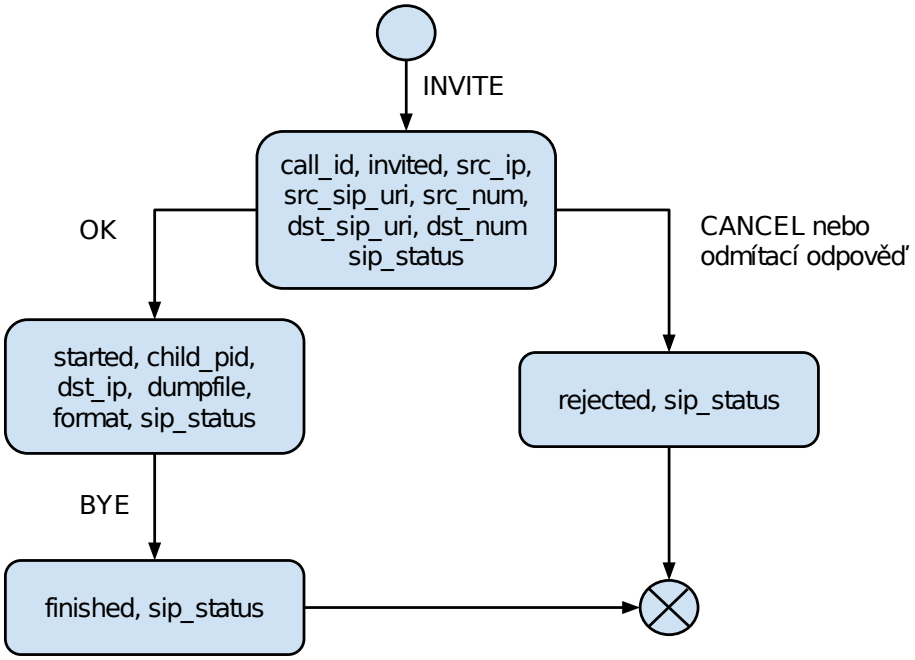
Typ/Délka[B]	Název	Popis
`char[256]`	`call_id`	Call-ID hovoru
`pid_t`	`child_pid`	Číslo hovorového procesu, který hovor nahrává
`SIP_STATUS`	`sip_status`	Hodnota udává aktuální stav volání
`char[PATH_MAX]`	`dump_file`	Název souboru, kam se zapisují pakety hovoru
`char[INET6_ADDRSTRLEN]`	`src_ip`	Zdrojová IP adresa
`char[INET6_ADDRSTRLEN]`	`dst_ip`	Cílová IP adresa
`unsigned int`	`src_port`	Zdrojový port
`unsigned int`	`dst_port`	Cílový port
`char[256]`	`src_sip_uri`	Zdrojová SIP-URI
`char[256]`	`dst_sip_uri`	Cílová SIP-URI
`char[64]`	`src_num`	Číslo volajícího
`char[64]`	`dst_num`	Číslo volaného
`int`	`format`	Kodek hovoru podle RTP/AVP
`timeval`	`invited`	Čas zachycení INVITE žádosti
`timeval`	`started`	Čas zachycení OK odpovědi (začal se nahrávat hovor)
`timeval`	`finished`	Čas zachycení BYE žádosti (nahrávání hovoru skončilo)
`timeval`	`rejected`	Čas zachycení žádosti CANCEL nebo odpovědi z množiny {404, 480, 486, 487, 603}
`timeval`	`crashed`	Nepoužito

Tabulka 7.1: Struktura TCALL

7.4.1 Životní cyklus

Na obrázku 7.2 je vidět, jak vzniká a zaniká struktura `TCALL`.

Ohodnocení hrany udává SIP zprávu, kterou hlavní proces zachytil. Struktura `TCALL` tedy vzniká při zachycené žádosti INVITE (neduplikované). Atributy v uzlech značí, že byly vyplněny nebo změněny. V každém uzlu je také struktura zaslána databázovému procesu pro aktualizaci hovoru v databázi. Pokud byla k danému volání zachycena pouze INVITE žádost, je v programu nastavený mechanismus, který takové instance `TCALL` maže, pokud od zachycení INVITE uplynulo více než 60 sekund (tato volání jsou označována jako „zombie“).



Obrázek 7.2: Životní cyklus struktury TCALL

7.5 Vytváření obslužných procesů

7.5.1 Procesy vs. vlákna

Rozhodoval jsem se, zda paralelismus v programu provedu pomocí procesů nebo vláken. Vlákna mají menší režii a pro povahu úlohy by se mohly zdát dostačující. Nakonec jsem se však rozhodl použít procesy, a to z důvodu bezpečnosti v programu. Program totiž musí být spuštěn s privilegii superuživatele (root), aby mohl síť odposlouchávat. Obslužné procesy ale tato privilegia již nepotřebují, takže se jich mohou vzdát (privilege separation). Podobně je řešený například webový server Apache, u kterého jsem se inspiroval.

7.5.2 Hlavní proces

Hlavní proces v reálném čase odposlouchává SIP signalizaci a provádí její zpracování. Pomocí funkce `pcap_loop()` zachytává příchozí pakety, hledá v nich SIP zprávy, operuje s globálním bufferem hovorů, vytváří hovorové procesy a posílá informace o hovorech databázovému procesu.

7.5.3 Databázový proces

Je-li program spuštěn v režimu spolupráce s databází, na začátku vytvoří obslužný databázový proces[15] (také zvaný db worker). Ten po celou dobu běhu přijímá požadavky od hlavního procesu a předává je databázi. Samotné předávání informací je řešeno pomocí jednosměrné roury[16]. Tento proces nepotřebuje privilegia superuživatele, takže se jich vzdá hned po vytvoření.

Databázový proces přijímá od hlavního procesu strukturu `TCALL`. Podle hodnoty atributu `sip_status` se rozhoduje, kterou rutinu provede. Všechny tyto rutiny volají databázové funkce připravené kolegou Robejškem, nevolají tedy přímo SQL příkazy INSERT nebo UPDATE. Všechny rutiny používají `Call-ID` jako primární klíč, vkládají také Sniffer ID. Atributy struktury `TCALL`, které se v každé rutině používají, lze vidět na obrázku 7.2. V ukázce kódu 7.3 lze vidět hlavní smyčku databázového procesu a hodnoty `sip_status`, podle kterých volí patřičnou rutinu.

7.5.4 Hovorový proces

Hovorové procesy mají za úkol samotné nahrávání hovorů. Pro každý hovor je vytvořen samostatný proces (také zvaný call worker), který zachytává pakety hovoru mezi IP adresami a porty účastníků. Pakety okamžitě píše do souboru formátu Pcap. Hovorový proces má tedy pouze za úkol naplnit soubor pakety.

7.6 Spolupráce procesů

Na obrázku 7.4 je vidět, jak spolu procesy komunikují. Hlavní a hovorový proces komunikují pomocí signálů (více v sekci 7.7 Signály). Poté co hovor, potažmo hovorový proces skončí, hlavní proces si od něj vyzvedne návratovou hodnotu a pošle příslušná data databázovému procesu pomocí roury. Jak vidíme, pouze hlavní proces potřebuje privilegia superuživatele a navíc přepíná mezi uživatelovým a superuživatelovým EUID[17].

[15] Hlavní proces poslouchá SIP signalizaci v reálném čase a nemůže se zdržovat jinou činností, než nezbytně nutnou. Databázovému spojení je tedy dedikován samostatný proces.

[16] Mechanismus IPC (Inter-Process Communication), přeposílání dat mezi procesy pomocí soketu a funkcí `read` a `write`.

[17] Podle efektivního UID uživatele (EUID) se řídí jeho právo k některým systémovým operacím.

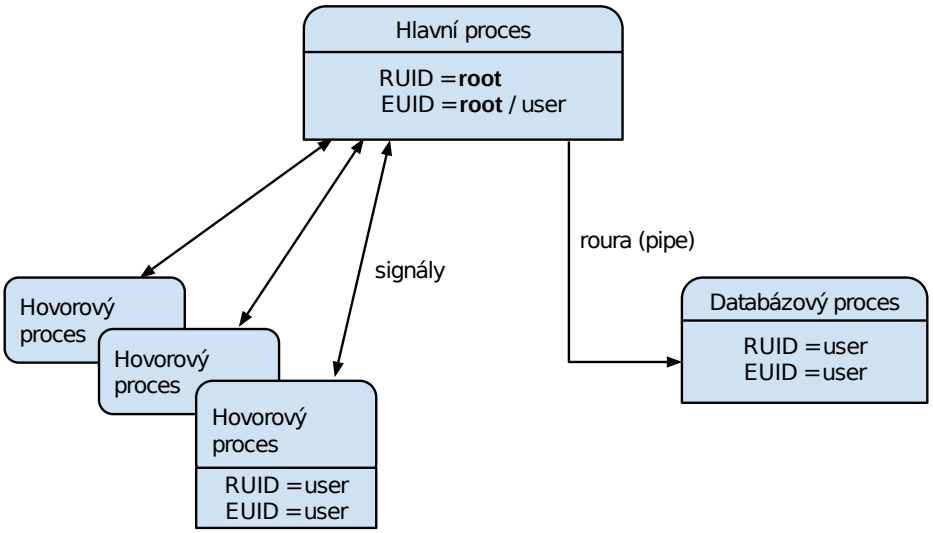
```
while ((b = read(g_fd[0], &buf, sizeof(TCALL))) > 0){
    if (b != sizeof(TCALL)){
        xlog(LOG_ERR, "DB worker BAD read "
                      "(only %d bytes)\n", b);
        continue;
    }

    call = (TCALL *)buf;

    /* basic validation of received TCALL */
    if (call->src_port < 1 || call->src_port > 65355){
        xlog(LOG_ERR, "DB worker BAD read "
                      "(broken TCALL structure)\n", b);
        continue;
    }

    switch (call->sip_status){
        case INVITE:
            insert_call(call, db_conn);
            break;
        case OK:
            update_started_call(call, db_conn);
            break;
        case CANCEL:
            update_rejected_call(call, db_conn);
            break;
        case BYE:
            update_finished_call(call, db_conn);
            break;
        default:
            break;
    }
}
```

Obrázek 7.3: Ukázka kódu databázového procesu



Obrázek 7.4: Architektura procesů

7.7 Signály

Komunikace mezi procesy a jejich ukončování je řešeno pomocí signálů. Každý proces (hlavní, hovorový, databázový) má mírně odlišné chování pro různé signály. Je proto nutné pro každý zvlášť zavést tzv. signal handler[18]. To zařídí funkce `sigaction(2)`. V tabulkách níže jsou uvedeny reakce na příslušné signály.

Při ukončujících signálech se uvolní všechny alokované prostředky. Ostatní signály, které nejsou v tabulkách uvedeny, jsou jednotlivými procesy ignorovány (kromě SIGKILL a SIGSTOP, ty se ignorovat nedají.)

[18]Funkce volaná asynchronně při přijetí signálu.

Signál	Hlavní proces
SIGTERM	Ukončí okamžitě všechny hovorové procesy a databázový proces (pošle jim SIGTERM) a ukončí svojí zachytávající smyčku (`pcap_loop()`).
SIGINT	Totéž jako SIGTERM. Použito pro klávesovou zkratku Ctrl+C zadanou uživatelem, když program neběží na pozadí (zkratka vyvolává právě SIGINT).
SIGQUIT	Nepřijímá nové hovory (ignoruje nově příchozí INVITE zprávy) a čeká na dokončení právě běžících hovorových procesů. Jakmile všechny skončí, zašle proces sám sobě SIGTERM.
SIGHUP	Znovu načte konfigurační soubor.
SIGUSR1	Vypíše informace o právě nahrávaných hovorech.

Tabulka 7.2: Reakce na signály – hlavní proces

Signál	Hovorový proces
SIGTERM	Ukončí svou zachytávající smyčku (`pcap_loop()`), uzavře soubor, do kterého zapisoval a vrátí návratovou hodnotu hlavnímu procesu.
SIGALRM	Totéž jako SIGTERM. SIGALRM je nastaven na počet sekund (direktiva `len` z konfiguračního souboru). Slouží pro případ, že by hlavní proces nezachytil BYE zprávu a tudíž neukončil hovorový proces.

Tabulka 7.3: Reakce na signály – hovorový proces

Signál	Databázový proces
SIGTERM	Ukončí svou čtecí smyčku, kterou přijímá požadavky od hlavního procesu pomocí roury. Dokončí však již započatý insert nebo update v databázi.

Tabulka 7.4: Reakce na signály – databázový proces

7.8 Logování

Logování je důležitou součástí programu. Jelikož se jedná o daemon, je potřeba ukládat informace o jeho průběhu. Program umožňuje více druhů logování: na standardní výstup, standardní chybový výstup, do souboru a pomocí ná-

stroje Syslog[19]. Program se inspiruje nástrojem Syslog a používá jeho logovací úrovně.

Úrovně jsou seřazeny vzestupně podle výřečnosti. Každá následující úroveň obsahuje tu předchozí (`LOG_DEBUG` je nejpodrobnější).

- `LOG_ERR` (chyby) – Takto jsou označovány případy, kdy se nepodařilo nahrát hovor nebo se nepodařilo hovor uložit do databáze. Například se nepovedlo vytvořit hovorový proces, hovorový proces nemá právo k zápisu do souboru, došlo místo na disku, databázový proces přijal nevalidní strukturu `TCALL` apod.

- `LOG_WARNING` (varování) – Indikuje nežádoucí stavy, např. zapisování do souboru s jiným než očekávaným jménem (pokud již soubor existuje), nemožnost nahrát hovor, protože globální buffer hovorů je plný (byla nastavena nenulová direktiva `max` v konfiguraci) a jiné. Také označuje potenciálně chybné stavy jako např. chybné CRC, nevalidní SDP parametry apod.

- `LOG_INFO` (oznámení) – Poskytuje informativní oznámení, např. že se začal nahrávat hovor, že se povedl insert do databáze apod.

- `LOG_DEBUG` (ladění) – Poskytuje různé detaily, např. realokace globálního bufferu pro hovory, shoda hovoru s pravidlem, podrobnosti o zachycení některých SIP zpráv apod.

O logování se v programu stará funkce `xlog`. Přesněji se jedná o globální ukazatel na funkci, která vypadá takto:

```
void (*xlog)(int level, const char * fmt, ...)
```

Parametr `level` udává právě zmiňovanou úroveň logování a parametrem `fmt` se předává samotná zpráva s libovolným počtem argumentů (jako u funkce `printf(3)`). Při startu programu se podle konfigurace zvolí, kam se bude logovat. Program rozlišuje dva případy. Když se zvolí logování do souboru, na standardní výstup nebo na standardní chybový výstup, je ukazateli xlog přiřazena funkce filelog, která je součástí programu. Pokud se zvolí logování do Syslogu, je ukazateli xlog přiřazena funkce syslog, která je definovaná v syslog.h (C poskytuje API pro Syslog). Protože syslog, filelog a xlog mají stejnou deklaraci, může být funkce xlog používaná takto univerzálně.

[19] Sofistikovaný logovací program, který umožňuje přijímat zprávy z více zdrojů (i po síti).

Kapitola 8

Testování

V průběhu implementování jsem testoval funkčnost programu pomocí svého IP telefonu (Cisco 303 IP Phone). Přestože je to Cisco telefon, podporuje protokol SIP. Volání jsem prováděl mezi IP telefonem a různými SIP klienty na svém notebooku (se systémem Debian), kde program Sniffer běžel. Od společnosti xPhoNet[20] jsem měl pro testování pronajatá tři čísla. Měl jsem také k dispozici přepínač se zrcadlením portů (port mirroring). Pomocí něho jsem mohl na svém notebooku odposlouchávat hovory z IP telefonu (například volání mezi IP telefonem a mým mobilním telefonem). Toto jednoduché testování pomohlo při vývoji aplikace. Poté přišly na řadu důkladnější testy.

8.1 Zátěžový test

Bohužel jsem neměl k dispozici infrastrukturu s reálným VoIP provozem, kde bych mohl aplikaci otestovat. Ing. Fišer ze společnosti xPhoNet mi však poskytl k testování své programy Hydra a Halali. Halali funguje jako simulátor SIP ústředny a Hydra umí generovat hovory (až 30 hovorů současně). Program Sniffer tedy naslouchal těmto hovorům pomocí zmiňovaného přepínače. Díky tomuto testování byla objevena a odstraněna řada drobných chyb (například pád programu při parsování OK odpovědi bez přiloženého SDP). Tento test jsem prováděl několikrát, přičemž poslední test běžel cca 2 hodiny a program zachytil cca 550 hovorů (celkem cca 900 volání).

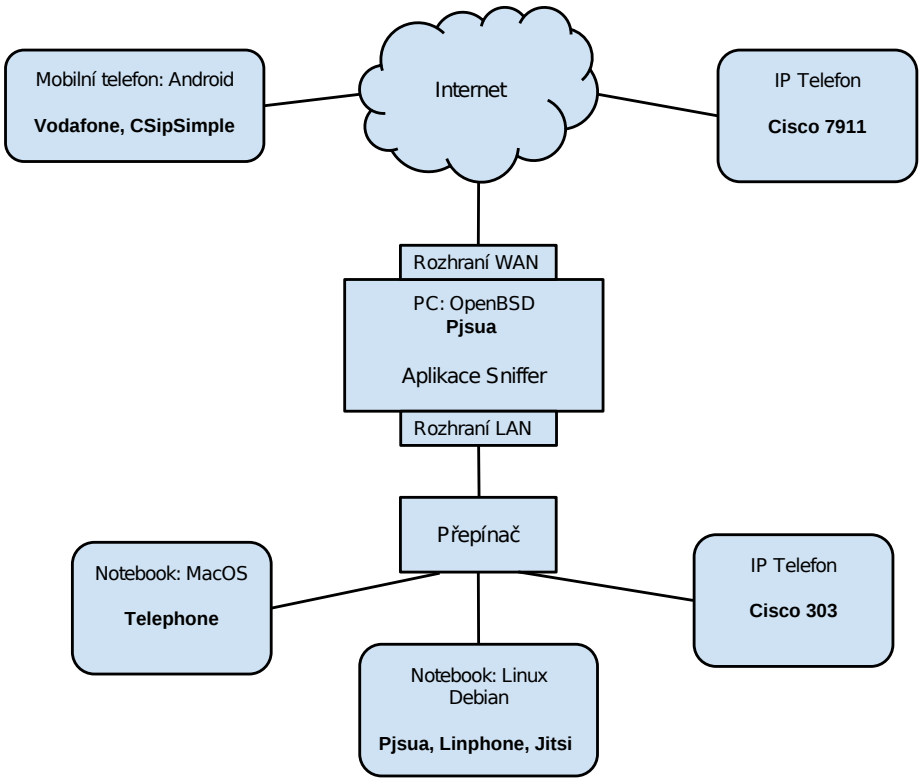
Podobně jsem také testoval Offline mód aplikace, kdy jsem si programem Wireshark odchytil takto generované hovory do souboru. Soubor měl ve finále 1.7 GB a obsahoval opět cca 550 hovorů. Sniffer tento soubor zpracovával zhruba 10 minut. Vytvářel totiž mnoho procesů (jeden na každý hovor, jako u živého odposlechu). Nicméně program úspěšně zachytil všechny hovory a uložil informace do databáze. Procesy pro Offline mód nejsou příliš efektivní, zde by se hodilo vláknové zpracování. Na druhou stranu v implementaci Offline

[20] Poskytovatel VoIP služeb: www.xphonet.cz.

módu jsou minimální rozdíly oproti implementaci živého odposlechu (právě díky použití procesů v obou případech).

8.2 Test kompatibility

Jeden celý den testování byl věnován testu kompatibility mezi různými SIP klienty. Testování proběhlo na fakultě FIT, kde jsme měli k dispozici různá zařízení. Na obrázku níže lze vidět schéma zapojení. Všechny tučné názvy jsou SIP klienti, pouze Vodafone je klasické mobilní číslo, které jsme také testovali.



Obrázek 8.1: Schéma testování kompatibility

Na PC tedy běžel program Sniffer a podle konfigurace naslouchal buď na rozhraní LAN, nebo WAN. PC také fungoval jako brána do Internetu (DHCP Server + NAT). Aplikace Sniffer tedy slyšela všechen provoz, který prošel ven nebo dovnitř LAN. Hovory mezi zařízeními v LAN byly také zachyceny, protože SIP klienti měli nastavený proxy server (v Internetu). Sniffer samozřejmě nemohl zachytit hovory mezi mobilním telefonem a IP telefonem Cisco 7911.

Tabulka 8.1 ukazuje všechny kombinace vyzkoušených volání. V levém sloupci jsou volající, v řádku volaní.

	Cisco 7911	Cisco 303	Tele phone	Pjsua (Deb)	Jitsi	Lin phone	Pjsua (BSD)	CSIP	Voda fone
Cisco 7911	●	✓	✓	✓	✓	✓	✓	OUT	OUT
Cisco 303	✓	●	✓	✓	✓	✓	✓	✓	✓
Tele phone	✓	✓	●	✓	✓	✓	✓	✓	✓
Pjsua (Deb)	✓	✓	✓	●	●	●	✓	✓	✓
Jitsi	✓	ACK Fail	✓	●	●	●	✓	✓	ACK Fail
Lin phone	✓	✓	✓	●	●	●	✓	✓	✓
Pjsua (BSD)	✓	code 603	✓	code 603	code 603	code 603	●	✓	✓
CSIP	OUT	✓	✓	✓	✓	✓	✓	●	●
Voda fone	OUT	✓	✓	✓	✓	✓	✓	●	●

Tabulka 8.1: Testování SIP klientů

Popis hodnot v tabulce:

- ✓ – Hovor se uskutečnil a úspěšně se nahrál (včetně správného zápisu do databáze)
- ● – Volání nemělo smysl testovat, protože se jedná o totožného klienta, resp. totožné zařízení
- OUT – Volání nemělo smysl testovat, protože ho aplikace Sniffer nemohla zachytit (proběhlo ve vnější síti)
- ACK Fail – Klient Jitsi občas nedokázal vytvořit ACK odpověď (chyba klienta) a volání neproběhlo
- code 603 – Některá z ústředen (xPhoNet nebo IPVOX) z neznámých důvodů odmítla pokus o volání kódem 603 (Declined)

Testování tedy proběhlo úspěšně, bylo korektně nahráno a zapsáno do databáze cca 120 volání. Všechny uvedené chyby nebyly způsobeny aplikací Sniffer. Testování však ukazuje, že ve VoIP a SIP světě lze narazit na různé problémy s kompatibilitou. Například případ, kdy ústředna zaslala chybný kód 603 (Declined), neodpovídal popisu kódu v `RFC 3261`.

8.2.1 Problémy s NAT

V případech, kdy Sniffer poslouchal na rozhraní WAN, docházelo k problému s NAT (Network Address Translation). Ten totiž na tomto systému (OpenBSD) prováděl randomizaci portů[21]. Sniffer tedy naslouchal hovoru na jiném portu, než bylo domluveno v protokolu SDP. Problém se vyřešil tím, že Sniffer dále naslouchal na rozhraní LAN. Funkcionalita byla zachovaná, protože síťový provoz vždy protekl oběma rozhraními. Ideálně by měl být Sniffer provozován na lokální síti tam, kde ještě neproběhl překlad adres pomocí NAT.

VoIP telefonie obecně naráží na problémy s NAT. Jelikož je VoIP peer-to-peer služba, může nastat problém v situacích, kdy se volající pokouší dovolat někomu, kdo má své zařízení v privátní síti. U klient-server služeb (webový prohlížeč) s tímto není problém, protože klient z privátní sítě iniciuje spojení, takže v NAT „otevře cestu“ k serveru. Pokud například v NAT neexistuje záznam, kam směrovat SIP port, příchozí INVITE zpráva z veřejné sítě bude zahozena. Jedním řešením je přesměrování portů (port forwarding), což může být složité, pokud je v privátní síti více VoIP klientů. Operátor xPhoNet používá metodu častého registrování (REGISTER), a sice každou minutu. To je většinou dostatečná doba, po kterou se v NAT tabulce udrží záznam o cestě.

8.2.2 Co se nepodařilo otestovat

Kvůli nedostupnosti prostředků jsme nemohli otestovat některé funkce programu. Například SIP signalizaci s použitím IPv6 či TCP nebo nasazení na VLAN síti. Na komplexnější test nahrávání podle pravidel taky z časových důvodů nedošlo, základní testy s malým počtem pravidel jsem však provedl.

[21] Zabezpečovací technika, kdy se při odchozím požadavku z LAN zvolí náhodný zdrojový port.

Kapitola 9

Zjištěné problémy a výhled do budoucna

Všechny cíle práce byly splněny, některé další funkce zůstaly pouze rozpracovány kvůli časové náročnosti zadaných cílů. Například zmiňovaná databázová funkce `update_crashed` zůstala nevyužitá. Je určena k nahlašování systémových chyb databázi, například pokud byl hovorový proces předčasně ukončen.

Aplikace by v budoucnu mohla podporovat i signalizaci pomocí protokolu SCCP nebo odposlouchávání konferenčních hovorů. Zasloužila by mnoho vylepšení, například více vyladit Offline mód pro efektivnější zpracování dump souborů, v pravidlech rozlišovat i rozsahy telefonních čísel, umožnit aktualizaci pravidel za běhu programu, důkladněji logovat signalizační provoz (např. celé SIP zprávy), více konfiguračních direktiv pro flexibilnější použití, také by potřebovala otestovat v reálném provozu apod.

V průběhu implementace jsem také zjišťoval, že volba procesů pro paralelní zpracování možná nebyla zcela vhodná. Původní idea byla použít procesy kvůli bezpečnosti (privilege separation). Narazil jsem ale na problém s knihovnou Pcap, kvůli kterému se hlavní proces nemohl zcela vzdát práv superuživatele. Hlavní proces ukončuje potomky pomocí funkce `kill`. Pokud by nastala v programu chyba, hlavní proces by takto mohl omylem ukončit jakýkoli jiný proces, protože má stále RUID superuživatele. Na druhou stranu byla u procesů snazší synchronizace a dedikovaný databázový proces se ukázal jako efektivní řešení.

Závěr

Výsledkem práce je funkční prototyp aplikace, která rozpoznává, ukládá a monitoruje VoIP hovory signalizované protokolem SIP. Rozpoznává nahrávací pravidla, podle kterých filtruje hovory. Cíle práce byly tedy splněny. Navíc ukládá metainformace do databáze a díky tomu spolupracuje s ostatními programy z komplexního řešení, což je nad rámec zadání této bakalářské práce.

Aplikace byla v rámci možností dostatečně otestována. Implementace monitorování protokolu SIP byla nečekaně časově náročná, tudíž se nepodařilo splnit volitelný aspekt práce, signalizaci protokolem SCCP. To by ovšem bylo složité, protože SCCP je proprietární protokol.

Jedná se o typ aplikace, který je třeba neustále rozvíjet a vylepšovat. Ve VoIP lze vždy narazit na odlišné chování klientů a ústředen, které ne vždy ctí pravidla popsána v dokumentech RFC. Přestože je SIP dobře strukturovaný, textový protokol, jeho implementace si často RFC vykládají po svém. Tato aplikace jakožto pasivní monitorovací nástroj musí těmto implementacím rozumět, což není vždy jednoduché.

Literatura

[1] Kučera, J.: *Extrakce a zpracování hlasu z VOIP paketů*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

[2] Robejšek, V.: *Webové rozhraní k VOIP hovorům*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

[3] Hallock, J.: A Brief History of VoIP. [online], 2004, [cit. 2015-05-05]. Dostupné z: `http://www.joehallock.com/edu/pdfs/Hallock_J_VoIP_Past.pdf`

[4] CESNET, z.s.p.o.: IP telefonie. [online], [cit. 2015-05-05]. Dostupné z: `http://www.cesnet.cz/sluzby/ip-telefonie/`

[5] Cisco Systems, Inc.: Quality of Service for Voice over IP. [online], [cit. 2015-05-05]. Dostupné z: `http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/qos_solutions/QoSVoIP/QoSVoIP.html`

[6] Federal Communications Commission: Voice Over Internet Protocol (VoIP). [online], [cit. 2015-05-05]. Dostupné z: `https://www.fcc.gov/encyclopedia/voice-over-internet-protocol-voip`

[7] Rosenberg, J.; Schulzrinne, H.; Camarillo, G.; aj.: RFC 3261: SIP: Session Initiation Protocol. Technická zpráva, IETF, 2002, [cit. 2015-05-05]. Dostupné z: `www.ietf.org/rfc/rfc3261.txt`

[8] Schulzrinne, H.; Casner, S.; Frederick, R.; aj.: RFC 3550: RTP: A Transport Protocol for Real-Time Applications. Technická zpráva, IETF, 2003, [cit. 2015-05-05]. Dostupné z: `www.ietf.org/rfc/rfc3550.txt`

[9] Johnston, A.: *SIP: Understanding the Session Initiation Protocol*. Artech House telecommunications library, Artech House, 2009, ISBN 978-1-60783-995-8.

[10] Wikipedia: Session Initiation Protocol. [cit. 2015-05-05]. Dostupné z: http://cs.wikipedia.org/wiki/Soubor:SIP_transakce.png

[11] Jacobson, V.; Handley, M.: RFC 4566: SDP: Session Description Protocol. Technická zpráva, IETF, 2006, [cit. 2015-05-05]. Dostupné z: www.ietf.org/html/rfc4566

[12] Rosenberg, J.; Schulzrinne, H.: RFC 3264: An Offer/Answer Model with the Session Description Protocol (SDP). Technická zpráva, IETF, 2002, [cit. 2015-05-05]. Dostupné z: www.ietf.org/rfc/rfc3264.txt

[13] Prokop, A.: Understanding Session Description Protocol (SDP). [online], 2013, [cit. 2015-05-05]. Dostupné z: https://andrewjprokop.wordpress.com/2013/09/30/understanding-session-description-protocol-sdp/

[14] Wikipedia: RTP audio video profile. [online], 2015, [cit. 2015-05-05]. Dostupné z: http://en.wikipedia.org/wiki/RTP_audio_video_profile

[15] Garcia, L. M.: Programming with Libpcap - Sniffing the Network From Our Own Application. *Hakin9*, 2008.

[16] Tcpdump – A powerful command-line packet analyzer. [cit. 2015-05-05]. Dostupné z: http://www.tcpdump.org/

[17] McCanne, S.; Jacobsonn, V.: The BSD Packet Filter: A New Architecture for User-level Packet Capture. Berkeley, CA, USA: USENIX Association, 1993, [cit. 2015-05-05]. Dostupné z: https://www.usenix.org/legacy/publications/library/proceedings/sd93/mccanne.pdf

[18] PF_RING^{TM} - High-speed packet capture, filtering and analysis. [cit. 2015-05-05]. Dostupné z: http://www.ntop.org/products/packet-capture/pf_ring/

[19] Wireshark Foundation: WLAN (IEEE 802.11) capture setup. [cit. 2015-05-05]. Dostupné z: https://wiki.wireshark.org/CaptureSetup/WLAN#Promiscuous_mode

Příloha A

Seznam použitých zkratek

API Application Programming Interface

AVP Audio Video Profile

BPF Berkeley Packet Filter

BSD Berkeley Software Distribution

CPU Central Processing Unit

CRC Cyclic Redundancy Check

EUID Effective User ID

GSM Global System for Mobile Communications

GUI Graphical user interface

HTTP Hypertext Transfer Protocol

IP Internet Protocol

LAN Local Area Network

LTE Long Term Evolution

MTU Maximum Transmission Unit

NAT Network Address Translation

NIC Network Interface Controller

PCAP Packet Capture

QoS Quality of Service

RFC Request for Comments

RTP Real-time Transfer Protocol

RUID Real User ID

SCCP Skinny Call Control Protocol

SDP Session Description Protocol

SIP Session Initiation Protocol

SQL Structured Query Language

TCP Transmission Control Protocol

UA User Agent

UAC User Agent Client

UAS User Agent Server

UDP User Datagram Protocol

UID User ID

URI Uniform Resource Identifier

VoIP Voice over Internet Protocol

VoLTE Voice over Long Term Evolution

VoWLAN Voice over Wireless Local Area Network

WAN Wide Area Network

Wi-Fi Wireless Fidelity

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `doc` ..... adresář s programátorskou dokumentací k aplikaci
  - `index.html` ..... dokumentace ve formátu HTML
- `src`
  - `impl` ..... zdrojové kódy implementace, instalační příručka, manuálová stránka
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF