Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Mobilné technológie a aplikácie
Zadanie 1 – SIP Proxy

Adam Bereník
Cvičenie: Streda 14:00

Adam Bereník
102863

Zadanie

Úlohou v tomto zadaní bolo sprevádzkovanie SIP Proxy (telefónnej ústredne) na realizovanie hovorov medzi SIP klientami s využitím nejakej knižnice.

Riešenie

Riešenie som implementoval v programovacom jazyku Python, verzia 3.10.2. Použil som zabudované Python knižnice, predovšetkým *socket* a *socketserver* a externú SIP knižnicu.
Odkaz na môj GitHub repozitár: https://github.com/okmada123/MTAA-Zadanie1

Funkcionality

Po spustení program umožňuje registráciu účastníka, realizáciu hlasového aj video hovoru, realizáciu konferenčného hovoru a tiež presmerovanie hovoru niektorým z jeho účastníkov na iného registrovaného používateľa.
Vytvoril som aj jednoduché logovanie hovorov a zmenil jeden zo stavových kódov SIP protokolu.

Použitá SIP knižnica

V mojom riešení používam knižnicu PySipFullProxy (https://github.com/tirfil/PySipFullProxy), konkrétne *sipfullproxy.py*. Hlavná časť tejto knižnice je trieda *UDPHandler*, ktorá poskytuje SIP Proxy funkcionalitu.

Zmeny v knižnici

Pre použitie tejto knižnice v aktuálnej verzii Pythonu, ktorú som používal (3.10.2) bolo nutné upraviť zopár drobných zmien v jej kóde.

Dictionary už nemá metódu *has_key*, ale pre overenie kľúča v slovníku sa používa syntax *in*.

```
#if registrar.has_key(fromm):
if fromm in registrar:
```

Dáta posielané cez sockety je teraz nutné kódovať na „raw bytes“ pri posielaní, a dekódovať pri prijímaní.

```
def handle(self):
    #data = self.request[0]
    data = self.request[0].decode("utf-8")
    ...
```

```
...
#text = string.join(data,"\r\n")
text = "\r\n".join(data).encode("utf-8")
socket.sendto(text , claddr)
...
```

Vytvorenie *stringu* spojením *listu* má teraz tiež trochu inú syntax.

Okrem toho knižnica zakazovala použitie lokálnych IP adries. Túto časť kódu som odstránil (je ponechaná ako komentár).

```
# rx_invalid = re.compile("^192\.168")
# rx_invalid2 = re.compile("^10\.")
...
    def processRegister(self):
        ...
        # if rx_invalid.search(contact) or rx_invalid2.search(contact):
        #     if fromm in registrar:
        #         del registrar[fromm]
        #     self.sendResponse("488 Not Acceptable Here")
        #     return
        ...
```

Po týchto zmenách bolo možné registrovať sa, uskutočniť hlasový, video a konferenčný hovor a aj presmerovať hovor.

Logovanie hovorov

Implementoval som aj vlastné logovanie hovorov a to tak, že som vytvoril funkcie na spracovanie jednotlivých signálov, ktoré idú cez SIP Proxy. Priamo v knižnici som potom na príslušné miesta pridal volanie týchto logovacích funkcií. Tieto sa nachádzajú v súbore *my_logging.py*.

Formát záznamov v logu je:

ČAS,UDALOSŤ,OD_KOHO,KOMU,INFO

pričom pole *INFO* je využité iba pri ukončení hovoru na zaznamenanie jeho dĺžky, a v prípade registrácie nie je použité pole *KOMU*.

Udalosti, ktoré sa zaznamenávajú sú:

- Registrácia
- Volanie (INVITE)
- Ukončenie hovoru pred jeho začatím (rozlišuje sa aj ktorá strana hovor odmietla)
- Zodvihnutie hovoru
- Prepínanie medzi audio a videohovorom, pauza hovoru – tieto udalosti sa v logu ďalej nerozlišujú. Sú označené ako *SWITCH INVITE*, teda ako nejaká zmena počas hovoru.
- Ukončenie hovoru s vypísaním dĺžky trvania

Moje logovanie sa ukladá do súboru *custom.log*. V SIP knižnici je tiež pôvodné logovanie, ktoré som nemenil a to sa ukladá do súboru *proxy.log*.

Úprava stavových kódov

Posledným zásahom do knižnice bolo upravenie stavových kódov. Rozhodol som sa meniť text pri kóde 200.

```
#self.sendResponse("200 0K")
self.sendResponse("200 OK THANKS")
```

V pôvodnej implementácii posielala SIP Proxy `[200 0K]` a tento kód som zmenil na `[200 OK THANKS]`.

Samotná zmena pri volaní funkcie *sendResponse* ale nie je dostatočná. Toto zmení status code pri správach, ktorými odpovedá klientovi priamo proxy (napríklad pri registrácii klienta). Správy s kódom `[200 OK]`, ktoré cez proxy iba prechádzajú (to znamená, že sú od niektorého klienta a určené pre iného klienta) by zostávali nezmenené. Toto som vyriešil vytvorením vlastnej funkcie *(replace_200_OK)* priamo v triede *UDPHandler*, ktorú volám vo funkcii *processCode* pred tým, ako sa packet prepošle cieľovému klientovi.

```
def processCode(self):
    origin = self.getOrigin()
    if len(origin) > 0:
        if origin in registrar:
            ...
            data = self.replace_200_OK(data) #replace 200 OK -> 200 OK THANKS
            text = "\r\n".join(data).encode("utf-8")
            socket.sendto(text,claddr)
```

Táto funkcia iba prehľadá dáta, ktoré sú pripravené na odoslanie, a ak v nich nájde kód `[200 OK]`, zmení ho na `[200 OK THANKS]`. Ešte je nutné upozorniť, že vo Wiresharkom zachytenej komunikácii sa stále vyskytujú aj kódy `[200 OK]`. Tieto prichádzajú zo strany klientov a nemám ich ako zmeniť. Všetky správy, ktoré idú smerom **od** SIP Proxy by mali mať pri stave 200 kód `[200 OK THANKS]`.

Spustenie programu

Program sa spúšťa spustením súboru *main.py* a funkcie *main*.

```
def main():
    ip_addr = socket.gethostbyname(socket.gethostname())
    port = 5060

    sipfullproxy.recordroute = f"Record-Route: <sip:{ip_addr}:{port};lr>"
    sipfullproxy.topvia = f"Via: SIP/2.0/UDP {ip_addr}:{port}"

    my_logging.initial_log()
    print(f"SIP proxy listening on {ip_addr}:{port}...")

    socketserver.UDPServer((ip_addr, port),
        sipfullproxy.UDPHandler).serve_forever()
```

Funkcia najprv zistí IP adresu zariadenia. Potom ju použije spolu s portom 5060 (štandardný SIP port) na nastavenie *stringov*, ktoré sa v knižnici používajú v SIP hlavičke.
Nasleduje funkcia, ktorá do logu zapíše, že bol spustený server a potom výpis do konzoly.
Nakoniec spustím *socketserver.UDPServer,* ktorý bude počúvať na adrese *IP:5060* a ako jeho *RequestHandlerClass* použijem triedu *UDPHandler* zo SIP knižnice.

Adam Bereník
102863

Testovacie scenáre

Pri testovaní funkcionalít som používal 3 rôzne zariadenia. SIP Klient bol vždy linphone (https://www.linphone.org/).

SIP Username	Lokálna IP Adresa	Operačný systém
pocophone	192.168.1.163	Android 10
desktop-linux	192.168.1.232	Kali Linux
t440s	192.168.1.146	Windows 10

SIP Proxy bežalo na 192.168.1.146:5060.

Spolu mám 4 scenáre, ku každému prikladám výpis z logu a komunikáciu zachytenú Wiresharkom (v GitHub repozitári).

Scenár 1 – hlasový hovor (log)

- Registrácia 2 účastníkov
- Hlasový hovor *(desktop-linux -> pocophone)*
    - Zrušenie volania *(z desktop-linux)*
    - Odmietnutie hovoru *(z pocophone)*
- Prijatie hovoru (začatý hlasový hovor)
- Ukončenie hovoru

Scenár 2 – video hovor (log)

- Registrácia 2 účastníkov
- Začatý hlasový hovor
- Požiadavka o prepnutie na video hovor
    - akceptovaná
- Prepnutie naspäť na hlasový hovor
- Ukončenie hovoru

Scenár 3 – konferenčný hovor (log)

- Registrácia 3 účastníkov
- Konferenčný hovor *(začatý z desktop-linux)*
    - prijatý na *t440s*
    - prijatý na *pocophone*
- Konferenčný hovor ukončený *(z desktop-linux)*

Scenár 4 – presmerovanie hovoru (log)

- Registrácia 3 účastníkov
- Začatý hlasový hovor *(desktop-linux -> pocophone)*
- Presmerovanie hovoru *(pocophone -> t440s)*
- Prijatie hovoru na *t440s*
- Zrušenie hovoru na *t440s*
- Pokračovanie hovoru medzi *desktop-linux* a *pocophone*
- Ukonečnie hovoru