Mobilné technológie a aplikácie
Dokumentácia k zadaniu

Lukáš Štrbo
Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
xstrbol@stuba.sk
ID: 110903

1. marec 2022

Zadanie č. 1 : SIP Proxy (telefónna ústredňa)
Cvičiaci: Ing. Marek Galinski, PhD.

Obsah

1 Zadanie

Sprevádzkovanie SIP Proxy, ktorá umožní prepájanie a realizáciu hovorov medzi štandardnými SIP klientami.

Na implementáciu SIP Proxy si môžete zvoliť akýkoľvek programovací jazyk a použiť akúkoľvek SIP knižnicu, ktorá pre daný programovací jazyk existuje. Vo výsledku však musíte spúšťať "váš kód", v ktorom sú zakomponované knižnice, ktoré poskytujú funkcionalitu SIP Proxy. Hovor musí byť realizovaný medzi dvomi fyzickými zariadeniami v rámci LAN siete

Povinné funkcionality

(a) Registrácia účastníka (bez nutnosti autentifikácie)

(b) Vytočenie hovoru a zvonenie na druhej strane

(c) Prijatie hovoru druhou stranou, fungujúci hlasový hovor

(d) Ukončenie hlasového hovoru (prijatého aj neprijatého)

Doplnkové funkcionality

(a) Možnosť zrealizovať konferenčný hovor (aspoň 3 účastníci)

(b) Možnosť presmerovať hovor

(c) Možnosť realizovať videohovor

(d) Logovanie "denníka hovorov"

(e) Úprava SIP stavových kódov v zdrojovom kóde proxy

2 Implementačné prostredie a repozitáre

1. **Repozitáre**
   - **Použitá SIP knižnica:** `https://github.com/tirfil/PySipFullProxy`
   - **Osobný repozitár:** `https://github.com/LukasStrbo/MTAA_Sip_Proxy`
2. **Implementačné prostredie**
   - **IDE:** PyCharm 2021.3.2 (Professional Edition)
   - **Programovací jazyk:** Python 3.9.7
   - **OS:** Microsoft Windows 10 [Version 10.0.19043.1526]
   - **Klient:** Linphone

3 Riešenie zadania

3.1 Implementácia riešenia

Riešenie bolo implementované pomocou SIP knižnice z platformy GitHub (viď. 2). SIP knižnica bola pravdepodobne naprogramovaná pre staršiu verziu programovacieho jazyka Python nakoľko po jej spustení sa vyskytli rôzne výnimky a chyby. Pre to bolo potrebné danú knižnicu opraviť a prispôsobiť ju novým štandardom novšej verzie progamovacieho jazyka Python.

3.2 Povinné funkcionality

PRe spojazdnenie povinných funkcionalít bola potrebná oprava SIP knižnice. Knižnica vykazovala po spustení isté výnimky a chyby. Chyby, ktoré boli v knižnici spôsobené boli hlavne spôsobené tým, že knižnica bola programovaná v staršej verzii jazyka Python.

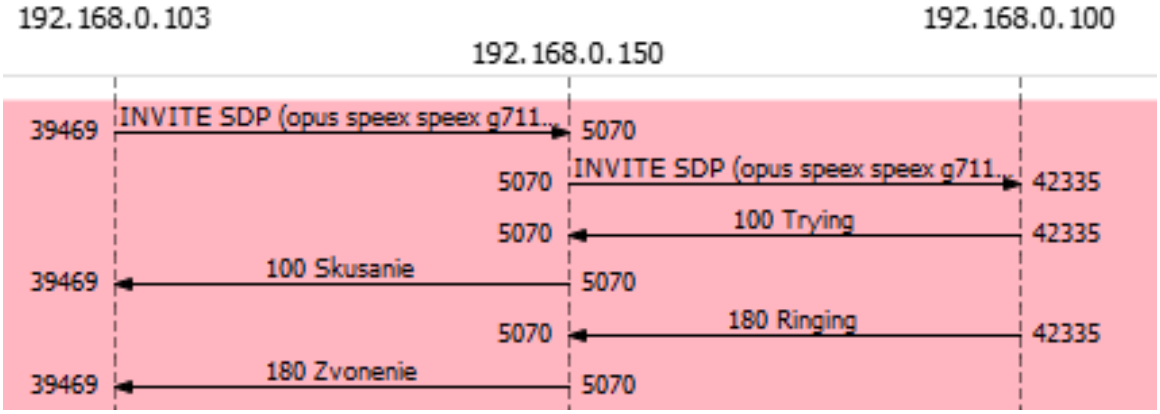
Opravené chyby v knižnici:

- Pri posielaní dát boli posielané datové typy **string** a nie **byte**
- Stará forma použitia príkazu join pre stringy
- Stará forma použitia kontroly nachádzajúceho sa kľúca v poli (z **X.has_key(Y)** na **Y in X**)

3.3 Doplnkové funkcionality

3.3.1 Úprava SIP stavových kódov

Na úpravu stavových kódov bolo potrebné pochopiť akým spôsobom sú kódy na SIP Proxy spracúvané.



Obr. 1: Okresaný flowchart hovoru medzi dvoma účasntikmi hovoru

Ak sa pozrieme na odchytené packety a následne na Call Flow poskytnutý programom Wireshark môžeme na obrázku 3.3.1 vidieť začiatok vyzvánania. Jedna strana pošle cez server druhej strane **INVITE**, ktorý následne spustí sekvenciu dvoch SIP kódov od druhej strany - **100 Trying, 180 Ringing**. Na grafe je možné vidieť, že SIP komunikácia prebieha tak, že všetky požiadavky a kódy idú cez SIP Proxy Server.

Pre to aby sme dokázali zmeniť, kódy potrebujeme upraviť knižnicu a prísť na to kde sa tieto kódy spracúvajú.

Okresaný a zjednodušený kód knižnice:

```
rx_request_uri = re.compile("^([^ ]*) sip:([^ ]*) SIP/2.0")
rx_code = re.compile("^SIP/2.0 ([^ ]*)")

class UDPHandler(socketserver.BaseRequestHandler):
        def processCode(self):
                # Kod funkcie

        def processRequest(self):
                if len(self.data) > 0:
                        request_uri = self.data[0]
                        # If strom kontroly regexov
                        if ...:

                        elif ...:

                        elif rx_code.search(request_uri):
                                self.processCode()

        def handle(self):
                self.data = data.split("\r\n")
                request_uri = self.data[0]
                if rx_request_uri.search(request_uri) or rx_code.search(request_uri):
                self.processRequest()
```

Kód knižnice je postavený na základe spracovávania všetkých požiadaviek pomocou funckie **handle()**, ktorá pomocou reg-ex výrazov zistí, či sa jedná o SIP požiadavku. Ak áno prejde sa do funckie **processRequest()**, ktorá skontroluje pomocou ďalších reg-ex výrazov o akú SIP požiadavku / správu sa jedná. Tu môžeme vidieť, že existuje rexgex výraz, ktorý kontroluje SIP stavové kódy a volá sa funckcia **processCode()**

Táto funckia spracováva všetky SIP stavové kódy. V tejto funckii bude potrebné tieot správy upravovať.

Okresaný a zjednodušený kód funkcie **processCode()**:

```
    def processCode(self):
        origin = self.getOrigin()
        if len(origin) > 0:
            if origin in registrar:
                socket, claddr = self.getSocketInfo(origin)
                self.data = self.removeRouteHeader()
                data = self.removeTopVia()

                # ############################
                # Vlastny kod na vymenu SIP stavovych kodovsprav
                for key in code_messages:
                    check_msg = "SIP/2.0 " + key
                    if data[0] == check_msg:
                        data[0] = "SIP/2.0 " + code_messages[key]
                        break
                # ############################
                text = "\r\n".join(data)
                socket.sendto(text.encode(), claddr)
```

Vlastný kód prezrie string, ktorý obsahuje SIP stavový kód spoločne s jeho správou a na základe externého súboru **check_msg.json**.

3.3.2 Denník hovorov

Denník hovor funguje na základe spracovácania požiadaviek serverom. Akonáhle je požiadavka pre server SIP požiadavka zavolá sa vlastná funkcia **call_log(data)**, ktorá spracuje požiadavku a pozrie sa na jej obsah na základe ktorej rozhodne či sa jedná o hovor.

Hlavné časti kódu spracovnia denníka hovorov:

```
rx_ringing = re.compile("^SIP/2.0 180.*")
rx_decline = re.compile("^SIP/2.0 603.*")
rx_ok = re.compile("^SIP/2.0 200.*")
rx_terminated = re.compile("^SIP/2.0 487.*")
rx_busy_here = re.compile("^SIP/2.0 486.*")
rx_bye = re.compile("^BYE")

class CallLog:
        # Trieda CallLog - Pre kazdy individualny hovor samostatny objekt
    call_id = ""
    fromm = ""
    to = ""
    start_ring = None
    start = None
    end = None
    status = ""

    def __init__(self, call_id, fromm, to, start_ring):
        self.fromm = fromm
        self.to = to
        self.start_ring = start_ring
        self.call_id = call_id

def call_log(data):
        # Funkcia, ktora spracuje poziadavku na zaklade CallID
    # Najdenie CALL-ID z prichodzich dat a vytvorneie premennej log
    if rx_ringing.search(req_uri):  # Ak niekto zacal volat a zvoni
        # Vytvorenie objektu CallLog na zaklade CallID
        elif log: # kontrola existenie dennika pre CallID
                if rx_decline.search(req_uri):  # Odmietnutie hovoru
                        # Zapisanie do suboru
                elif rx_terminated.search(req_uri):  # Zrusenie zvoenia volajucim
                        # Zapisanie do suboru
                elif rx_busy_here.search(req_uri):  # Neodpovedanie
                        # Zapisanie do suboru
                elif rx_bye.search(req_uri):  # Ukoncenie hovoru
                        # Zapisanie do suboru
                elif rx_ok.search(req_uri):  # Zaciatok telefonovania po zdvihnuti
                        # Zapisanie casu zaciatku hovoru
```

Zjendoduseny kod spracovania dennika hovoru v triede spracovania SIP požiadaviek:

```
class UDPHandler(socketserver.BaseRequestHandler):
        def processRequest(self):
                if len(self.data) > 0:
                    request_uri = self.data[0]
                    call_log(self.data)  # Dennik hovorov - Vlastna Funkcia

                        # IF-ELSE Strom pre konkretnu poziadavku na zklade Reg-EX
```

3.4 Zhrnutie funkcionality

Na testovanie funkcionality SIP Proxy servera bol použitý klient **Linphone**. SIP Proxy server dokáže registrovať účastníkov bez nutnosti autentifikácie a sprostredkovať hovor medzi dvoma účastníkmi so všetkými funkcionalitami ako vyzvánanie, zrušenie hovoru, podržanie hovoru ale aj chat. Funkčné je aj ak účastník hovor odmietne alebo ak volájúci sa rozhodne na volané číslo prestať volať ešte v momente keď hovor nebol zdvihnutý. Je možné vykonať aj konferenčný hovor, ktorý bol otestovaný medzi 3ma účastnikmi. Úspešné bolo aj vykonanie videohovoru a presmerovanie hovoru spoločne s podržaním účasntíka, ktorý tento hovor presmeroval.

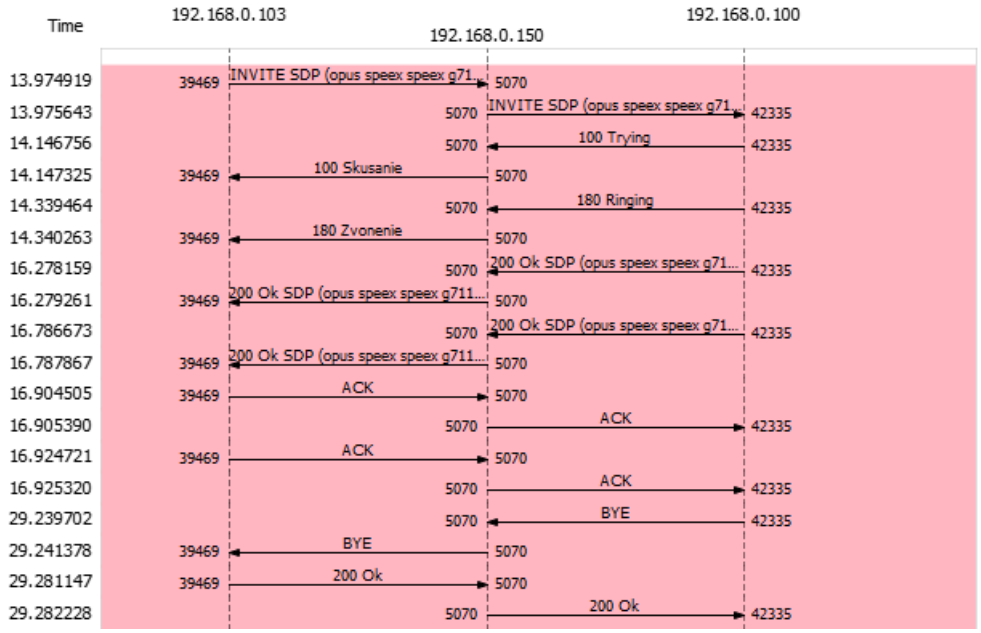
Doplnkovou funkcionalitou je aj logovanie denníka hovorov a úprava stavových SIP kódov. Logovanie zaznamenáva všetky hovory, ktoré boli zdvihnuté, zamietnuté alebo zamietnuté volajúcim. Logovanie poksytuje aj záznam konferenčného hovoru s tým, že v denníku sa takýto hovor zobrazuje ako dva rozdiele hovory medzi účastníkmi. Mimo iné denník zaznamená stav hovoru (ukončený, odmietnutý, odmietnutý volajúcim) a zároveň čas odkedy volajúci začal volať, čas zdvihnutia a čas ukončenia hovoru.

Doplnkové funkcionality ako videohovor, presmerovanie hovoru a konferenčný hovor už boli podporované použitou SIP knižnicou a klientom.

4 Opis SIP Komunikácie

4.1 Odmietnutie, zrušenie, hovor

Pozrieme sa na záznam paketov z troch scenárov medzi dvoma klientami. Jedná sa normálne vyzváňanie avšak jeden z nich je odmietnutý druhou stranou, ďalší je zrušený volajúcim a posledný je normálny hovor s klasickým ukončením.

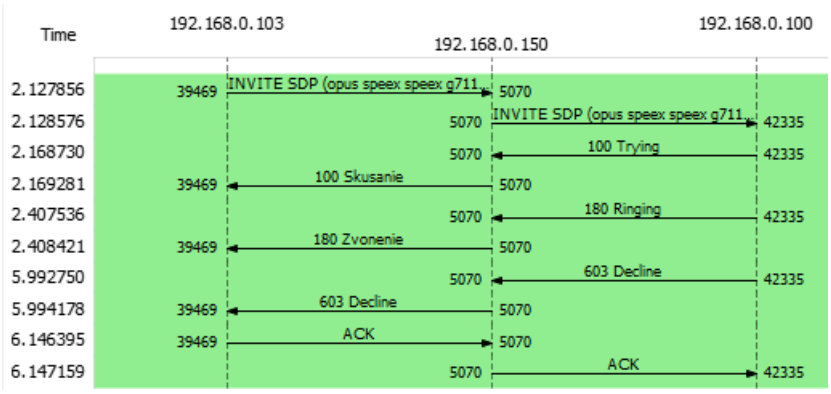


Obr. 2: Diagram hovoru medzi dvoma účastníkmi

Na tomto diagrame vidíme, že priebeh hovoru začína vždy požiadavkou **INVITE** od strany, ktorá chce hovor začať. V strede sa nachádza SIP Proxy Server, ktorý tút správu preposiela klientovi, ktorému je hovor určený. Následne z druhej strany vidíme odpoveď kódmi **100 Trying** a **180 Ringing**. Tu si môžeme všimnúť aj úspešné zmenenie znenia SIP kódov.

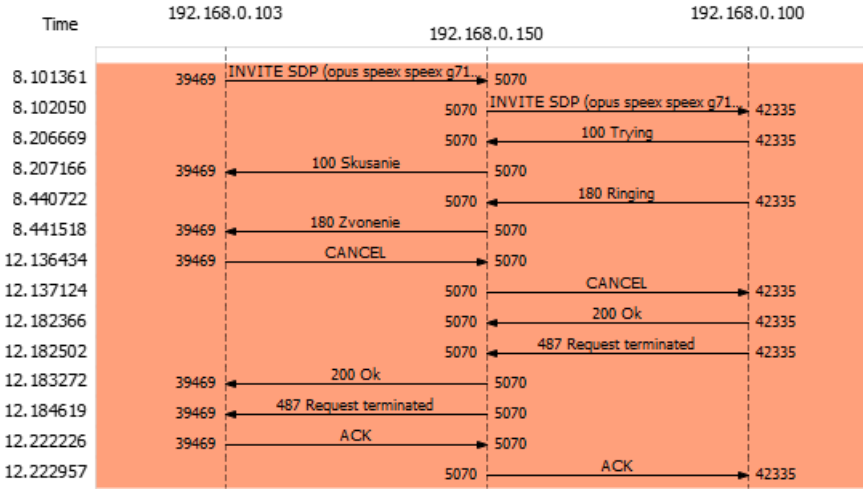
Ak to druhá strana zdvihne, pošle sa **200 OK** a následne na **200 OK** sa posiela potvrdzovacia správa **ACK**. V tomto momente sa hovor začal a medzi účastníkmi prebieha RTP / OPUS (UDP) komunikácia.

Zloženie hovoru funguje tak, že strana, ktorá chce zložiť pošle druhej strane správu **BYE**, ktorá je opätovne potvredná kódom **200 OK**. V tejto chvíli je hovor zrušený.



Obr. 3: Diagram hovoru, ktorý bol odmietnutý

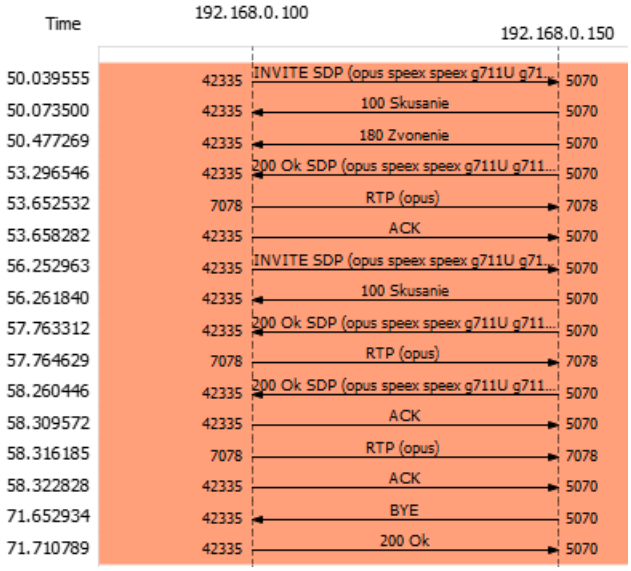
Hovor, ktorý bol zrušený funguje rovnako ako normálny zdvihnutý hovor avšak na zmenu, že miesto poslania kódu **200 OK**, čím potvrdíme zdvihnutie sa posiela **603 Decline**, ktorá hovorí volajúcemu, že hovor bol zrušený (tzv. Obsadené).



Obr. 4: Diagram hovoru, ktorý bol zrušený volajúcim

Zrušenie volajúcim funguje na podobnom princípe avšak ak volajúci zruší vytačanie druhej strane, volajúci pošle SIP požiadavku **CANCEL**, na ktorú je následne odpovedané **200 OK** ako potvrdenie a následne sa posiela **487 Request Terminated**. Volajúci ešte opútovne potvrdí správou **ACK**. Týmto bol hovor zrušený volajúcim a na druhej strane prestane vyzváňať telefón.

4.2 Videohovor



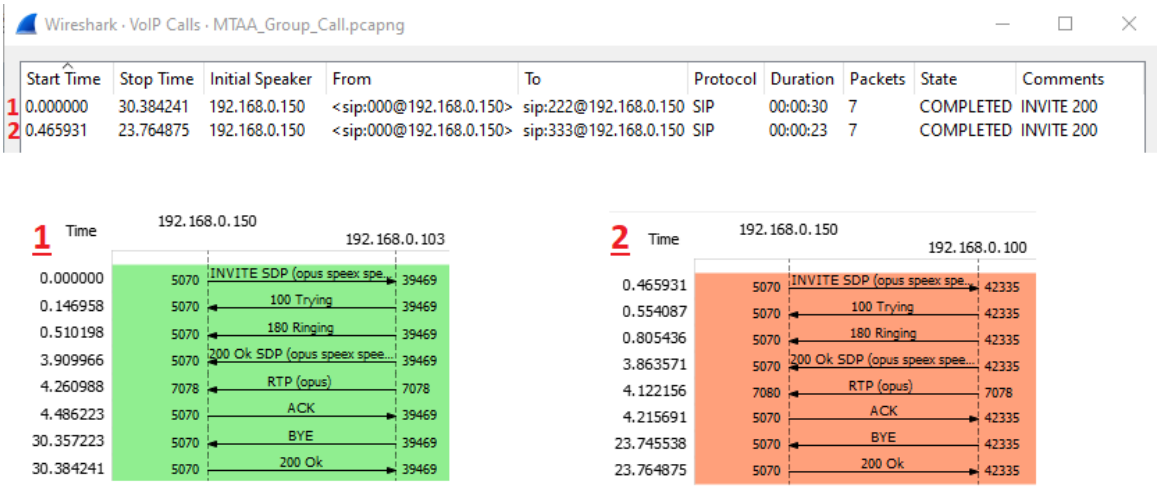
Obr. 5: Diagram hovoru a následného pozvania k videohovoru

V tomto grafe nevidíme 3 strany ako v predchádzajúcich grafoch. NA tomto grafe vidíme iba volajúceho a server. Avšak požiadavky sú iba na druhú stranu zreplikované cez server.

Videohovor funguje na rovnakom princípe ako normálny hovor. Volajúci poslal najrpv **INVITE**, hovor sa prijal a začala **RTP** hlasová komunikácia medzi používateľmi s definovanými kódekmi pre komunikáciu pomocou protokola **SDP**. Nateraz je medzi účastníkmi dohodnutý iba audio hovor, čiže aj kodeky, na ktorých sa dohodlo obsahujú iba audio kodeky.

Volajúci chcel v tomto prípade uskutočniť videohovor. Pre to sa znova poslal **INVITE** avšak už aj s video kodekmi. Audio komunikácia avšak stále prebieha, no ostatný proces je rovnaký ako keď sa počiatočne volá no bez správy **180 Ringing**. Následne po odpovedi **ACK** od volajúceho, ktorý chcel uskutočniť videohovor sa začína pomocou RTP prenášať aj video záznam s kamier klientov.

4.3 Konferenčný hovor



Obr. 6: Diagram konferenčného hovoru

Konferenčný hovor prebieha ako dva samostatné normálne hovory.

4.4 Presmerovanie, podržanie

Obr. 7: Diagram presmerovaného hovoru v rámci 3och klientov

Presmerovanie hovor funguje nasledovne. Klient, ktorý chce presmerovať spoluúčastníka hovoru na iné číslo, pošle správu **REFER**, ktorá hovorí o tom, že odkazujeme druhú stranu na iný kontakt. Druhá strana nám to akceptuje kódom **202 Accepted**. Následne budú medzi tymito klietnami prebiahť takzvané RE-INVITE, ktoré budú hovoriť o statuse v akom sa hovor nachádza. Po akceptovaní sa pošle **INVITE**, ktorý hovorí o tom, že hovor je podržaný. V prípade ak druhá strana ukončí hovor s novým kontaktom a navráti s ak nášmu hovoru pošle sa nový **INVITE**, ktorý hovorí o tom, že hovor sa navracia s podržania.

5 Záver

V tomto zadaní sme si vyskúšali spojazdniť SIP telefónnu ústredňu pomocou extenrej knižnice. Riešenie zadania je funkčné a spĺňa všetky povinné funkcionality. Riešenie bolo overené pomocou programu Wireshark.