26/2/2022

SIP Proxy

MTAA ZAD Č. 1

Samuel Hetteš, ID: 110968

STU FIIT 2020/2021

# ZADANIE

### Hlavná myšlienka zadania:

Na vašom počítači (alebo virtuálnom počítači) sprevádzkujte SIP Proxy, ktorá umožní prepájanie a realizáciu hovorov medzi štandardnými SIP klientami.

### Doplňujúce informácie k zadaniu:

Na implementáciu vašej 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. To znamená, že **nemôžete** zobrať existujúcu SIP Proxy ako napr. Asterisk, kde len skompilujete alebo priamo spustíte cudziu binárku… Hovor **musí** byť realizovaný medzi dvomi **fyzickými** zariadeniami v rámci LAN siete.

### Rozsah povinných funkcionalít:

* Registrácia účastníka (bez nutnosti autentifikácie)
* Vytočenie hovoru a zvonenie na druhej strane
* Prijatie hovoru druhou stranou, fungujúci hlasový hovor
* Ukončenie hlasového hovoru (prijatého aj neprijatého)

Ak sú splnené **všetky** tieto podmienky, študent získava 5 bodov, ktoré sú minimom na absolvovanie tohoto zadania.

### Doplnkové funkcionality (ktoré môžete, ale nemusíte urobiť):

* Možnosť zrealizovať konferenčný hovor (aspoň 3 účastníci)
* Možnosť presmerovať hovor
* Možnosť realizovať videohovor
* Logovanie “denníka hovorov” – kto kedy komu volal, kedy bol ktorý hovor prijatý, kedy bol ktorý hovor ukončený, do ľubovoľného textového súboru v ľubovoľnom formáte
* Úprava SIP stavových kódov z zdrojovom kóde proxy, napr. “486 Busy Here” zmeníte na “486 Obsadené”

Každá doplnková funkcionalita predstavuje plus 1 bod.

**Počas prezentácie zadania musíte byť schopní na zariadení, kde beží ústredňa urobiť SIP trace a otvoriť ho pomocou tcpdump alebo Wireshark, a v primeranom rozsahu vysvetliť cvičiacemu, ako daná signalizácia prebieha.**

### Forma odovzdania:

* Dokumentácia, kde opíšete, ako ste vase riešenie implementovali, aké knižnice ste použili, a ako ste ústredňu sprevádzkovali, vo formáte PDF do miesta odovzdania v AIS. V PDF dokumente musí byť odkaz na váš repozitár v Gite (verejný rezpoitár, ľubovoľná služba)
* Vo vašom repozitári sa musí nachádzať PCAP trace z každého scenára, ktorý vaša implementácia poskytuje.

# IMPLEMENTÁCIA

Na implementáciu SIP proxy som sa rozhodol pre programovací jazyk **Python**, konkrétnejšie verziu 3.9 a použil nasledujúcu knižnicu: [**https://github.com/tirfil/PySipFullProxy**](https://github.com/tirfil/PySipFullProxy)

Môj vlastný main súbor, spolu s touto knižnicou sa nacháza na nasledujúcom odkaze: [**https://github.com/Kesuera/FIIT\_MTAA/ZAD1**](https://github.com/Kesuera/FIIT_MTAA/ZAD1)

Samotná knižnica je troška staršia a preto ju bolo potrebné poupraviť, aby fungovala správne, vykonal som tieto **úpravy**:

* Importy
* Kódovanie a dekódovanie prichádzajúcich a odchádzajúcich správ
* Odstránenie kontroly, ktorá neumožňovala registrovanie privátných IP adries

Po týchto úpravách sa mi podarilo knižnicu rozbehať.

Knižnica má vlastnú main funkciu, odkiaľ je spúšťaná. Tú som odstránil a pre **počiatočné nastavenia**, ktoré v nej boli som v knižnici vytvoril funkciu **start\_server()**. Následne som v mojom vlastnom main súbore importoval knižnicu a odtiaľ spustil túto funkciu – **owrapovanie**.

Vo funkcii start\_server sa najprv nastaví **topvia a recordroute,** ďalej sa vytvorí **socket server** a všetky prichádzajúce requesty **handluje** classa UDPHadnler, kde sa nachádza hlavná logika knižnice.

Hlavné metódy classy UDPHandler:

* **handle() –** handluje prichádzajúce requesty, dekóduje ich a volá funkciu processRequest
* **processRequest() –** určí typ requestu, na základe čoho ďalej zavolá už konkrétnu funkciu pre spracovanie requestu
* **processRegister() –** spracovanie registrácie účastníka, ktorý sa napája na sip proxy
* **processInvite() –** spracováva všetky prichádzajúce invity
* **processAck() –** spracováva prichádzajúce ACK správy
* **processNonInvite() –** spracováva správy typu bye (ukončenie hovoru), cancel, info, options, prack, update, refer, message
* **processCode() –** spracováva requesty obsahujúce SIP status kódy

Okrem týchto sú tu ešte ďalšie napríklad pre vrátenie príjemcu, odosielateľa, socket info, poslanie odpovede, kontroly platnosti adresy, odstránenie a pridanie topvia, odstránenie routovacej hlavičky alebo zmenu request uri.

Samotná knižnica mala už priamo **implementovanú** **väčšinu** **doplnkových** **funkcionalít**.

**Upravil** **som** akurát **stavové** **kódy** a taktiež pridal **výpis** **IP** **adresy** a **portu**, na ktorom proxy beží a **odstránil** pôvodné **logovanie**, ktoré som **upravil** podľa potrieb tohto zadania. Na jeho implementáciu som využil knižnicu **logging**. Loguje sa vo **formáte**: dátum a čas – id hovoru – informácia. Nasledujúci príklad demonštruje výstup logu po prebehnutí hovoru medzi 2 účastníkmi:

Obrázok, na ktorom je text

Automaticky generovaný popis

Pre **uľahčenie** **logovania** som vytvoril **triedu** **Call**, ktorá si udržiava informáciu o ID hovoru a statuse. Globálne som zadefinoval pole, do ktorého sa vkladajú prebiehajúce hovory a následne podľa ich ID sa vyhľadávajú. V samotnej classe UDPHandler som vytvoril funkciu **findCall()**, ktorá vráti hovor pre dané ID a **4 funkcie pre logovanie informácií** o hovore.

# splnené funkcionality

* Registrácia účastníka (bez nutnosti autentifikácie) ✅
* Vytočenie hovoru a zvonenie na druhej strane ✅
* Prijatie hovoru druhou stranou, fungujúci hlasový hovor ✅
* Ukončenie hlasového hovoru (prijatého aj neprijatého) ✅
* Možnosť zrealizovať konferenčný hovor (aspoň 3 účastníci) ✅
* Možnosť presmerovať hovor ✅
* Možnosť realizovať videohovor ✅
* Logovanie “denníka hovorov” – kto kedy komu volal, kedy bol ktorý hovor prijatý, kedy bol ktorý hovor ukončený, do ľubovoľného textového súboru v ľubovoľnom formáte ✅
* Úprava SIP stavových kódov z zdrojovom kóde proxy, napr. “486 Busy Here” zmeníte na “486 Obsadené” ✅