# Zadanie 1 – SIP Proxy

Github repozitár: https://github.com/zrohacova/mt1sip

## Opis zadania

Na svojom počítači som sprostredkovala SIP Proxy, ktorá umožnila realizáciu hovorov medzi zariadeniami v rámci jednej LAN siete.

## Programovací jazyk a knižnice

Ako programovací jazyk som si vybrala Python. V danom jazyku som použila implementáciu[1] z internetu, ktorú som upravila pre účely zadania. Daný repozitár poskytuje niekoľko .py súborov, vrátane proxy cez TCP a proxy s autentifikáciou. Mne však stačila obyčajná proxy, čo je v tomto prípade *sipfullproxy.*py súbor. Samotná implementácia podporoval hlavný aj aj väčšinu doplnkových scenárov zo zadania.

## Zmeny v porovnaní s originálom

Základom však bolo danú implementáciu upraviť. Bola písaná v staršej verzii Pythonu, kde niektoré veci nefungovali pre Python 3. Išlo najmä o nahradenie starých metód novými, napr. *has_key* je metóda na vyhľadávanie v dictionary, ale v Python 3 sa využíva *in*, pridanie *decode* a *encode* pri posielaní cez sockety alebo odstránenie riadkov, ktorý znefunkčnil pripájanie sa v rámci vlastnej LAN siete. Program pred tým využíval knižnicu *SocketServer*, ktorej ekvivalent v novej verzii je *socketserver*. Z implementácii som potom ešte odstránila main a nahradila vlastným. Okrem toho som popridávala výpisy do konzoly, aby sa jednoduchšie dalo sledovať, že čo sa na proxy deje.

## Opis implementácie

Pre server a websockety boli požité knižnice *socketserver* a *socket*, pre účely loggovania boli použité knižnice *logging* a *time* a knižnica *re* pre regex výrazy.

Hlavná logika programu je v stiahnutom súbore *sipfullproxy.py*, kde sa o funkcionalitu stará trieda *UDPHandler*. Program zaznamená každý jeden request, ktorý má v protokole SIP a spracuje ho v *processRequest* funkcii. Na základe metódy requestu sa spustí požadovaná funkcia. Ak sa na proxy pripojí zariadenie prvý krát, príde žiadosť na registráciu, o ktorú sa postará *processRegister* funkcia. Tu sa do slovníka *registrar* zapíšu základné údaje o zariadení a v ďalších funkciách sa len kontroluje, či daný slovník obsahuje zariadenie a či je platné. Na konci sa odošle potvrdenie o registrácií. Takto sa zaregistruje každé zariadenie, ktoré sa chce na sieť pripojiť. Ak nejaké zariadenie chce začať hovor, odošle *Invite* request. Táto požiadavka sa spracuje v *processInvite*, kde sa zaznamená, že ktoré zariadenie chce akému volať. Podobne, ak sa pošle request s ACK, tento request sa spracuje v processAck, kde sa tiež zaznamená a pošle ďalej cez socket. Keď príde hocijaký iný request, čiže BYE alebo CANCEL, spustí sa *processNonInvite*, kde sa tiež zaznamená, že o čo ide. Ak prichádzajú rámce z proxy/druhého zariadenia, ktoré označujú nejaký stav (ako Trying, Ringing, Decline,...), tieto requesty sa spracujú v *processCode*, kde sa dané odpovede preposielajú cez socket.

V súbore *main.py* sa potom vyskytuje *main*¸ ktorý nastaví logovanie a spustí celú proxy. Nastavia sa tiež kanály na zaznamenávanie komunikácie.

[1] https://github.com/tirfil/PySipFullProxy

## Spustenie a práca s programom

Program sa spustí v súbore *main.py*. Tu len stačí spustiť funkciu *main*, ktorá spustí aj samotnú proxy. Ja som program spúšťala v IDE Pycharm, kde nebolo treba nič ďalšie riešiť.

Ďalej treba len klientov na pripojenie. Ja som využila Linphone klientov, kde som si zadala meno a ako SIP server IP adresu. Prenos som nastavila cez UDP, klient sa pekne pripojil na server a mohla začať prebiehať konverzácia. Klientov som mala nainštalovaných na telefóne a tablete, ktorým aj v súboroch prislúcha daný názov, a proxy bola spustená na počítači. Ak som potrebovala počítač ako zariadenie, tam som spustila Zoiper klient, kde som podobne všetko nastavila s názvom *sue* alebo *pc*.

## Hlavný scenár

Za hlavný scenár považujem základnú požadovanú funkcionalitu zo zadania, čiže zaregistrovanie účastníka, vytočenie a zvonenie, prijatie/odmietnutie hovoru druhou stranou a ukončenie prijatého hovoru. Toto všetko program od začiatku bez problémov podporoval. Pre tieto základné požiadavky stačí len program spustiť a uskutočniť hovory.

Registrácia zariadení je zaznamenaná v *registration.pcap*.

Uskutočnenie hovoru, teda vytočenie, zvonenie, prijatie druhou stranou, uskutočnenie hovoru a následné ukončenie je demonštrovaný *main.pcap*. Tu je pekne vidno, ako jedna strana(mobil) sa snaží nadviazať hovor s druhou stranou(tablet). Druhá strana odosiela odpoveď, teda Trying, Ringing a následne Ok. Hovor je nadviazaný a prebieha. Ku koncu strana, ktorá chce zložiť posiela BYE request na čo druhá strana odpovedá Ok.

Situácia, kde zariadenie volá druhú stranu, ktorá hovor odmietne je v *declined.pcap*. Zavolanie druhému zariadeniu a následné zrušenie hovoru bez toho, aby bol hovor prijatý, je zaznamenané v *terminated_call.pcap.* Situácia, že je jedno zariadenie je nedostupné, teda chcem uskutočniť hovor so zariadením, ktoré nie je prihlásené, je v *unavailable.pcap*.

## Doplnkové funkcionality

Ako som už spomenula, samotná implementácia poskytovala funkcionalitu aj pre doplnkové požiadavky. Pri scenároch som už trace robila na sieti, kde už boli predom registrované zariadenia.

### Logovanie

Samotný program poskytoval aj logovanie, ja som však toto logovanie zmenila pre potreby zadania. Teda zaloguje sa len, že kto chce komu volať, kto hovor komu prijal/neprijal a kto hovor ukončil. Každý log obsahuje aj id hovoru ako prišlo v rámci, aby sa dalo lepšie orientovať v tom.

Pri spustení sa vytvorí súbor *proxy.log*, ktorý sa uloží do rovnakého priečinku. V súbore sa potom zaznamenajú spomenuté údaje spolu s časom danej akcie a zaznamená sa tam aj adresa zdroju a destinácie. Do existujúceho súboru sa nové údaje zapisujú na koniec.

Log vyzerá napríklad takto:

```
19:05:51:INFO:(Call-ID: DyUME3dusX): mobil@192.168.2.192 chce zacat hovor s tab@192.168.2.192
19:05:53:INFO:(Call-ID: DyUME3dusX): mobil@192.168.2.192 zacal hovor s tab@192.168.2.192
19:05:55:INFO:(Call-ID: DyUME3dusX): tab@192.168.2.192 ukoncil hovor s mobil@192.168.2.192
```

### Upravenie stavových kódov

V samotnom programe sa posielali rôzne odpovede s kódmi. Tieto kódy som všetky upravila, väčšinou len preložila do slovenčiny (teda 200 OK sa zmenilo na 200 Dobre). To je vidno veľmi dobre aj v súbore *unavailable.pcap* alebo i v *forwarding.pcap* , kde sú už kódy upravené.

### Uskutočnenie video hovoru

Video hovor prebiehal tak, že si klienti zavolali ako normálny hovor. Následne som v jednom zariadení vybrala zapnúť video, kedy sa odoslala žiadosť o zapnutie videa druhej strane. Na druhom zariadení som túto žiadosť potvrdila a video hovor mohol začať. Tento scenár je zaznamenaný v *video.pcap*.

### Presmerovanie hovoru

V tomto scenári som si zavolala z jedného zariadenia(mobil) na druhé(tablet). Počas hovoru som vybrala na mobile možnosť pre presmerovanie (v Linphone som usúdila, že je to ikonka telefónneho slúchadla so šípkou) a zadala adresu počítača. Následne prebiehal hovor medzi tabletom a počítačom. Následne bol hovor ukončený najprv na tablete, čo ukončilo presmerovaný hovor a potom na mobile. Tento scenár je v *forwarding.pcap*.

### Konferenčný hovor

Pri konferenčnom hovore som najprv uskutočnila hovor medzi telefónom a počítačom. Následne som na mobile pridala do hovoru aj tablet. V Linphone som potom vybrala možnosť konferenčného hovoru a všetky tri zariadenia mohli v daný moment medzi sebou komunikovať. Ukončenie hovoru na mobile ukončilo hovor na všetkých zariadeniach. Toto je zdokumentované v *conference_call.pcap*.