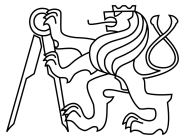
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Signalizace a nahrávání VOIP hovorů
Student:	Matěj Polák
Vedoucí:	Mgr. Jan Starý, Ph.D.
Studijní program:	Informatika
Studijní obor:	Informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

1. Rozšiřte existující aplikaci VOIPCall z předchozích BP (Šuster, Kučera, Robejšek) o podporu Cisco technologií. Stávající aplikace podporuje SIP protocol - rozšiřte aplikaci o možnost komunikace proprietárním Cisco Skinny protokolem. Prozkoumejte možnosti šifrované komunikace s Cisco Call Managerem a podle zjištěných možností implementujte i zachytávání šifrované signalizace a šifrovaných hovorů.

2. Implementaci proveďte na UNIX plaformě v jazyce C. Využijte existujících knihoven a nástrojů (libpcap, tcpdump, ...). Dbejte na korektnost, čistotu, přenositelnost a rozšiřitelnost kódu.

3. Vytvořenou funkcionalitu zdokumentujte ve standardní manuálové stránce.

4. Vytvořenou aplikaci řádně otestujte, metody testování zdokumentujte.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 1. prosince 2015

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

Bakalářská práce

Signalizace a nahrávání VOIP hovorů

Matěj Polák

Vedoucí práce: Mgr. Jan Starý, Ph.D.

16. května 2016

Poděkování

Děkuji Mgr. Janovi Starému, Ph.D. za rady a vedení práce. Dále děkuji rodině a také všem, co mě podporovali.

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 16. května 2016 .....

Odkaz na tuto práci

Polák, Matěj. *Signalizace a nahrávání VOIP hovorů*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

VoIP provoz na síti lze monitorovat, signalizované VoIP hovory zachytávat, a ze zachycených packetů sestavovat audio soubory. V předchozích pracích vznikla síťová aplikace, která nahrává VoIP hovory signalizované protokolem SIP. Cílem této práce je rozšířit stávající aplikaci o podporu proprietárního signalizačního protokolu Skinny Client Control Protocol firmy Cisco.

Klíčová slova VoIP telefonie, telefonní hovory, signalizace VoIP hovorů, Skinny Client Control Protocol, Skinny

Abstract

VoIP traffic in the network can be monitored, signaled VoIP calls can be captured, and audio streams can be extracted from the the captured packets. Previous works have implemented a network application recording VoIP calls signaled with the SIP protocol. Our purpose is to extend this application to support the proprietary Skinny Client Control Protocol used by Cisco.

Keywords VoIP telephony, phone calls, VoIP call signaling, Skinny Client Control Protocol, Skinny

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Kromě běžně používaných mobilních telefonů a sítí GSM se dnes používají čím dál šířeji i telefony, které k přenosu hlasu používají běžnou počítačovou síť a Internet. Tato technologie se nazývá souhrnně VoIP telefonie (Voice over IP). Na úrovni aplikačních protokolů se většinou jedná o nějaký signalizační protokol, pomocí kterého se jednotliví účastníci dorozumívají o použitém audio kodeku, číslech portů a podobně, a nějaký protokol, který doprovází přenos samotných zvukových dat.

Nejznámější aplikací pro internetové telefonování je bezpochyby Skype. Ten používá vlastní šifrovaný protokol, takže je velice obtížné (a v praxi nemožné) bez znalosti šifrovacích algoritmů a klíčů a použitého protokolu hovory přes Skype zachytávat a dále zpracovávat.

Kromě Skypu se však používají i otevřenější technologie a protokoly. Dva nejpoužívanější signalizační protokoly jsou SIP a Skinny. Protokolu SIP se věnuje práce F. Šustera [1], která spolu s pracemi J. Kučery [2] a V. Robejška [3] implementuje původní aplikaci na nahrávání VoIP hovorů. Moje práce implementuje druhý zmíněný protokol — Skinny Client Control Protocol, zvaný též zkráceně Skinny.

Kapitola 1

Cíl práce

Cílem práce je rozšířit stávající projekt o podporu protokolu Skinny tak, aby aplikace dokázala odchytávat hovory uskutečněné v místní síti (LAN), informace o těchto hovorech ukládat do databáze (tak jako dosud SIP hovory), a zároveň zachytávat i zvuková data v podobě, ze které lze extrahovat samotný audio obsah. Zároveň je kladen důraz na přenositelnost výsledného zdrojového kódu. Aplikace je napsána v jazyce C s využitím standardních knihoven systému UNIX. Vzhledem k tomu, že se jedná o rozšíření předchozí implementace, je tato volba nejjednodušší variantou. Součástí práce je programátorská a uživatelská dokumentace. Z komentářů ve zdrojovém kódu lze pomocí programu Doxygen vytvořit programátorskou dokumentaci ve formátu HTML. Uživatelskou dokumentaci tvoří manuálová stránka. Nedílnou součástí práce je řádné otestování implementované funkcionality.

Kapitola 2

Nástroje

Při práci jsem používal různé další nástroje, které popisuje tato kapitola.

2.1 Wireshark

Wireshark je program sloužící k analýze zachycených síťových dat, které dokáže přehledně zobrazit. Obsahuje tzv. *dissectory* pro jednotlivé protokoly. Tyto dissectory slouží k „rozebrání“ binárních dat jednotlivých packetů a zobrazení jejich hodnot v přehledné stromové struktuře. Kromě toho umožňují vyhledávání hodnot různých položek v zachycených packetech. Jeden z těchto dissectorů je implementován i pro protokol Skinny a umožňuje detailní zobrazení jednotlivých položek ve Skinny zprávách. Zdrojový kód tohoto dissectoru je k dispozici například na [4]. Jedná se o jeden z hlavních zdrojů informací o struktuře protokolu Skinny, který jsem byl schopen najít.

2.2 Pcap a Berkeley Packet Filter

Pcap je standardní knihovna pro práci s packety zachycené ze síťového provozu. Na Pcap jsou postaveny i další nástroje použité v této práci, jako třeba tcpdump nebo Wireshark. Tato knihovna zachycuje jednotlivé rámce (*frames*), které je potřeba rozebrat po jednotlivých vrstvách ISO/OSI modelu a extrahovat z nich pro aplikaci důležité údaje. Zároveň knihovna umožňuje packety filtrovat podle síťového protokolu, čísel portů apod. K tomu používá implementaci zvanou Berkeley Packet Filter, která umožňuje filtrování packetů v uživatelském prostoru operačního systému (*user space*) s ohledem na vysoký výkon a mohou nejmenší režii při zpracovávání packetů.

Kapitola 3

VoIP obecně

VoIP (Voice over Internet Protocol) je obecné označení pro jakoukoliv technologii umožňující realizaci telefonních hovorů přes počítačovou síť pomocí protokolu IP. Počítačovou sítí se myslí jak místní síť (LAN) tak Internet. VoIP telefonní hovory nemusí být pouze hlasové (zvukové), mohou obsahovat i video. Pro účely této práce se ale budu zabývat jen hovory obsahujícími pouze zvuková data.

Síť podporující protokol IP přepojuje packety (*packet switching*), nikoliv okruhy (*circuit switching*), kdy je pro každý hovor vyhrazena cesta sítí mezi volajícím a volaným po celou dobu hovoru. Zároveň musí být síť pro VoIP telefonii dostatečně rychlá, tj. schopná přenést v daném čase určité minimální množství dat nutné pro plynulý přenos zvuku. To v dnešní době rychlého připojení přestává být problém, minimální nutnou rychlost pro VoIP hovory splňuje i relativně nízkorychlostní připojení.

VoIP se používá většinou ve stejné síti, ve které se používají i jiné síťové technologie. Díky tomu jsou v této síti přítomny i datové packety nijak nesouvisející s telefonními hovory. Na rozdíl od jiných datových přenosů, u kterých většinou záleží na kvalitě přenosu i na úkor rychlosti, u VoIP je tomu přesně naopak. Pokud například trvá načtení webové stránky nebo stažení nějakého souboru o sekundu nebo několik sekund déle, než obvykle, uživateli to většinou nevadí a počká. Případné chyby přenosu jsou tedy raději opravovány a chybné packety posílány znovu. Příkladem takovéto přenosu je protokol TCP. Pro VoIP je tomu ale přesně naopak — pokud by se nějaký packet, který byl součástí zvukových dat, zpozdil o sekundu a čekalo by se na jeho odeslání znovu, pro uživatele by VoIP bylo nepoužitelné. Proto se pro přenos těchto packetů používá nepotvrzovaný protokol UDP, u kterého nejsou jednotlivé packety kontrolovány a v případě chyby jsou raději zahozeny, protože je mnohem důležitější, aby byla tyto data doručena včas i za cenu možných chyb přenosu, než doručena se zpožděním.

S tím také souvisí QoS (Quality of Service) — pomocí QoS je možné vyhradit pro VoIP packety nějakou šířku datového toku sítí. Tím lze docílit toho,

že i kdyby by byla síť zahlcena jinými datovými přenosy, VoIP hovory by tím nebyly ovlivněny a bylo by možné tuto síť stále používat pro telefonování.

Pro VoIP se obecně používají dva různé protokoly. První je signalizační, kterým si zařízení mezi sebou vyměňují informace o tom, kdo komu volá, jestli druhý telefon vyzvání nebo jestli druhý telefon hovor přijal nebo odmítl. Obecně se pro jakékoli události, které nějak souvisí s VoIP a nejedná se přímo o zvuková data, používá právě nějaký ze signalizačních protokolů (více o těchto protokolech je uvedeno v dalších kapitolách této práce).

Druhým protokolem je transportní. Tento protokol se, na rozdíl od signalizačních, používá pouze jeden. Jedná se o univerzální protokol RTP (Real time Transport Protocol), ve kterém jsou obsažená zvuková data zkomprimována nějakým kodekem.

Kromě signalizačního a transportního protokolu využívá VoIP obvyklé služby DNS, DHCP, TFTP, DNS, a NTP, které nejsou pro VoIP specifické.

Typické pro VoIP je využití technologie PoE (Power over Ethernet). Jedná se o napájení VoIP telefonů přímo pomocí ethernetových kabelů, tedy stejných kabelů, kterými se tato zařízení připojují do počítačové sítě. To podstatně usnadňuje zapojování nových telefonů — stačí připojit pouze jeden kabel [5].

3.1 Právní pozadí odposlechu hovorů

Obecně platí, že nelze odposlouchávat a nahrávat hovory bez souhlasu nahrávaných osob. Konkrétně toto právo upravuje Zákon o ochraně osobních údajů (více v kapitole 2.2 v [1]). Těmito právními aspekty se nebudu zabývat. Pro testování programu byly použity bezobsažné hovory na VoIP telefonech a mobilních telefonech autora a školitele.[1]

3.2 Signalizační protokoly

Ve VoIP je potřeba nějakým způsobem informovat všechny zúčastněné strany hovoru o jednotlivých událostech. Například o vytočení čísla, zvonění druhého telefonu, zvednutí nebo zavěšení sluchátka nebo odmítnutí hovoru. Před začátkem přenosu zvukových dat je potřeba si vyměnit informaci o použitém typu komprimace zvukových dat. Právě k tomuto účelu slouží signalizační protokoly. Nejpoužívanější jsou tyto tři: SIP, Skinny Client Control Protocol (SCCP nebo také jenom Skinny) a H.323.

3.2.1 SIP

SIP (Session Initation Protocol) je běžně používaný protokol pro signalizaci VoIP hovorů. Jedná se o textový protokol, který svou strukturou připomíná SMTP nebo HTTP, se kterými sdílí systém požadavků (*request*) a odpovědí

[1] Například: „Začal přenos dat — vidíme packety?“ „Ano, zavěšuji.“

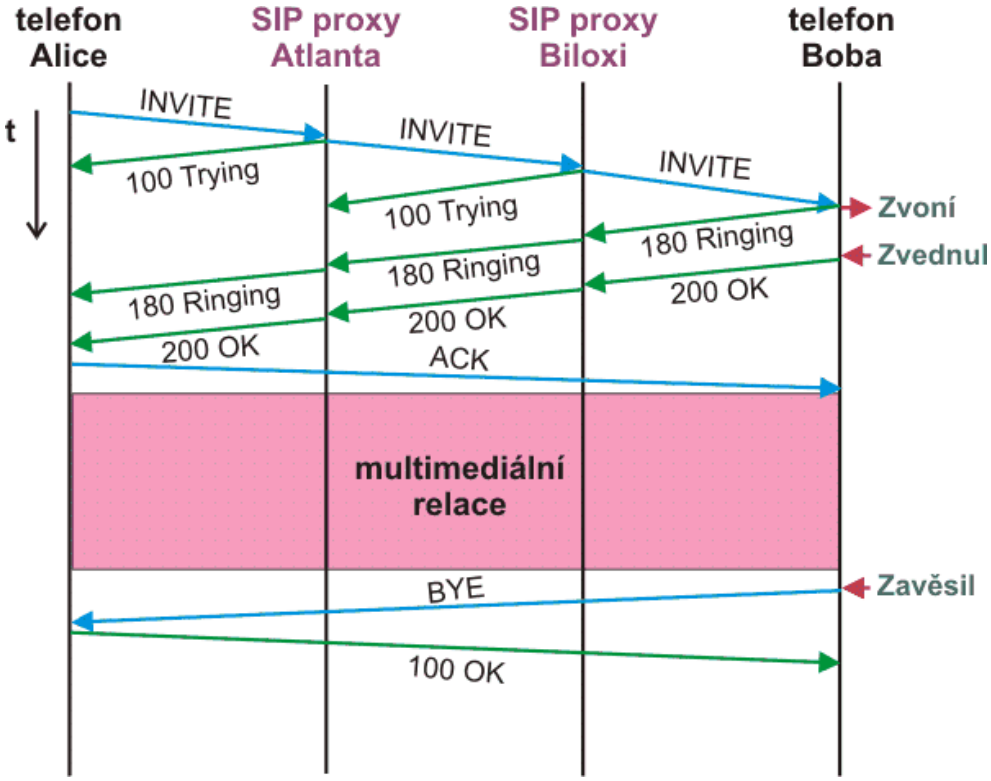
```
▽ Session Initiation Protocol (INVITE)
  ▽ Request-Line: INVITE sip:324143242@192.168.222.3:5060 SIP/2.0
      Method: INVITE
    ▷ Request-URI: sip:324143242@192.168.222.3:5060
      [Resent Packet: False]
  ▽ Message Header
    ▷ Via: SIP/2.0/UDP 82.117.136.131:5060;branch=z9hG4bK274198f8;rport
    ▷ From: "775681626" <sip:775681626@82.117.136.131>;tag=as52d4fe42
    ▷ To: <sip:324143242@192.168.222.3:5060>
    ▷ Contact: <sip:775681626@82.117.136.131>
      Call-ID: 106a0c817a81175801888f692bda2a0c@82.117.136.131
    ▷ CSeq: 102 INVITE
      User-Agent: Asterisk PBX
      Max-Forwards: 70
      Date: Sat, 25 Oct 2014 14:20:56 GMT
      Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
      Content-Type: application/sdp
      Content-Length: 340
  ▷ Message Body
```

```
0000  00 0e 08 d7 37 1e 00 0d  b9 12 9f 2e 08 00 45 10   ....7... ......E.
0010  03 8d 21 aa 00 00 33 11  00 00 52 75 88 83 c0 a8   ..!...3. ..Ru....
0020  de 03 13 c4 13 c4 03 79  3a 25 49 4e 56 49 54 45   .......y :%INVITE
0030  20 73 69 70 3a 33 32 34  31 34 33 32 34 32 40 31    sip:324 143242@1
0040  39 32 2e 31 36 38 2e 32  32 32 2e 33 3a 35 30 36   92.168.2 22.3:506
0050  30 20 53 49 50 2f 32 2e  30 0d 0a 56 69 61 3a 20   0 SIP/2. 0..Via:
0060  53 49 50 2f 32 2e 30 2f  55 44 50 20 38 32 2e 31   SIP/2.0/ UDP 82.1
0070  31 37 2e 31 33 36 2e 31  33 31 3a 35 30 36 30 3b   17.136.1 31:5060;
0080  62 72 61 6e 63 68 3d 7a  39 68 47 34 62 4b 32 37   branch=z 9hG4bK27
0090  34 31 39 38 66 38 3b 72  70 6f 72 74 0d 0a 46 72   4198f8;r port..Fr
00a0  6f 6d 3a 20 22 37 37 35  36 38 31 36 32 36 22 20   om: "775 681626"
00b0  3c 73 69 70 3a 37 37 35  36 38 31 36 32 36 40 38   <sip:775 681626@8
00c0  32 2e 31 31 37 2e 31 33  36 2e 31 33 31 3e 3b 74   2.117.13 6.131>;t
00d0  61 67 3d 61 73 35 32 64  34 66 65 34 32 0d 0a 54   ag=as52d 4fe42..T
00e0  6f 3a 20 3c 73 69 70 3a  33 32 34 31 34 33 32 34   o: <sip: 32414324
00f0  32 40 31 39 32 2e 31 36  38 2e 32 32 32 2e 33 3a   2@192.16 8.222.3:
0100  35 30 36 30 3e 0d 0a 43  6f 6e 74 61 63 74 3a 20   5060>..C ontact:
0110  3c 73 69 70 3a 37 37 35  36 38 31 36 32 36 40 38   <sip:775 681626@8
0120  32 2e 31 31 37 2e 31 33  36 2e 31 33 31 3e 0d 0a   2.117.13 6.131>..
0130  43 61 6c 6c 2d 49 44 3a  20 31 30 36 61 30 63 38   Call-ID:  106a0c8
0140  31 37 61 38 31 31 37 35  38 30 31 38 38 38 66 36   17a81175 801888f6
0150  39 32 62 64 61 32 61 30  63 40 38 32 2e 31 31 37   92bda2a0 c@82.117
0160  2e 31 33 36 2e 31 33 31  0d 0a 43 53 65 71 3a 20   .136.131 ..CSeq:
0170  31 30 32 20 49 4e 56 49  54 45 0d 0a 55 73 65 72   102 INVI TE..User
0180  2d 41 67 65 6e 74 3a 20  41 73 74 65 72 69 73 6b   -Agent:  Asterisk
0190  20 50 42 58 0d 0a 4d 61  78 2d 46 6f 72 77 61 72    PBX..Ma x-Forwar
01a0  64 73 3a 20 37 30 0d 0a  44 61 74 65 3a 20 53 61   ds: 70.. Date: Sa
01b0  74 2c 20 32 35 20 4f 63  74 20 32 30 31 34 20 31   t, 25 Oc t 2014 1
01c0  34 3a 32 30 3a 35 36 20  47 4d 54 0d 0a 41 6c 6c   4:20:56  GMT..All
01d0  6f 77 3a 20 49 4e 56 49  54 45 2c 20 41 43 4b 2c   ow: INVI TE, ACK,
01e0  20 43 41 4e 43 45 4c 2c  20 4f 50 54 49 4f 4e 53    CANCEL,  OPTIONS
01f0  2c 20 42 59 45 2c 20 52  45 46 45 52 2c 20 53 55   , BYE, R EFER, SU
0200  42 53 43 52 49 42 45 2c  20 4e 4f 54 49 46 59 0d   BSCRIBE,  NOTIFY.
0210  0a 43 6f 6e 74 65 6e 74  2d 54 79 70 65 3a 20 61   .Content -Type: a
```

Obrázek 3.1: Ukázka zprávy SIP protokolu z programu Wireshark

(*response*) a použití stavových kódů [6]. Na obrázku 3.1 je zachycena struktura jedné ze SIP zpráv. Díky této podobnosti není obtížné SIP packety rozpoznat a i bez speciálního softwaru pochopit, o jakou informaci v jaké SIP zprávě jde. Pro SIP signalizaci se používá jak protokol UDP tak TCP.

Není cílem této práce popisovat detaily protokolu SIP. Zaměřím se především na rozdíly mezi SIP a Skinny. Kromě hlavního rozdílu — formátu signalizačních zpráv — je největší rozdíl ve způsobu posílání těchto zpráv: SIP pro svojí funkčnost nepotřebuje žádný server (Call Manager) v místní síti, telefony s podporou SIP jsou schopny fungovat samy rovnou po připojení do sítě. Také umístění hlavní logiky komunikace celého protokolu je jinde — SIP telefony si samy volí tzv. Call ID, tedy identifikátor hovoru a již po výměně cca třech zpráv lze přejít na posílání samotných zvukových dat hovoru. Na obrázku 3.2



Obrázek 3.2: Signalizace typického hovoru při využití SIP protokolu [7]

je vidět průběh typického SIP hovoru. K navázání hovoru stačí odeslat pouze zprávu INVITE, počkat na potvrzení o zvednutí sluchátka (zpráva OK) a hovor může začít. Skinny před navázáním spojení odešle mnohem více zpráv (viz podrobně v následující kapitole).

Více informací o protokolu SIP poskytuje učebnice [5], kompletní definice protokolu je obsahem RFC 3261 [8].

3.2.2 Skinny Client Control Protocol

Skinny Client Control Protocol (SCCP), nebo zkráceně jen Skinny, je proprietární VoIP protokol používaný zařízeními společnosti Cisco, který pracuje na protokolu TCP. Pro komunikaci používá port 2000, pro šifrovanou komunikaci port 2443. Pro šifrování se používá TLS, které je dostupné od Call Manageru verze 4 [9]. Na rozdíl od protokolu SIP se jedná o binární, nikoli textový protokol. Jednotlivé položky zprávy jsou tedy ve většině případů binární čísla s daným počtem bytů a pořadím, přičemž pořadí jednotlivých bytů je *little endian* (nejméně významný byte na nejnižší adrese). Tím se tedy liší od zbytku síťové komunikace, která používá pro všechny hodnoty *big endian* (nejméně významný byte na nejvyšší adrese). O tomto pořadí bytů ve Skinny

jsem nenašel žádnou informaci, a to ani v [5], kde je protokol Skinny jinak velice podrobně popsán. Nikde v protokolu (alespoň ve verzích, které jsem měl k dispozici) není žádná proměnná, která by umožňovala zvolit pořadí bytů; předpokládám tedy, že se jedná o *little endian* vždy a že je to vlastnost protokolu, přestože není nikde zmíněná.

Pro fungování tohoto protokolu je potřeba, aby ve stejné síti, ve které jsou zapojené telefony, byl také zapojen systém nazývaný Cisco Unified Communications Manager (zmiňovaný v této práci jako Call Manager), který zajišťuje komunikaci mezi jednotlivými telefony, obsluhuje jejich události a zajišťuje spojení hovoru po vytočení čísla. Kromě toho také poskytuje operační systém a konfigurační soubory (pomocí protokolu TFTP) pro jednotlivé telefony, které si tento systém a konfigurační soubory samy při prvním zapojení stahují a následně používají pro síťovou komunikaci.

Jednotlivé zprávy jsou posílány uvnitř packetů protokolu TCP, přičemž jeden packet může obsahovat i více zpráv. Zprávy uvádí v hlavičce svou délku, takže pokud je packet delší, než uvedená délka zprávy, znamená to, že obsahuje další zprávu. Není zde ovšem žádná položka udávající počet zpráv v packetu, ani jestli je daná Skinny zpráva poslední nebo následuje ještě nějaká. Podrobněji se těmto zprávám věnuje kapitola 5.

3.2.3 H.323

Tento protokol byl původně vyvinut hlavně pro multimediální konference v místní síti (LAN). Postupně se začal používat i pro účely VoIP. V této práci ho zmiňuji jenom pro úplnost, více informací lze nalézt například v [5] nebo v [10].

Kapitola 4

Rozšíření aplikace

4.1 Původní stav

Cílem původního projektu, který je touto prací rozšířen o podporu Skinny protokolu, bylo vytvořit nástroj pro odchytávání VoIP hovorů na síti, jejich zpracování a zpětné přehrávání. Tento projekt se skládá ze tří částí:

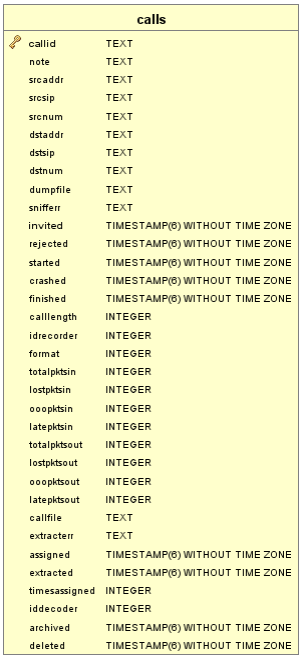
První je část pro detekci signalizačních dat hovoru v komunikaci zachycené buďto na síti, nebo v souboru. Tato komponenta je popsána v práci F. Šustera [1]. V původní verzi podporuje pouze protokol SIP, ostatní protokoly ignoruje. Pro každý nahrávaný hovor se spouští samostatný UNIXový proces.

Druhá část extrahuje zvuková data ze zachycených hovorů a vytváří zvukové soubory, který lze později přehrát. Popsána je v práci J. Kučery [2].

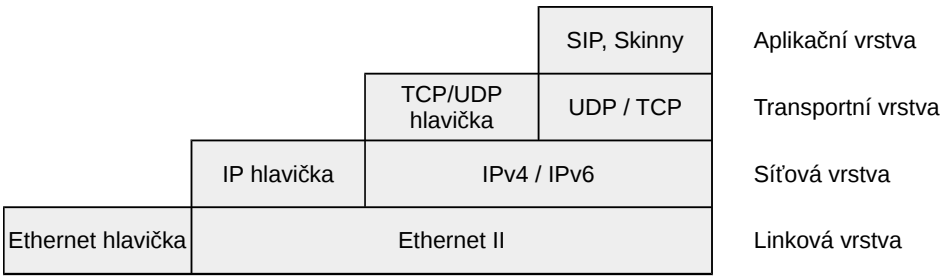
Třetí částí je webové rozhraní pro správu zachycených hovorů a návrh databáze, ve které jsou uloženy informace o hovorech. Je popsána v práci V. Robejška [3]. Pro webové rozhraní byl použit jazyk PHP a pro databázi databázový systém *PostgreSQL*. Jednotlivé komponenty přistupují k databázi výradně pomocí vložených procedur (*stored procedures*), nespouští tedy svůj vlastní SQL kód. Schéma databáze je na obrázku 3.1 v [3]. Relevantní pro moji práci je pouze tabulka `calls` (obrázek 4.1), která obsahuje zachycené hovory. Ostatní tabulky obsahují například informace o nahrávacích pravidlech nebo rekordérech a této práce se netýkají.

Celý projekt je uložen v repozitáři verzovacího systému Git. Díky tomu je možné všechny úpravy zdrojového kódu zpětně dohledat. Lze například zjistit, kdo a kdy upravil jaký řádek jakého souboru. Kromě zjištění historie umožňuje Git také pohodlnou synchronizaci změn mezi více uživateli. Více informací o tomto systému lze najít v [11].

Pro extrakci jednotlivých packetů ze síťové komunikace se používá standardní knihovna *libpcap* implementující rozhraní Pcap a pro filtrování packetů podle portů a IP adres Berkeley Packet Filter. Rozebrání packetů je znázorněno na obrázku 4.2. Zpracovány jsou pouze packety protokolu TCP nebo UDP. V případě SIP se packety dále zpracovávají jako SIP zprávy použitím



Obrázek 4.1: Sloupce v tabulce `calls` [3].



Obrázek 4.2: Rozebrání packetu podle ISO/OSI

knihovny *libosip* a extrahují se signalizační data.

Kromě živého zachytávání packetů ze sítě umožňuje libpcap také čtení ze souboru obsahujícího záznam ze síťové komunikace z minulosti. Díky tomu lze program spustit tak, aby četl packety z předem uloženého souboru místo ze sítě. Jediný a podstatný rozdíl je v rychlosti zpracování, protože při čtení ze souboru není nutné čekat na další packet, všechny jsou dostupné okamžitě.

Součástí dosavadního řešení je struktura TCALL, která reprezentuje hovor. V tabulce 4.1 jsou vypsány všechny položky této struktuty — určené původně pouze pro SIP.

Typ[Délka]	Název	Popis
`char[256]`	`call_id`	Unikátní identifikátor hovoru
`pid_t`	`child_pid`	Číslo procesu, který hovor nahrává
`SIP_STATUS`	`sip_status`	Aktuální stav hovoru
`char[PATH_MAX]`	`dump_file`	Název souboru, do kterého se zapisují packety hovoru
`char[INET6_ADDRSTRLEN]`	`src_ip`	Zdrojová IP adresa zvukových dat
`char[INET6_ADDRSTRLEN]`	`dst_ip`	Cílová IP adresa zvukových dat
`unsigned int`	`src_port`	Zdrojový port zvukových dat
`unsigned int`	`dst_port`	Cílový port zvukových dat
`char[256]`	`src_sip_uri`	Zdrojová SIP URI hovoru
`char[256]`	`src_dst_uri`	Cílová SIP URI hovoru
`char[64]`	`src_num`	Číslo volajícího
`char[64]`	`dst_num`	Číslo volaného
`int`	`format`	Číslo kodeku zvukových dat
`timeval`	`invited`	Čas zachycení INVITE žádosti (začátek vyzvánění)
`timeval`	`started`	Čas zachycení OK odpovědi (začátek nahrávání)
`timeval`	`finished`	Čas zachycení BYE žádosti (konec nahrávání)
`timeval`	`rejected`	Čas zachycení žádosti CANCEL (odmítnutí hovoru)
`timeval`	`crashed`	Nepoužito, v nové verzi použito pro případ, kdy se nepovede začít hovor nahrávat

Tabulka 4.1: Původní struktura TCALL, tabulka 7.1 v [1]

4.2 Můj přínos

Moje práce doplňuje stávající projekt o podporu protokolu Skinny. To si vyžádalo podstatné úpravy původního zdrojového kódu, protože nebyl psán s předpokladem, že bude někdy potřeba pracovat i s jinými protokoly.

Díky tomu, že zdrojový kód této aplikace byl už od začátku stavěn pouze pro SIP, bylo nutné nejprve nějakým způsobem abstrahovat práci s jednotlivými hovory, které kromě SIP můžou být i v jiném protokolu. Všechny tyto potřebné změny jsou popsány v kapitole Implementace 6.

Kapitola 5

Skinny protokol

Tato kapitola se věnuje detailně struktuře a jednotlivým zprávám Skinny protokolu.

5.1 Struktura zpráv

Zprávy Skinny protokolu mají společnou hlavičku, která se skládá se dvou čtyřbytových položek. První určuje velikost celkové zprávy v bytech, druhá pak verzi Skinny protokolu. Tato verze určuje, jak budou vypadat další zprávy, jejichž struktura se může v každé verzi lišit.

Ve všech zdrojích, které jsem měl k dispozici, jsem našel pouze informaci o tom, že jako verze se standardně používá 0 (nazývaná někdy jako „basic“) [12]. V některých zdrojích bylo dokonce uvedeno, že tyto čtyři byty jsou „vyhrazeny“ („reserved“) a obsahují vždy nuly. V reálném prostředí byla tato verze ovšem 22.

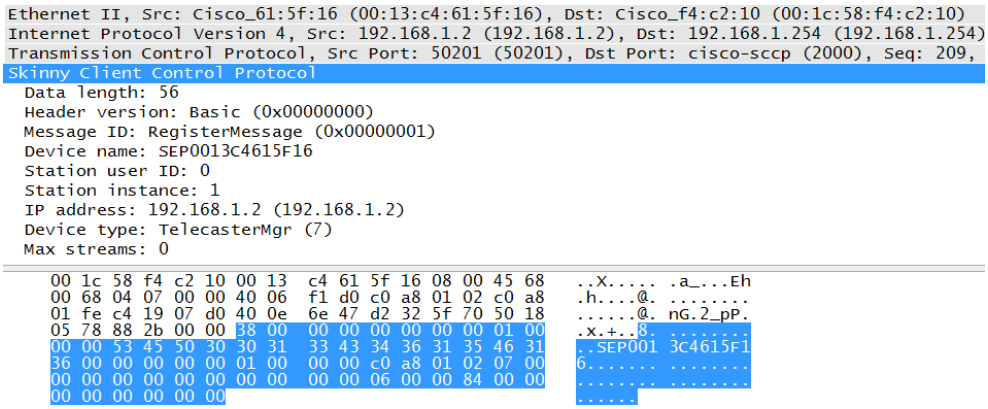
Další položkou je čtyřbytová hodnota, která určuje typ zprávy. Následující byty už jsou specifické pro každou zprávu. Obecně se dá říct, že většina položek má právě čtyři byty, pouze pokud se jedná o řetězec, tak ve verzi 0 je řešen jako pole bytů s pevnou délkou, kde každý byte je jeden znak v ASCII kódu, a byte s hodnotou 0 slouží jako ukončení řetězce. Ve verzi 22 jsou řetězce také ukončeny hodnotou 0, ale není zde pevná délka. To ztěžuje čtení zpráv novějších verzí, protože se nelze spolehnout, že požadovaná hodnota bude na předem známé pozici.

Třetí čtyřbytovou položkou je messageId — číselný kód určující typ zprávy. Následují samotná data zprávy uvedené velikosti, zmenšené o 4 byty; odečítají se proto, že ve velikosti zprávy je zahrnuta i položka určující typ zprávy. Podle tohoto údaje lze určit, jestli v packetu jsou další zprávy nebo ne. Základní strukturu každé zprávy ve Skinny protokolu a velikosti jednotlivých položek v každé zprávě zobrazuje tabulka 5.1

Příklad jedné takové zprávy, kde je krásně vidět výše popsaná struktura, je na obrázku 5.1, jedná se o registrační zprávu posílanou telefonem Call Ma-

Velikost zprávy	Verze protokolu	Typ zprávy	Samotná data zprávy
4 byty	4 byty	4 byty	
		počet bytů = Velikost zprávy (hodnota v prvních čtyřech bytech)	

Tabulka 5.1: Základní struktura Skinny zprávy



Obrázek 5.1: Screenshot z aplikace Wireshark ukazující jednu z mnoha možných Skinny zpráv [5].

nageru a zachycenou programem Wireshark. Právě díky Wiresharku jsem byl schopen všechny Skinny zprávy přečíst a zpracovat. Dissector pro protokol Skinny je podle zdrojového kódu schopen rozeznat 159 typů Skinny zpráv. Není ale třeba se v aplikaci zabývat všemi zprávami; seznam těch nejdůležitějších, které mají vliv na odchytávání hovoru, je v tabulkách 5.2 a 5.3.

5.1.1 Bezpečnost při parsování zpráv

Kvůli binárnímu formátu Skinny zpráv je potřeba při parsování dávat pozor na přetečení, tedy jestli skutečná velikost zprávy odpovídá očekávané velikosti zprávy a že nebudou čteny byty, které už nepatří do packetu, z jiné části paměti. To platí jak pro čtyřbytové číselné položky tak i pro položky obsahující řetězec s fixní velikostí. V případě novější verze protokolu, kde jsou řetězce ukončeny znakem 0 a nemají fixní velikosti, je třeba kontrolovat, jestli řetězec skončil před koncem packetu. V případě absence těchto kontrol je možné, aby někdo poslal po síti speciálně upravený packet a způsobil přetečení (*buffer overflow*) v programu. Běžným důsledkem je pád aplikace, v extrémním případě tato programátorská chyba může vést ke spuštění kódu v tomto upraveném packetu na zařízení, které se snaží tento packet přečíst. Jisté verze systému od společnosti Cisco trpěly touto zranitelností a v případě úspěšného útoku mohlo dojít k restartu Call Manageru [9]. Moje implementace parsování

Kód zprávy	Název zprávy	Stručný popis
Zprávy posílané z telefonu na Call Manager		
`0x0000`	*KeepAliveMessage*	Kontrola dostupnosti telefonu
`0x0001`	*RegisterMessage*	Registrace telefonu k příslušnému Call Manageru
`0x0003`	*KeypadButtonMessage*	Stisknutí číselné klávesy při vytáčení
`0x0006`	*OffHookMessage*	Zvednutém sluchátka
`0x0007`	*OnHookMessage*	Položení sluchátka
`0x0022`	*OpenReceiveChannelAck*	Potvrzení otevření portů pro přijímání zvukových dat
`0x0154`	*StartMediaTransmissionAck*	Potvrzení otevření portů pro odesílání zvukových dat

Tabulka 5.2: Seznam nejdůležitějších zpráv pro odchytávání hovoru, posílaných z telefonu na Call Manager, další zprávy je možné najít v [4].

Skinny hlídá před přečtením každé položky, jestli nedošlo k přetečení, které, pokud je detekováno, způsobí vypsání chyby a ignorování packetu.

5.2 Životní cyklus Skinny telefonu

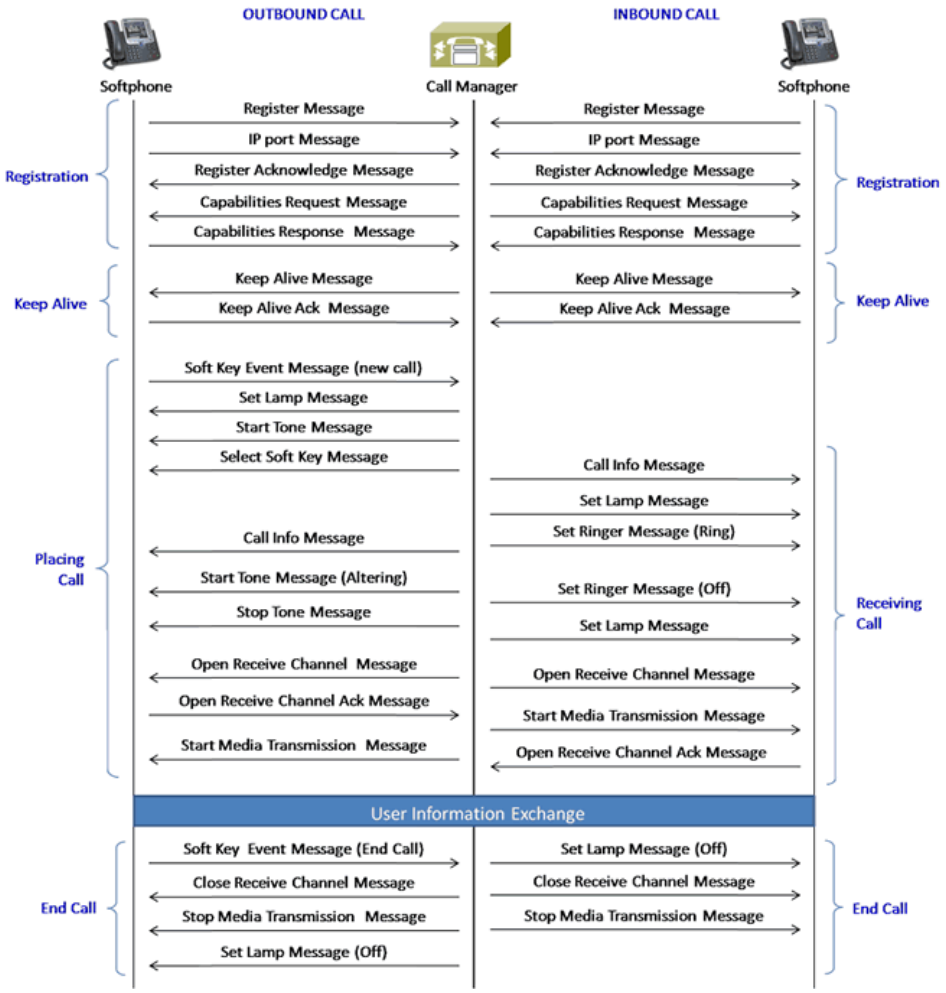
Po připojení Cisco telefonu do sítě LAN dojde k žádosti o IP adresu přes protokol DHCP. Server, který na tuto žádost odpovídá může být úplně jiný, než Call Manager. Důležité ale je, že tato DHCP odpověď obsahuje kromě požadované IP adresy telefonu i IP adresu TFTP serveru (jako DHCP option 150), ze kterého si následně telefon stáhne obraz systému a další nutné systémové konfigurační soubory, které bude poté používat [5].

Po stažení těchto dat se telefon zaregistruje u Call Manageru. Postup registrace a popis zpráv, které si při ní vyměňují telefon s Call Managerem není zde potřeba uvádět. V případě zájmu je možné více informací najít například v kapitole „SCCP Call Flows“ (Appendix D) v [13]. Po úspěšné registraci posílá telefon Call Manageru veškeré události, na které Call Manager reaguje dalšími instrukcemi. To je jeden z hlavních rozdílů oproti SIP — ve Skinny fungují telefony jako jakési „opravdu hloupé terminály“ [12], které odesílají zprávy o stiscích všech tlačítek a jiných událostech a očekávají od Call Manageru instrukce, co zobrazit na displeji, jaké tóny přehrávat, jak a kdy vyzvánět nebo na jakých portech vysílat nebo přijímat data hovoru. Podrobnější popis vysílaných zpráv je na obrázku 5.2.

V [14] jsou také přesně popsány jednotlivé zprávy a jejich významy. Zde se budu soustředit pouze na ty, které mají nějaký vliv na samotný průběh

Kód zprávy	Název zprávy	Stručný popis
Zprávy posílané z Call Manageru na telefon		
0x0081	*RegisterAckMessage*	Potvrzení žádosti o registraci telefonu
0x0085	*SetRingerMessage*	Nastavení stavu vyzvánění
0x008A	*StartMediaTransmission*	Požadavek na začátek odesílání zvukových dat
0x008B	*StopMediaTransmission*	Požadavek na ukončení odesílání zvukových dat
0x008F	*CallInfoMessage*	Informace o hovoru, obsahuje čísla volaného a volajícího, informace o přesměrování a další, nevýznamné pro další zpracování
0x0099	*DisplayTextMessage*	Požadavek na zobrazení textu na displeji
0x009A	*ClearDisplay*	Požadavek na vymazání textu na displeji
0x0100	*KeepAliveAckMessage*	Potvrzení přijetí zprávy *KeepAliveMessage*
0x0105	*OpenReceiveChannel*	Požadavek na otevření portů pro přijímání zvukových dat
0x0106	*CloseReceiveChannel*	Požadavek na uzavření portů pro přijímání zvukových dat
0x0111	*CallStateMessage*	Informace o změně stavu hovoru, obsahuje položku callState, která určuje požadovaný stav
0x011D	*DialedNumberMessage*	Informace o právě vytočeném čísle, odeslána po zadání všech čísel jako potvrzení vytočení čísla
0x014A	*CallInfoMessageV2*	Stejné jako *CallInfoMessage*, pouze s dynamickou délkou řetězců (ukončených znakem 0, ne statickou délkou, jako u *CallInfoMessage*)

Tabulka 5.3: Seznam nejdůležitějších zpráv pro odchytávání hovoru, posílaných z Call Manageru na telefon, další zprávy je možné najít v [4].

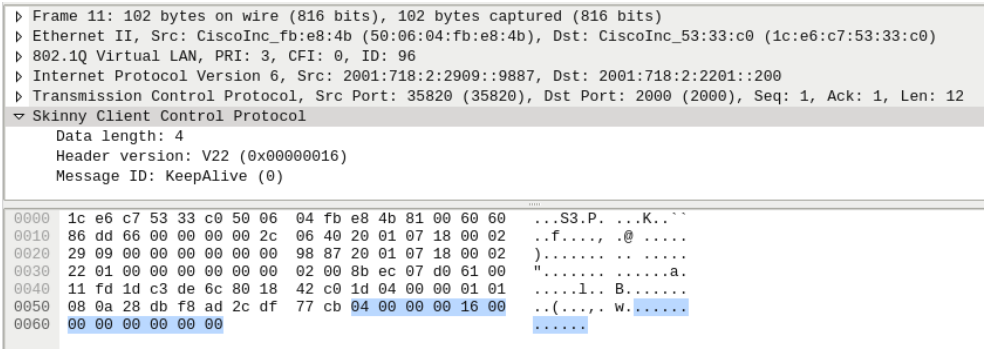


Obrázek 5.2: Graf časové posloupnosti vysílaných Skinny zpráv v průběhu typického hovoru [15]. Na obrázku jsou chybně vyznačeny zprávy *KeepAlive* a *KeepAliveAck*, správně by měly být opačně. Zpráva *KeepAlive* je posílána z telefonu na Call Manager a *KeepAliveAck* z Call Manageru na telefon

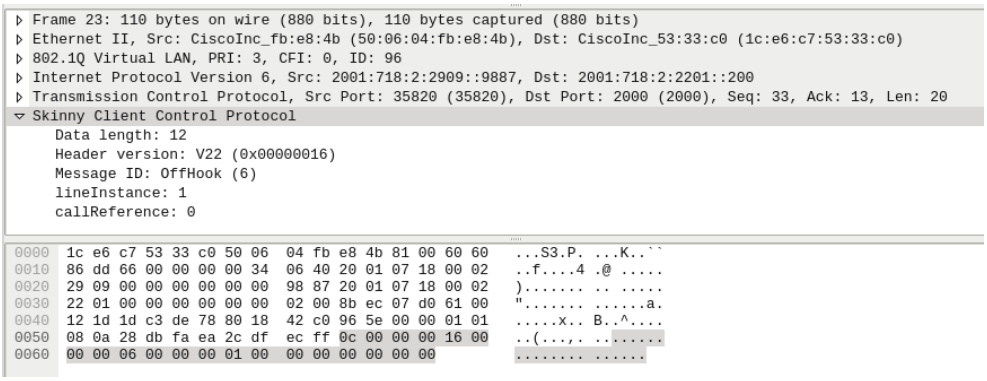
hovoru. Některé zprávy obsahují pouze typ zprávy a v podstatě žádná další relevantní data (např. *OffHook* — pouze informuje Call Manager, že bylo zvednuto sluchátko) a některá i několik desítek dalších položek s detailními informacemi (např. *CallInfo* — obsahuje informace kdo komu volá, a to jak číslo, tak název, informace směru hovoru a přesměrování).

5.2.1 Popis jednotlivých zpráv

Vzhledem k tomu, že se jedná o binární protokol, jsou jednotlivé položky pojmenovány pro lepší referenci. V následujícím textu budu tedy místo kon-



Obrázek 5.3: Screenshot z programu Wireshark zobrazující Skinny zprávu *KeepAlive*



Obrázek 5.4: Screenshot z programu Wireshark zobrazující Skinny zprávu *OffHook*

krétních číselných údajů (typu pátá čtveřice bytů ve Skinny zprávě) a hodnot používat označení klíčovými slovy, která jsou běžně používaná v literatuře věnující se tomuto protokolu. Konkrétní informace o tom, o jaké se jedná hodnoty, lze dohledat například ve zdrojovém kódu dissectoru Wiresharku [4].

Před samotným hovorem (a po něm) si posílají telefon s Call Managerem pouze zprávy *KeepAlive* (messageId = 0, z telefonu na Call Manager) a *KeepAliveAck* (messageId = 256, z Call Manageru na telefon). Jedná se o zprávy s nulovou délkou, takže každá zpráva se skládá pouze z 12 bytů, jak je vidět na obrázku 5.3.

Začátek hovoru a vytočení čísla typicky začíná zvednutím sluchátka. Po zvednutí sluchátka je vyslána z telefonu zpráva *OffHook* (obrázek 5.4).

Z Call Manageru následují jako odpověď zprávy *SetLamp*, *SelectSoftKeys*, *DisplayPromptStatus*, které nejsou svým obsahem důležité pro samotný hovor. Tyto zprávy mohou být odeslány najednou v jednom packetu, nebo každá zpráva v samostatném packetu. Při testování byly pozorovány obě varianty.

Důležitá je až zpráva *CallState* s hodnotou položky callState, která je nastavena na hodnotu *OffHook* (prakticky se jedná o čtvrtou čtveřici bytů ve Skinny zprávě, ihned za hlavičkou a identifikátorem zprávy, s hodnotou 1). V této zprávě se poprvé objevuje položka callReference, která je dále přítomná v každé další zprávě vázané ke konkrétnímu hovoru. Jedná se v podstatě o unikátní identifikátor hovoru, respektive identifikátor spojení mezi telefonem a Call Managerem. Pro jeden hovor mezi dvěma telefony komunikujícími přes jeden Call Manager se nejedná totiž o stejnou hodnotu — Call Manager komunikuje s každým telefonem pomocí jiné hodnoty callReference. Díky tomu je obtížnější identifikovat, že se jedná o stejný hovor a ne o dva. Tento problém je detailně vysvětlen v kapitole 6.4.

Po této zprávě posílá telefon po každém stisknutí (číselné) klávesy zprávy *KeypadButton* s informací o tom, které číslo bylo stisknuté. Po odeslání celého vytáčeného čísla Call Manager odpoví zprávou *DialedNumber*, která obsahuje informaci o kompletním vytočeném čísle. Tato zpráva byla v prvních verzích implementace rozhodující pro začátek signalizace hovoru, později byla detekce upravena na zprávu *CallState*, která oznamuje logickou změnu hovoru, s položkou callState s hodnotou nastavenou na *RingOut* nebo *RingIn*. Změna byla zvolena hlavně kvůli konzistenci s ostatními změnami hovoru.

Následující odstavce ilustruje diagram 5.5 znázorňující zprávy, které jsou použity k detekci hovorů přes Skinny protokol.

Pro detekci první zmínky o hovoru (a uložení informací o hovoru do databáze) používá můj program zprávu *CallState* s hodnotou callState nastavenou na *RingIn* (pro příchozí hovor) nebo *RingOut* (odchozí hovor). Jak je z názvu patrné, tato zpráva signalizuje začátek zvonění telefonu a tedy je logickým ekvivalentem INVITE zprávy v protokolu SIP. Po zachycení této zprávy je hovor registrován v programu a v připojené databázi. Tyto zprávy byly pro signalizaci začátku hovoru zvoleny ještě z jiného důvodu — taková zpráva se v průběhu signalizace vyskytne právě jednou (na začátku a nikdy jindy), a jednoznačně signalizuje začátek nového hovoru.

Další důležitou zprávou je *CallInfo*, respektive *CallInfoV2* (významem stejné jako *CallInfo*, jenom použito ve verzi 22 místo *CallInfo*, jediným rozdílem je formát řetězců, které nemají pevnou délku). Tuto zprávu posílá Call Manager telefonům, a to nejen při startu hovoru, ale i několikrát v jeho průběhu. Obsahuje informace o probíhajícím hovoru, a to zejména číslo volaného a volajícího, které jsou po zachycení této zprávy uloženy k zachytávanému hovoru. Identifikaci hovoru určuje opět položka callReference přítomná ve všech těchto zprávách. Ani po této zprávě ale program nemá dostatek informací k tomu, aby věděl, jak má hovor nahrávat. Chybí totiž informace o IP adresách a portech, ze kterých budou telefony posílat zvuková data a formát těchto zvukových dat, který bude uložen do databáze.

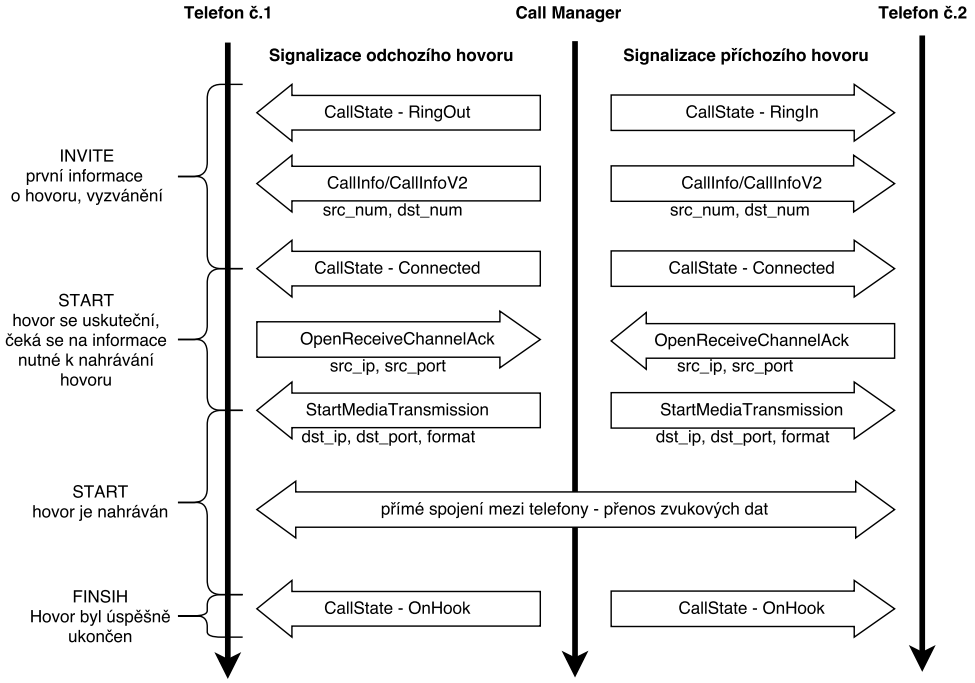
Následuje znovu zpráva *CallState*, tentokrát s hodnotou callState nastavenou na *Connected*, poslaná od Call Manageru oběma telefonům. Na základě této zprávy označí program hovor za nahrávaný. Bohužel ještě stále nejsou

k dispozici IP adresy a porty odesílatele a příjemce.

Tyto údaje obsahuje následná série zpráv mezi Call Managerem a telefony. Nejprve pošle Call Manager telefonům zprávu *OpenReceiveChannel* s informací o požadovaném formátu zvukových dat. Vzhledem k tomu, že stejná informace je následně poslána ve zprávě *StartMediaTransmission*, je tato zpráva ignorována. Odpovědí je ale zpráva *OpenReceiveChannelAck*, která obsahuje zdrojovou IP adresu a port, odkud budou telefony vysílat. Call Manager následně posílá zprávu *StartMediaTransmission*, která obsahuje kromě formátu zvukových dat také cílovou IP adresu. Tím je výčet všech potřebných informací kompletní. Telefony sice ještě posílají zprávu *StartMediaTransmissionAck* potvrzující předchozí zprávu z Call Manageru, ale tato zpráva už neobsahuje žádné nové informace — pouze už známou zdrojovou IP adresu a port.

Jakmile jsou tedy informace známy, může začít odposlouchávání samotného hovoru a ukládání dat do souboru, což už řeší existující implementace. Protokol RTP pro přenos zvukových dat je stejný pro Skinny i pro SIP.

Pro ukončení hovoru se jako rozlišující zpráva bere opět zpráva *CallState*, tentokrát ovšem s položkou callState nastavenou na hodnotu *OnHook*. Jako lepší zpráva se původně zdála být *StopTransmission* (ukončení přenosu dat), protože ta implicitně znamená ukončení hovoru, ale tato zpráva je posílána i při podržení hovoru (hold), kdy je posílání zvukových dat pouze dočasně pozastaveno. Nelze ji tedy použít jako signalizaci konce hovoru.



Obrázek 5.5: Časový diagram zachycovaných Skinny zpráv mezi telefony a Call Managerem během typického hovoru

Kapitola 6

Implementace

6.1 Propojení s existujícím projektem

Nejprve bylo potřeba vymyslet, jak rozšířit stávající implementaci o novou funkcionalitu — Skinny protokol. Bylo nutné nějakým způsobem abstrahovat strukturu TCALL, která reprezentuje nahrávaný hovor, aby ji bylo možné použít i pro Skinny protokol. Dále bylo potřeba abstrahovat stav hovoru (položka `call_status`) od konkrétního protokolu. V původním projektu byl stav hovoru totéž, jako poslední zpráva SIP protokolu, což bylo sice vyhovující, ale pro další rozšíření nepoužitelné. Také bylo potřeba zobecnit způsob rozebírání packetu vráceného z knihovny Pcap, protože hlavní funkce pro rozebírání packetu se jmenovala `get_sip_message()`, jako argument měla přímo data z rámce a vracela SIP zprávu vytvořenou z dat v packetu. To bylo potřeba upravit tak, aby výsledkem rozebrání packetu byly přímo data v TCP nebo UDP protokolu, případně ještě informace o zdrojových a cílových IP adresách a portech pro budoucí zpracování nebo párování s předchozími packety, a teprve potom předat packet k dalšímu zpracování a parsování konkrétního VoIP protokolu.

6.2 Zobecnění struktury TCALL

Struktura TCALL obsahuje všechny potřebné informace o aktuálně prováděném hovoru. Původně obsahovala kromě obecných informací o hovoru, které se dají použít i pro jiné protokoly než SIP, také položky `src_sip_uri` a `dst_sip_uri` označující SIP adresu volajícího a volaného. Tyto položky byly odstraněny. Dále byly do této struktury přidány informace o protokolu, kterého se hovor týká (v současné době pouze Skinny nebo SIP) a IP adresy signalizačních packetů (`sig_src_ip` a `sig_dst_ip`), které byly potřeba pro spárování nebo kontrolu zachycených packetů — tj. jestli zachycené packety s daným identifikátorem hovoru opravdu patří k danému hovoru. Spárování je potřeba v případě, kdy zpráva neobsahuje identifikátor hovoru; konkrétně

Skinny zpráva *OpenReceiveChannelAck* ve verzi protokolu 0 neobsahuje callReference.

6.3 Zobecnění stavu hovoru

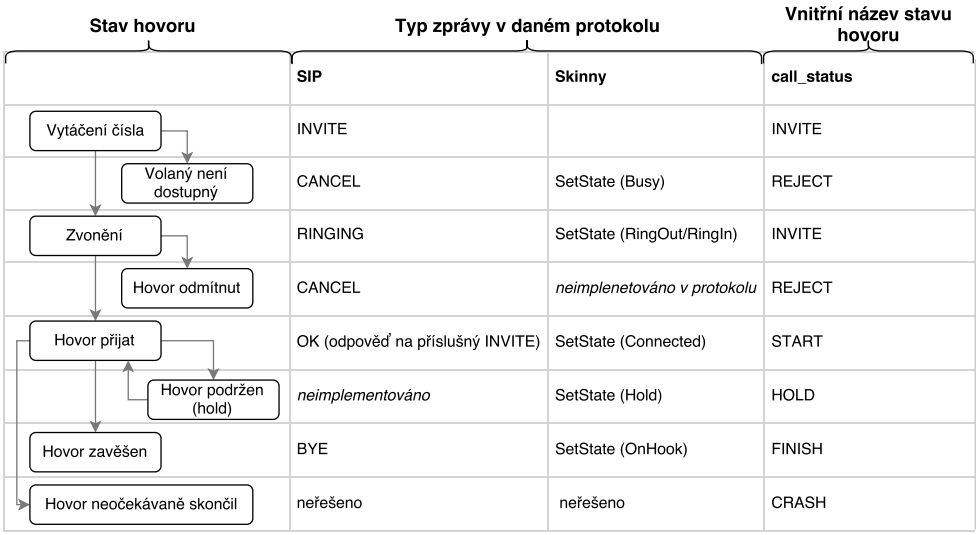
V původní implementaci stav hovoru, tj. hodnota proměnné `call_status` ve struktuře TCALL, reprezentoval poslední SIP zprávu, což ale není přijatelné, pokud má TCALL sloužit pro více protokolů. Původní výčet INVITE, RINGING, OK, CANCEL a BYE jsem tedy upravil na INVITE, REJECT, START, CRASH a FINISH. Později jsem ještě přidal stav HOLD označující podržený (anglicky „hold“) hovor a IGNORED pro označení hovoru, jehož zachycené zprávy se mají z nějakého důvodu ignorovat. V případě Skinny hovoru na jedné síti, kdy jsou zachyceny signalizační zprávy volaného i volajícího, jsou totiž vytvořeny dva TCALL hovory, protože nelze předem určit, že se jedná o stejný hovor (hodnota callReference se u těchto dvou hovorů liší a není jisté, jestli se lze spolehnout na čísla volaného a volajícího). Čeká se tedy na informace o IP adresách a portech, na kterých se budou vysílat zvuková data. Tyto informace jednoznačně určují stejný hovor. Pokud se zjistí, že už jeden hovor se stejnými IP adresami a porty existuje, jeden z nich se označí jako IGNORED, aby se zamezilo vytvoření duplicitních záznamů v databázi a nahrávání jednoho hovoru dvakrát. Tato situace je více rozepsaná v následující kapitole.

Stav INVITE reprezentuje hovor, o kterém se ví, že bude probíhat, ale ještě nedošlo ke spojení obou stran — typicky se jedná situace mezi vytočením čísla, během vyzvánění a před zvednutím sluchátka na druhé straně.

Do stavu REJECT je hovor nastaven, pokud je druhou stranou odmítnut. Nerozlišuje se důvod odmítnutí, může se jednat o explicitní odmítnutí uživatelem, nedostupnost nebo obsazenost druhého telefonu.

Stav START reprezentuje probíhající a aktuálně nahrávaný hovor, případně hovor, který se bude nahrávat, ale ještě nejsou k dispozici informace o IP adresách a portech, kde bude komunikace probíhat. Tato situace typicky nastává v okamžiku po přijetí Skinny zprávy *CallState* Connected, kdy je jasné, že jsou telefony spojeny, ale ještě nedošlo k výměně IP adres a portů ve zprávách *StartMediaTransmission* a *OpenReceiveChannelAck*.

Pokud se hovor nezdaří začít nahrávat, nastaví se jeho stav na CRASH. Původní verze programu byla už na tuto situaci připravena formou časového razítka v proměnné `crashed` struktury TCALL, která ale zůstala nepoužita. V mém rozšíření jsem ji použil pro situaci, kdy se nepodaří vytvořit proces pro nahrávání hovoru nebo se nepodaří otevřít soubor, do kterého se nahrávána data mají zapisovat. Od tohoto okamžiku je hovor ignorován, stejně jako v případě IGNORED, kromě toho je ale v databázi záznam o času, kdy došlo k přechodu do stavu CRASH (ve sloupci crashed).



Obrázek 6.1: Časový diagram stavů hovorů, přechodů mezi nimi, a zpráv v protokolu SIP a Skinny způsobujících tyto přechody, včetně vnitřního stavu hovoru nezávislého na protokolu

A konečně — stav FINISH — hovory v tomto stavu jsou považovány za úspěšně ukončené. Tyto hovory jsou odstraněny z paměti a soubor pro zapisování zachycovaných zvukových dat je uzavřen.

Na obrázku 6.1 jsou znázorněny přechody mezi stavy hovoru a příslušné zprávy v protokolech SIP a Skinny, které tyto přechody způsobují.

6.4 Problém při detekci hovoru v jedné síti

Na rozdíl od protokolu SIP, ve kterém komunikují telefony mezi sebou přímo, ve Skinny probíhá signalizační komunikace přes Call Manager, který si sám určuje identifikátor hovorů. V případě jednoho hovoru zvolí jiný identifikátor (callReference) pro volající telefon a pro volaný telefon. To je problém, pokud jsou oba telefony v jedné síti a je zachycena komunikace obou telefonů s Call Managerem. Nelze totiž určit, jestli se jedná o stejný hovor a jeden telefon volá druhý, nebo se jedná o dva nezávislé hovory, jejichž druhé strany nejsou v dané síti a jejichž signalizační packety nejsou tudíž zachyceny. Minimálně v prvních signalizačních zprávách není žádná informace, podle které by se dalo poznat, že zprávy patří stejnému hovoru. Už při prvních Skinny zprávách se ukládá hovor do databáze, ale až v okamžiku, kdy si telefony s Call Managerem vymění IP adresy a porty, na kterých budou vysílat zvuková data, lze jednoznačně učit, že se jedná o jeden hovor zachycený dvakrát. Pro tento problém jsem zvážil tyto možnosti řešení:

1. Odložit zapsání příchozích hovorů do databáze.

   V případě, že byla zachycena zpráva signalizující příchozí hovor (*CallState* RingIn), by hovor nebyl vložen do databáze, ale počkalo by se, až budou známy IP adresy a porty, a ověřilo by se, že už tento hovor není zachycen jako odchozí. Toto řešení má zjevnou nevýhodu v tom, že v databázi by se příchozí hovory objevovaly až v okamžiku začátku hovoru, nikoli již v okamžiku vyzvánění telefonu, jako pro všechny ostatní hovory. Toto řešení bylo tudíž zavrhnuto.

2. Pokusit se spárovat odchozí a příchozí hovor.

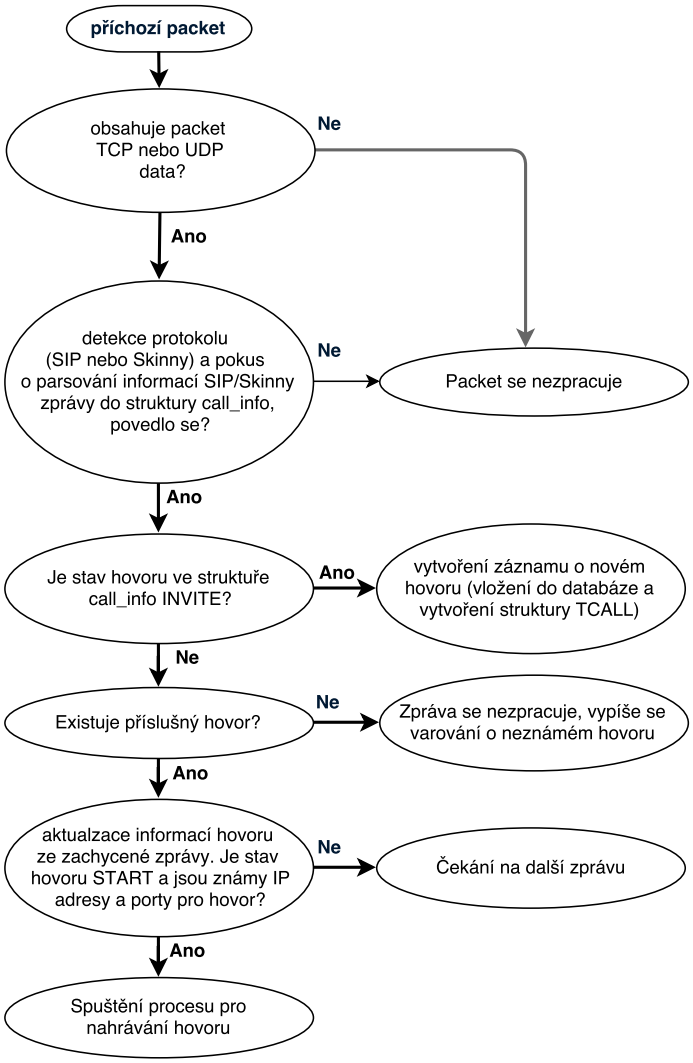
   Jediná možnost, jak rozpoznat, že příchozí hovor je už jednou zachycen jako odchozí, je zmíněná položka callReference. Jiná položka ve zprávách *RingOut* nebo *RingIn* ani není. Bohužel se tato čísla pro příchozí a odchozí hovor liší. V reálné komunikaci odchycené na skutečných telefonech bylo vypozorováno, že se tato čísla liší právě o 1. Call Manager nejspíš tato čísla generuje inkrementálně pro každou novou komunikaci. Bohužel nelze určit, jestli je to pravidlem. I pokud by to pravidlem bylo, může se stát, že při vysokém provozu vygeneruje Call Manager mezi identifikátorem hovoru pro odchozí a příchozí hovor ještě jeden nebo několik dalších identifikátorů pro úplně jiné hovory.

   Ještě před výměnou IP adres a portů Call Manager vysílá zprávu *CallInfo* (nebo *CallInfoV2*), ve které obou zúčastněným stranám oznamuje informace o hovoru včetně volaného a volajícího čísla. Podle těchto čísel by bylo nejspíš možné také spárovat příchozí a odchozí hovor do jednoho. Bohužel není jisté, jestli jsou tato čísla poslána oběma telefonům stejná. Takže i přesto, že se tato zpráva posílá před zprávou *CallState RingIn/RingOut*, nelze jednoznačně určit, jestli se na tuto informaci lze spolehnout.

   Řešení spárováním odchozího a příchozího hovoru ihned po detekci bylo tedy také zavrhnuto.

3. Vložit oba hovory do databáze a po zjištění duplicity jeden smazat.

   Tato možnost spočívá v tom, že se oba zachycené hovory (respektive jeden hovor zachycený dvakrát) uloží do databáze, a teprve v okamžiku, kdy se jednoznačně podle IP adres a portů pozná, že se jedná o stejný hovor, se z databáze jeden smaže. Je tedy potřeba jeden z těchto hovorů označit jako ignorovaný (stav IGNORED), aby další zprávy patřící tomuto hovoru byly rozpoznány a nevytvářely zbytečně chybové hlášení o neexistujícím hovoru. Toto řešení bylo nakonec zvoleno jako nejlepší a implementováno — i přes zjevnou nevýhodu, že na krátký okamžik budou v databázi dva hovory.



Obrázek 6.2: Zjednodušený diagram průběhu odchytávání obecného hovoru nezávisle na protokolu

6.5 Práce s packety a struktura call_info

Kromě změny struktury reprezentující hovor bylo potřeba také abstrahovat změny, které vyvolají implementované protokoly. Pro tuto situaci jsem vytvořil strukturu call_info, která se svým obsahem podobá struktuře TCALL. Na diagramu 6.2 je zobrazen zjednodušený algoritmus pro zachytávání signalizačních dat hovorů. Ve skutečnosti je situace o něco složitější, ale pro základní představu je tento diagram dostačující.

6.6 Změny v databázi

Projekt využívá *PostgreSQL* databázi a všechny přístupy se dějí formou uložených procedur, které se volají z kódu aplikace. Tento přístup nebyl nijak změněn, pouze byly upraveny argumenty funkce `updateCallStarted()` pro úpravu údajů o hovoru. Bylo totiž potřeba, aby umožňovala vložit do databáze informace o volaném a volajícím čísle. V původní verzi (pouze se SIP) se totiž počítalo s tím, že tyto informace jsou známy už při vkládání hovoru do databáze (po SIP zpráv INVITE tomu tak opravdu je), což ale neplatí u Skinny, kde jsou tyto informace dostupné až ze zprávy *CallInfo* (nebo *CallInfoV2*). Ta je ale odeslána a zachycena až po zprávě *CallState* (s hodnotou callState nastavenou na Connected), která teprve způsobuje vložení záznamu o hovoru do databáze.

Jedinou další změnou v databázi je nová uložená procedura `deleteCall()` pro mazání hovoru.

Kapitola 7

Testování

Při implementaci nového protokolu byl kladen důraz i na zachování stávající funkcionality. Po nutných změnách, mnohdy zásadních, nadále funguje odchytávání protokolu SIP.

7.1 Nové testovací soubory

Git repozitář dosavadního řešení obsahoval soubory se zachycenými SIP packety. Jedná se o soubory s příponou pcap, zachycené nástrojem tcpdump. Tyto soubory lze bez problému otevřít a zachycené packety si prohlížet například pomocí programu Wireshark a slouží primárně pro účely testovaní, aby bylo možné testovat program i bez dostupnosti sítě se zapojenými telefony. Tyto soubory je program schopen i po přidání podpory pro nový protokol správně zpracovat.

Do repozitáře byly ke stávajícím souborům se SIP protokolem přidány nové, obsahující protokol Skinny. Kromě standardního hovoru jsem se snažil vytvořit soubor pro každý možný stav, který může při hovoru nastat. Zde je tedy výčet nových souborů a situací, které mohou nastat a program je schopen je vyřešit a korektně zpracovat.

- skinny-incomming-mobile.pcap — příchozí hovor z externí sítě (z mobilního telefonu) na Skinny telefon v místní síti (LAN). V souboru je zachycena komunikace mezi Skinny telefonem a Call Managerem a následně zvuková data hovoru, také mezi Skinny telefonem a Call Managerem, který v tomto případě slouží jako prostředník.

- skinny-outgoing-mobile.pcap — odchozí hovor ze Skinny telefonu v místní síti na mobilní telefon v externí síti. Prakticky stejná data jako v předchozím souboru, ale místo signalizace příchozího hovoru je mezi Call Managerem a Skinny telefonem signalizace odchozího hovoru.

- skinny-outgoing-mobile-redial.pcap — stejné jako u předchozího souboru, s jediným rozdílem, že je využito tlačítko pro znovu vytočení čísla v předchozím hovoru. Původně v souboru měla být signalizace odmítnutého hovoru, ale bylo zjištěno, že pokud se hovor týká mobilního telefonu, případně telefonu v síti nějakého jiného operátora, hovor je v Call Manageru považován za přijatý už při vyzvánění. Nelze tedy pouze z pohledu signalizace packetů v LAN určit, zda byl hovor reálně uskutečněn, nebo pouze vyzváněl. Šlo by to nejspíš určit z analýzy zvukových dat, ale to už je mimo rámec této práce.

- skinny-samenet.pcap — Signalizace a hovor mezi Skinny telefony ve stejné místní síti. Soubor obsahuje signalizaci mezi jedním Call Managerem a dvěma telefony. Jedná se tedy o dvě různé signalizace (jedna pro odchozí a druhá pro příchozí hovor).

- skinny-hold.pcap — Hovor mezi Skinny telefonem v místní síti a mobilním telefonem, který je v průběhu podržen (hold) a následně znovu obnoven.

7.2 Problém s verzemi Skinny protokolu

Vzhledem k uzavřenosti protokolu bylo možné najít dokumentaci pouze pro starší verze, konkrétně většina zdrojů se shodovala s tím, že položka pro verzi v hlavičce Skinny zpráv obsahuje pouze nuly a/nebo je rezervována. Většina informací se tedy týká převážně pouze verze 0. Podle zdrojového kódu Wiresharku a dalších zdrojů existují i jiné verze, ale o těch se mi nepodařilo najít podrobnější informace. O verzi 22, se kterou jsem se reálně setkal na testovacích telefonech, jsem bohužel nenašel informace skoro žádné. Naštěstí se tolik neliší od předchozích verzí a minimálně v hlavních signalizačních zprávách se formát zpráv úplně shoduje s celkem dobře dokumentovanou verzí 0.

7.3 Podpora pro podržený hovor

V původním návrhu byl hovor brán pouze jako spojení dvou telefonů, které začne vyzváněním, následuje výměna informací a poté výměna zvukových dat, tj. samotný průběh hovoru. To je bohužel přílišné zjednodušení. Existuje funkce podržení hovoru, která umožňuje probíhající hovor pozastavit a po nějaké době znovu obnovit. Ve Skinny protokolu se tato skutečnost signalizuje zprávou *StopMediaTransmission* poslanou z Call Manageru při pozastavení hovoru a zprávou *StartMediaTransmission* při obnovení hovoru. Tyto zprávy nelze tedy požadovat za začátek nebo konec hovoru, jak to bylo v prvních verzích implementace. Novější verze tedy využívá zpráv *CallState*, které obsahují logický stav hovoru.

Zpráva *StartMediaTransmission* a odpověď na ní *StartMediaTransmissionAck* obsahují čísla nových portů. Je potřeba změnit filtr pro odchytávání dat a nastavit pro něj nové porty. Pro odchytávání hlasových packetů každého hovoru se vytváří samostatný proces. Při odchytávání paketů ze sítě v reálném čase hovoru je typicky dost času na odeslání signálu těmto procesům, ale při čtení síťové komunikace ze souboru může signál o změně portů dorazit pozdě — až po tom, kdy nahrávací proces přečte packety s novými porty, které ale ignoruje. Řešením by bylo přepsat procesy na vlákna a využít k tomuto účelu stejný paměťový prostor a zámky. To by znamenalo zásadní, navíc časově dosti náročný zásah do struktury celé stávající aplikace. Jednodušší je nahrávací proces ukončit a pro nové porty založit proces nový. Ten poté pokračuje v zapisování packetů do existujícího pcap souboru. Tím se problém vyřešil a aplikace je schopna zpracovávat podržené hovory.

Závěr

Výsledkem práce je rozšíření aplikace na odchytávání hovorů v protokolu SIP o podporu proprietárního protokolu Skinny Client Control Protocol. I přesto, že se jedná o proprietární protokol a nalezené informace a dokumentace se vztahují spíše ke starším verzím, podařilo se mi implementovat aplikaci schopnou odchytávání komunikace a zároveň tuto funkcionalitu propojit s existujícím projektem. Výsledkem je funkční prototyp schopný rozlišit SIP a Skinny hovory na síti, uložit metadata o všech zachycených hovorech do databáze a packety obsahující zvuková data do souboru. Struktura aplikace je stále naivní: během jednoho hovoru mohou nastat události, které současná implementace ignoruje. Snažil jsem ale vyřešit nejdůležitější situace, které mohou nastat a které se mi podařilo otestovat.

Možná rozšíření

Jedním z možných rozšíření je podpora konferenčních hovorů. Současná architektura vychází z předpokladu, že hovor má právě dva účastníky, pro které hovor začíná a končí ve stejném čase a používá jediný audio formát. Žádný z těchto předpokladů u konferenčního hovoru obecně neplatí. Pro implementaci podpory konferenčních hovorů by bylo potřeba celý projekt přepracovat.

Další možností rozšíření je podpora video hovorů — Skinny protokol umožňuje přenášet kromě zvukových dat i video data a obecně jakákoli data. Jediný rozdíl mezi jednotlivým druhem dat je v podstatě domluva mezi telefony o formátu přenášených dat. Pro video data existují ještě zprávy *StartMultiMediaTransmission*, *StopMultiMediaTransmission* a *StartMultiMediaTransmissionAck* jako alternativy k *StartMediaTransmission*, *StopMediaTransmission* a *StartMediaTransmissionAck*. Tyto zprávy nesou informace nejen o formátu zvukových ale i o formátu video dat. Pro podporu signalizace videa by tedy mělo stačit dekódovat tyto zprávy.

Rozšíření by bylo také možné v podpoře šifrovaných signalizací a hovorů. Standardně jsou totiž všechny signalizační zprávy Skinny hovoru šifrované

pomocí TLS a místo na portu 2000 probíhají na portu 2443. Pro testování a zkoumání reálných hovorů bylo potřeba na straně Call Manageru toto šifrování dočasně pro testované telefony vypnout.

Dále je možné implementovat živý odposlech, tedy umožnit přehrávání právě zachytávaného hovoru. Tato úprava by si ale vyžádala zásadní změny v celé architektuře aplikace, především nějakou formu komunikace mezi nahrávajícím a dekódujícím procesem, které jsou dosud nezávislými komponentami.

Literatura

[1] Šuster, F.: *Extrakce VOIP dat ze síťového provozu*. Bakalářská práce, České vysoké učení technické, 2015.

[2] Kučera, J.: *Extrakce a zpracování hlasu z VOIP paketů*. Bakalářská práce, České vysoké učení technické, 2015.

[3] Robejšek, V.: *Webové rozhraní k VOIP hovorům*. Bakalářská práce, České vysoké učení technické, 2015.

[4] Mayer, J.: Dissector for the Skinny Client Control Protocol. Online, [vid. 2016-04-26]. Dostupné z: `https://github.com/boundary/wireshark/blob/master/epan/dissectors/packet-skinny.c`

[5] Hartpence, B.: *Packet Guide to Voice over IP: A system administrator's guide to VoIP technologies*. O'Reilly Media, 2013, ISBN 978-1-449-33967-8.

[6] Wallace, K.: *Cisco Voice over IP (CVOICE) (Authorized Self-Study Guide) (3rd Edition)*. Cisco Press, 2008, ISBN 1-58705-554-6.

[7] [obrázek] SIP transakce. Online, [vid. 2016-04-25]. Dostupné z: `https://cs.wikipedia.org/wiki/Session_Initiation_Protocol#/media/File:SIP_transakce.png`

[8] Rosenberg, J.; Schulzrinne, H.; Camarillo, G.; aj.: SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), Červen 2002. Dostupné z: `http://www.ietf.org/rfc/rfc3261.txt`

[9] Porter, T.; Jr., J. K.; Baskin, B.: *Practical VoIP Security*. Syngress, 2006, ISBN 1-59749-060-1.

[10] *Network Protocols Handbook*. Javvin Technologies, 2005, ISBN 0974094528.

[11] Git. Online, [vid. 2016-05-03]. Dostupné z: https://git-scm.com/

[12] VoIP Lecture Notes. Online, [vid. 2016-05-10]. Dostupné z: http://www.technologeeks.com/Courses/VoIP.pdf

[13] Cisco: Cisco ATA 186 and Cisco ATA 188 Analog Telephone Adaptor Administrator's Guide (SCCP). 2013, online, [vid. 2016-04-18]. Dostupné z: http://www.cisco.com/c/en/us/td/docs/voice_ip_comm/cata/186_188/2_15_ms/english/administration/guide/sccp/sccp.pdf

[14] Alexander, J.; Pearce, C.; Smith, A.; aj.: *Cisco CallManager Fundamentals (2nd Edition)*. Cisco Press, 2005, ISBN 1-58705-192-3.

[15] [obrázek] Skinny call flow diagram. Online, [vid. 2016-04-26]. Dostupné z: http://www.gl.com/skinny-protocol-emulation-using-maps.html

Příloha A

Seznam použitých zkratek

ASCII American Standard Code for Information Interchange

CD Compact Disk

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

GSM Global System for Mobile Communications

HTML HyperText Markup Language

HTTP Hypertext Transport Protocol

IP Internet Protocol

LAN Local Area Network

NTP Network Time Protocol

PDF Portable Document Format

PoE Power over Ethernet

RFC Request for Comment

RTP Real-time Transport Protocol

SCCP Skinny Client Control Protocol

SIP Session Initiation Protocol

SQL Structured Query Language

TCP Transmission Control Protocol

TFTP Trivial File Transport Protocol

TLS Transport Layer Security

UDP User Datagram Protocol

VoIP Voice over Internet Protocol

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `net` ..... zdrojové kódy aplikace pro odchytávání VoIP dat včetně podpory pro Skinny
    - `doc` ..... Programátorská dokumentace ve formátu HTML
    - `create.sql` ..... SQL skript pro vytvoření tabulek v databázi
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF