

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Telekomunikacja
SPECJALNOŚĆ: Sieci teleinformatyczne

PRACA DYPLOMOWA
INŻYNIERSKA

Projekt i uruchomienie edukacyjnego programu
komputerowego prezentującego zasady
funkcjonowania komunikacji w środowisku SIP.

Project and implementation of educational software
application for presentation of communication rules
in SIP environment.

AUTOR:
Jakub Rosa

PROWADZĄCY PRACĘ:
dr inż. Janusz Klink, KTT

OCENA PRACY:

Spis treści

1.	Wstęp	5
2.	Cel i zakres pracy	6
3.	Wykorzystywane technologie	6
3.1.	JAIN SIP API	7
3.2.	Kotlin	8
3.3.	JavaFx	9
3.4.	TornadoFx	11
3.5.	Voice over Internet Protocol.....	12
4.	Protokoły sygnalizacyjne w sieciach multimedialnych.....	13
4.1.	Środowisko Skype	13
4.2.	WebRTC	14
4.3.	Środowisko H.323	16
4.4.	Środowisko SIP	16
5.	Istniejące programy prezentujące komunikację w środowisku SIP	18
5.1.	SIP Inspector.....	18
5.2.	StarTrinity SIP Tester™	19
5.3.	SIPp	20
6.	Projekt stanowiska laboratoryjnego i programu prezentującego komunikację w środowisku SIP	21
6.1.	Stanowisko laboratoryjne	21
6.2.	Program	22
6.2.1.	Założenia programu	22
6.2.2.	Wymagania i uruchomienie programu	22
6.2.2.1.	Instalacja oprogramowania Java.....	22
6.2.2.2.	Instalacja oprogramowania Maven	23
6.2.3.	Obsługa sygnalizacji SIP	24
6.2.4.	MVC	26

6.2.5.	Interfejs graficzny aplikacji	27
6.2.5.1.	Ekran powitalny	28
6.2.5.2.	Strefa nauki	29
6.2.5.3.	Strefa Symulacji.....	30
6.2.5.4.	Słownik Pojęć	32
6.2.6.	Scenariusze w „Strefie Nauki”	32
6.2.6.1.	Poprawna nowa rejestracja	33
6.2.6.2.	Zaktualizowanie pól kontaktów	34
6.2.6.3.	Aktualna lista kontaktów	35
6.2.6.4.	Wyrejestrowanie klienta	35
6.2.6.5.	Poprawne ustanowienie sesji	36
6.2.6.6.	Połączenie pomiędzy serwerami proxy	38
6.2.6.7.	Wielopoziomowa autoryzacja.....	40
6.2.7.	Internacjonalizacja aplikacji	43
6.3.	Przyszłość programu.....	43
7.	Wnioski.....	44
8.	Bibliografia.....	46

*Pracę niniejszą dedykuję rodzicom **Juliannie i Robertowi Rosie**, którzy mnie wspierali w każdym etapie mojego życia, oraz mojej dziewczynie **Agnieszce Pilch**, która była dla mnie oparciem podczas tworzenia niniejszej pracy inżynierskiej.*

1. Wstęp

W obecnych czasach coraz częściej odchodzi się od wykorzystywania odrębnych sieci telekomunikacyjnych dla różnych usług. Spowodowane jest to tym, że usługi często dostarczane są za pomocą różnych aplikacji. Dane transportuje się za pomocą protokołu IP. Warstwa usługowa jest dzięki temu niezależna od warstwy transportowej, dlatego też usługi mogą być dostarczane za pomocą jednej sieci.

Aktualnie największą siecią na świecie jest Internet. W celu realizacji połączeń głosowych za pomocą tej sieci należało przygotować protokół, który pozwoliłby na inicjację sesji pomiędzy użytkownikami. Aktualnie najpopularniejszymi protokołami wykorzystywanymi w połączeniach głosowych są protokoły SIP, H.323 i Skype. Same w sobie nie potrafią transmitować mowy, dlatego razem z nimi używa się protokołów transmisji danych w czasie rzeczywistym.

Protokół inicjalizacji sesji (SIP) został opracowany w celu tworzenia, utrzymania i kończenia sesji pomiędzy różnymi terminalami użytkowników. Pozwala na ustalenie lokalizacji i dostępności użytkowników, oraz udostępnia informacje o możliwości oprogramowania lub urządzenia wykorzystywanego przez użytkownika. Bez tych podstawowych zadań komunikacja w tak dużej sieci, jaką jest Internet, byłaby niemożliwa. Aktualnie ten protokół jest używany w różnych metodach komunikacji. Dzięki coraz łatwiejszemu dostępowi do szybkiego Internetu zyskuje sobie popularność. Można to łatwo zauważyć, ponieważ coraz więcej rozwiązań telefonicznych opartych o protokół SIP jest oferowane na rynku polskim [1].

W niniejszej pracy skupiono się na dwóch zagadnieniach. Pierwszym z nich jest tworzenie programu edukacyjnego w języku programowania Kotlin, który ma oferować narzędzia do nauki środowiska SIP. Język Kotlin został wydany w 2011 roku i w przeciągu ostatnich lat zyskał sobie bardzo dużą popularność przy tworzeniu oprogramowań serwerowych i aplikacji na platformę Android. Kod w tym języku jest bardziej zwięzły, co ułatwia skupienie się na logice biznesowej danej aplikacji. Jedną z jego największych zalet jest pełna integracja z kodem pisanym w języku Java, dlatego w łatwy sposób można integrować stare projekty z nowym językiem programowania.

Drugim zagadnieniem w pracy jest przygotowanie programu edukacyjnego który w łatwy i przyjemny sposób pokaże zasadę działania protokołów komunikacyjnych w środowisku SIP. Aktualnie istnieje bardzo mało programów, które pokazują zasadę działania protokołu SIP, a jeżeli już są, to są bardzo uciążliwe w użytkowaniu dla osoby, która dopiero

poznaje środowisko SIP. Program używany podczas laboratorium „Urządzenia i systemy multimedialne”, czyli „SIP Inspector” w nowszych wersjach posiada płatną licencję, co uniemożliwia korzystanie z niego przez studenta podczas nauki poza uczelnią. Ponadto wersja oprogramowania używana podczas ćwiczeń w laboratorium jest przestarzała, mało intuicyjna i łatwo w niej popełnić błąd podczas tworzenia zapytań do serwera.

W pracy podjęto się opracowania programu, który można uruchomić na istniejącej infrastrukturze w laboratorium. Program ma za zadanie przedstawić w prosty i intuicyjny sposób zasady działania w środowisku SIP.

2. Cel i zakres pracy

Celem pracy jest przygotowanie laboratoryjnego narzędzia edukacyjnego z zakresu protokołów komunikacyjnych wykorzystywanych w systemach multimedialnych. W głównej mierze skupie się na komunikacji w środowisku SIP. Za pomocą nowoczesnych technologii informatycznych powstanie aplikacja przyjazna użytkownikowi, bazująca na stylu „material design”, stworzonym przez Google. Dzięki swojej prostocie styl ten jest przyjazny dla użytkownika oraz posiada żywe i dobrze dobrane kolory [2].

Program będzie posiadał trzy części: edukacyjną, symulacyjną i informacyjną. W części edukacyjnej użytkownikowi zostaną przedstawione podstawowe scenariusze przepływów połączeń. Za pomocą prostej animacji i opisu, użytkownik będzie mógł obserwować poszczególne fazy przepływu połączenia, a także zobaczyć jego opis i przykładowe dane sygnalizacji SIP. Druga część programu, odpowiedzialna za symulację, będzie udostępniała możliwość tworzenia pakietów sygnalizacji SIP i ich wysłania. Użytkownik będzie tworzył zapytania za pomocą formularza. Pola do wypełnienia i zaznaczenia będą posiadały ikony pomocy, dzięki którym będzie można w prosty sposób podejrzeć, co należy wpisać lub zaznaczyć w danym polu. Ostatnia część będzie posiadała słownik terminów i pojęć związanych ze środowiskiem SIP. W opisach pojęć zostaną zawarte tylko podstawowe informacje z odnośnikiem do odpowiedniej normy RFC. Odnośnik zostanie zawarty w celu uzupełnienia wiedzy.

3. Wykorzystywane technologie

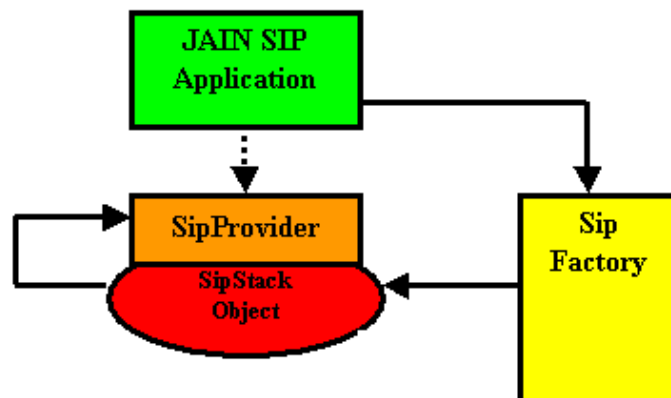
W programie zostaną wykorzystane nowoczesne technologie informatyczne, używane aktualnie w celu tworzenia komercyjnego oprogramowania. Interfejs użytkownika zostanie stworzony w oparciu o styl „material design”, powszechnie używany w wielu aplikacjach i stronach internetowych.

3.1. JAIN SIP API

Biblioteka JAIN¹ SIP API jest ustandaryzowanym interfejsem SIP, używanym w języku programowania JAVA. Zapewnia interfejs do przesyłania informacji między klientami SIP a serwerami SIP. Jest to najstarsza biblioteka, która zostanie użyta do tworzenia programu. Wspiera protokół SIP według normy RFC3261. Posiada trzy główne moduły:

- SipEvent – wiadomości SIP są enkapsulowane jako obiekty Javy i przekazywane pomiędzy SipProvider a SipListener,
- SipProvider – zapewnia aplikacji dostęp do zaimplementowanego interfejsu protokołu SIP,
- SipListener – przetwarza przychodzące wiadomości według zaimplementowanej logiki

W celu zainicjalizowania biblioteki w aplikacji należy stworzyć obiekt SipFactory. Następnie z jego pomocą stworzyć obiekt SipStack, który będzie odpowiedzialny za wysyłanie i odbieranie informacji. Należy stworzyć obiekt SipProvider, aby móc ustawić w programie port, na którym użytkownik chce nasłuchiwać wiadomości przychodzące i to, skąd będzie wysyłać [3].

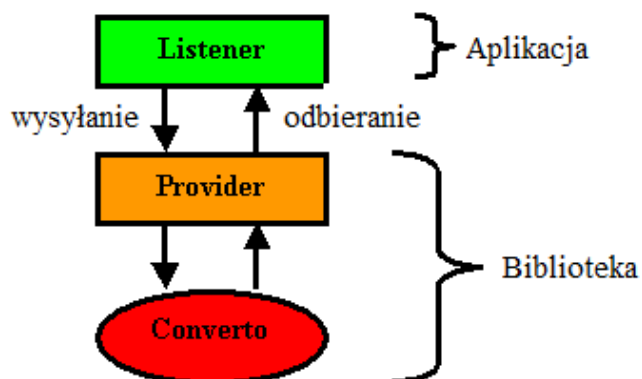


Rysunek 1. Inicjalizacja biblioteki JAIN SIP API [3]

W celu wysłania i odebrania wiadomości biblioteka udostępnia słuchacza (z ang. listener). Wszystkie wiadomości z aplikacji do obiektu SipProvider przechodzą przez słuchacza. Jest to miejsce, w którym użytkownik może zaimplementować swoją logikę odnośnie obsługi wiadomości. Oczywiście przed wysłaniem wiadomości należy zbudować

¹ JAIN – skrót od Java API's for the Integrated Networks

odpowiedni obiekt, który zostanie przekazany do słuchacza. Także wiadomości przychodzące będą w formie jakiegoś obiektu, który użytkownik musi odpowiednio przetworzyć.



Rysunek 2. Trasa wiadomości przy użyciu biblioteki JAIN SIP API [3]

3.2. Kotlin

Język programowania „Kotlin” został stworzony przez firmę JetBrains, która jest odpowiedzialna za produkcję oprogramowania dla programistów. Programiści z tej firmy postanowili stworzyć nowy język programowania, ponieważ większość języków, które można było uruchomić na wirtualnej maszynie Javy, nie miała cech, których potrzebowali. Wyjątkiem był Scala². jednak ten język programowania nie kompilował się wystarczająco szybko. Pierwszy raz Kotlin został zaprezentowany w 2011 roku podczas konferencji „Language Summit” w Santa Clara w Kalifornii, a w 2012 roku firma otworzyła kod projektu na licencji „Apache 2.0”. Dzięki nowej licencji można było zacząć używać języka Kotlin zarówno w oprogramowaniach typu „Open Source”, a także w programach z zamkniętym kodem. Podczas uwolnienia kodu³ firma wydała także dodatki do swoich programów pozwalające na programowanie w języku Kotlin, które są rozwijane równolegle razem z językiem [4]. W 2017 roku Google wsparło Kotlin jako oficjalny język programowania aplikacji na Androida, co jeszcze bardziej się przyczyniło do jego wzrostu popularności.

Kotlin jest to statycznie typowany język programowania podobny do Scala, Gosu⁴ i Ceylon⁵. Można go skompilować zarówno do kodu wykonywalnego na wirtualnej maszynie Javy, a także do kodu w JavaScript. Język ten jest zwięzły i pragmatyczny. Naprawia wiele

² Scala – język programowania łączący cechy języków funkcyjnych i obiektowych

³ Uwolnienie kodu – przekształcenie zamkniętego oprogramowania w oprogramowanie z otwartym kodem źródłowym, publiczne udostępnienie kodu źródłowego aplikacji

⁴ Gosu – zorientowany obiektowo język programowania ogólnego przeznaczenia

⁵ Ceylon – statycznie typowany język programowania dla maszyn wirtualnych Javy i JavaScriptu

błędów i niedociągnięć, z którymi zmagają się programiści Javy podczas procesu tworzenia oprogramowania. Główną zaletą Kotlina jest bardzo dobra współpraca z kodem Java oraz współpraca ze wszystkimi bibliotekami i platformami Java. Można go stosować niemal wszędzie tam, gdzie używana jest Java, np. do tworzenia aplikacji serwerowych lub do aplikacji dla systemu Android.

Kotlin wykorzystuje statyczne domniemanie typów danych. Oznacza to, że każdy typ zmiennych jest znany na etapie kompilacji kodu i nie trzeba jawnie określać typu zmiennych. Na podstawie kontekstu typ jest określany automatycznie. Ponadto kompilator sprawdza, czy metody i pola z obiektów, których używamy na pewno istnieją. Wykorzystywana metoda typowania danych poprawia wydajność pracy kodu, czyli podczas działania program nie musi wyszukiwać funkcji do których ma się odwołać. Łatwiejsza jest także praca z nieznanym kodem, ponieważ wiadomo jakiego rodzaju obiektu i pola może spodziewać się programista. Kod jest mniejszy, przez to bardziej przejrzysty dla programisty i łatwiejszy w utrzymaniu.

Bardzo dużo pojęć i terminów w języku programowania Kotlin jest wziętych z Javy, jednak jedną z największych różnic jest możliwość wykorzystania programowania funkcyjnego. Pozwala to na napisanie bardziej zwięzłego, krótszego i eleganckiego kodu. Funkcje można zapisać za pomocą zmiennych i podawać je jako argumenty do innych funkcji. Powoła to na uniknięcie duplikowania kodu. W ten sposób lepiej wykorzystuje się siłę abstrakcji. Kod jest uogólniony i pozwala na ograniczenie zmian w kodzie tylko do kluczowych elementów dla danego problemu.

Programowanie funkcyjne pozwala na ograniczenie problemów z modyfikowaniem tych samych danych przez różne wątki w programach wielowątkowych. Poprzez używanie czystych funkcji i niezmiennych struktur danych nie trzeba stosować specjalnych technik synchronizacyjnych w kodzie, które tylko dodatkowo by obciążły nasz program. Ponadto posiada zabezpieczenie „Null Safety”. Pozwala ono na wykrycie zmiennych posiadających wartość „null” już na etapie kompilacji. Dzięki temu w programie występuje o wiele mniej błędów związanych z niezadeklarowaniem wartości pola i programy działają stabilniej [5].

3.3. JavaFx

JavaFx to biblioteka, która umożliwia tworzenie aplikacji GUI⁶ w języku programowania Java. Pozwala na wysoką wydajność podczas pracy z dużymi zestawami danych. Biblioteka ta została stworzona w 2005 roku przez Chrisa Olivera. W 2014 roku wraz

⁶ GUI – z ang. graphical user interface – graficzny interfejs użytkownika

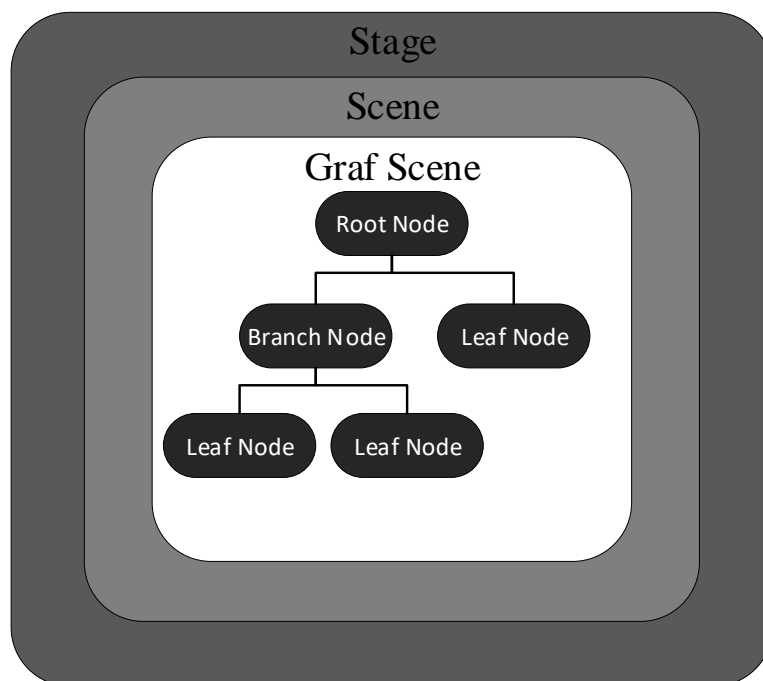
z wydaniem Javy w wersji 8 JavaFx w wersji 8 stała się integralną częścią tego oprogramowania. Aplikacje napisane przy użyciu tej technologii mogą wydajnie pracować na wielu platformach i różnych urządzeniach. Chociaż nie posiadają tak wielkiej popularności jak aplikacje oparte o HTML5, nadal wiele firm korzysta z tej biblioteki w przypadku używania „wewnętrznych” aplikacji biznesowych. Biblioteka zawiera w sobie szereg funkcji, które ułatwiają tworzenie graficznego interfejsu dla programu, tj. przygotowane kontrolki⁷, obiekty 2D i 3D, interfejs do komunikacji z użytkownikiem, itp. Pozwala na łączenie animacji graficznej i czynności wykonywanych przez użytkownika. Jest także kompatybilna z poprzednikiem, czyli z Swing, co ułatwiało zmianę jednej technologii na drugą [6].

JavaFx wykorzystuje GPU⁸ poprzez używanie wydajnego przyspieszenia sprzętowego grafiki „Prism”. W celu renderowania grafiki w systemach Windows jest używany DirectX, a w systemach Mac i Linux OpenGL. W przypadku, gdy system nie ma karty graficznej Prism, następuje renderowanie programowe grafiki za pomocą Java2D. W celu tworzenia okien wykorzystywany jest rodzimy system operacyjny, na którym jest uruchomiona dana aplikacja. W przeciwieństwie do poprzednio używanego AWT JavaFx wykorzystuje mechanizm kolejki zdarzeń aktualnie używanego systemu operacyjnego, w celu planowania swoich zdarzeń.

Biblioteka JavaFx jest dostępna dla języków programowania, które mogą być wykonane na wirtualnej maszynie Javy, tj. Java, Groovy, JRuby i Kotlin. Posiada bardzo intuicyjną aplikację do budowania interfejsu graficznego, tj. „Scene Builder”. Użytkownik metodą „przeciągnij i upuść” jest w stanie w łatwy i prosty sposób stworzyć aplikację graficzną. Dostępna jest także bardzo duża baza predefiniowanych kontrolek interfejsu użytkownika, a każdą z nich można dostosować za pomocą podstawowej znajomości CSS. Biblioteka posiada duże wsparcie dla multimediów.

⁷ Kontrolka – z ang. controls – element wchodzący w interakcje z użytkownikiem, np. przycisk, pole do wpisania tekstu

⁸ GPU – z ang. Graphical Processing Unit – procesor graficzny



Rysunek 3. Diagram struktury aplikacji JavaFx [7]

Aplikacja z użyciem biblioteki JavaFx posiada trzy główne komponenty: stage, scene i nodes. Stage zawiera wszystkie obiekty GUI. Podczas uruchomienia aplikacji jest tworzony podstawowy stage, na którym są umieszczane pozostałe elementy graficzne JavaFx. Posiada dwa główne parametry: wysokość i szerokość. To one odpowiadają za wielkość okna po uruchomieniu aplikacji. Scene przedstawia kontekst aplikacji JavaFx. Zawiera wszystkie nodes reprezentujące zawartość graficzną i kontrolki. Mogą zawierać obiekty geometryczne przedstawione w 2D lub 3D, przyciski, pola wyboru, kontenery obiektów, elementy multimedialne [8].

3.4. TornadoFx

Język programowania Kotlin nie posiada bezpośredniego i łatwego wsparcia dla JavaFx, dlatego powstała biblioteka pomocnicza TornadoFx. Biblioteka ta posiada bogate i funkcjonalne interfejsy użytkownika dostosowane do wyżej wymienionego języka programowania. Pozwala na łatwe utrzymywanie i ewoluowanie kodu w aplikacji.

Kotlin pracuje na wyższym poziomie abstrakcji i zapewnia praktyczne funkcje językowe niedostępne w Javie, dlatego w 2016 roku powstało rozszerzenie TornadoFx, które stara się znacznie zminimalizować ilość kodu potrzebnego do zbudowania aplikacji JavaFx. Autorem tego modułu jest Edwin Syse. Pozwala w pełni wykorzystać możliwość języka Kotlin poprzez wprowadzenie bezpiecznych konstruktorów i wstrzykiwania zależności. Umożliwia

to szybkie tworzenie kontrolek i interfejsów użytkownika. TornadoFx doskonale przedstawia jak Kotlin może uprościć pisanie kodu. Sprawia, że kod jest elegancki i bardziej czytelny w porównaniu do standardowego oprogramowania JavaFx [9].

3.5. Voice over Internet Protocol

Ze względu na wykorzystywanie techniki VoIP na zajęciach w laboratorium, zostanie przedstawiona technika VoIP, która używa środowiska SIP do komunikacji.

Technika VoIP rozwinęła się z sieci analogowych. W tej sieci mowa jest przesyłana przez synchroniczne kanały cyfrowe. Pozwala to na realizację usług izochronicznych (z ang. isochronous services). Usługi izochroniczne charakteryzują się możliwością przesyłania dużej ilości danych w równych odstępach czasu. W usługach tego typu nie jest gwarantowane bezbłędne przesyłanie danych oraz nie występuje retransmisja błędnych informacji [10]. Przesyłane są dane asynchroniczne (mowa, obraz) przez synchroniczny kanał telekomunikacyjny. Pierwszy raz system telefonii IP został uruchomiony w 1996 roku przez izraelską firmę VocalTec.

Wykorzystanie sieci pakietowych w telefonii obecnie jest bardzo ekonomicznym rozwiązaniem. Nie ma określonego typu danych, dlatego można przysyłać dźwięk dowolnie zakodowany, a także nie ma potrzeby budowania dodatkowej infrastruktury pod tą usługę. Poza tym sieć pakietowa ma niskie koszty użytkowania. Według raportu o stanie rynku telekomunikacyjnego w Polsce w 2017 r. przygotowanym przez UKE w ostatnich latach, czas połączeń wykonywanych za pomocą telefonii VoIP spada. W roku 2016 wielkość wolumenu ruchu w sieci zmniejszyła się średnio o 18%, jednak pomimo tego spadku ilość klientów rośnie i nadal operatorzy rynku telekomunikacyjnego chętnie korzystają z techniki VoIP [11].

W celu przesłania dźwięku mowy, w technice VoIP wykorzystuje się protokół IP. Wywodzi się on z wojskowej sieci ARPANET stosowanej w USA w latach 1969-72. Podczas przesłania mowy w pierwszej kolejności należy doprowadzić sygnał analogowy do postaci cyfrowej, a następnie ten sygnał skompresować i podzielić na pakiety. Pakiety są to usystematyzowane zbiory danych. Po tych czynnościach zakodowane cyfrowo próbki przesyłane są do drugiego użytkownika. Dzieląc sygnał na pakiety w taki sposób może wystąpić problem z zapewnieniem odpowiedniej jakości obsługi transmisji głosu. Jakość transmisji jest zależna od aktualnego obciążenia sieci, dostępnej przepustowości oraz od opóźnień. Technika VoIP sama w sobie nie zapewnia żadnych mechanizmów kontroli jakości połączenia. Problemem jest także stosowany mechanizm w protokole IP „best effort”. Pakiety są wysyłane o możliwie jak najwyższej szybkości, ale bez gwarancji jakości usługi.

Nigdy nie wiadomo w jakiej kolejności przyjdą pakiety do odbiorcy oraz czy dotrą do miejsca docelowego. W celu zwiększania jakości usług stosuje się inne protokoły znajdujące się w wyższych warstwach modelu ISO/OSI (np. protokół UDP) [12].

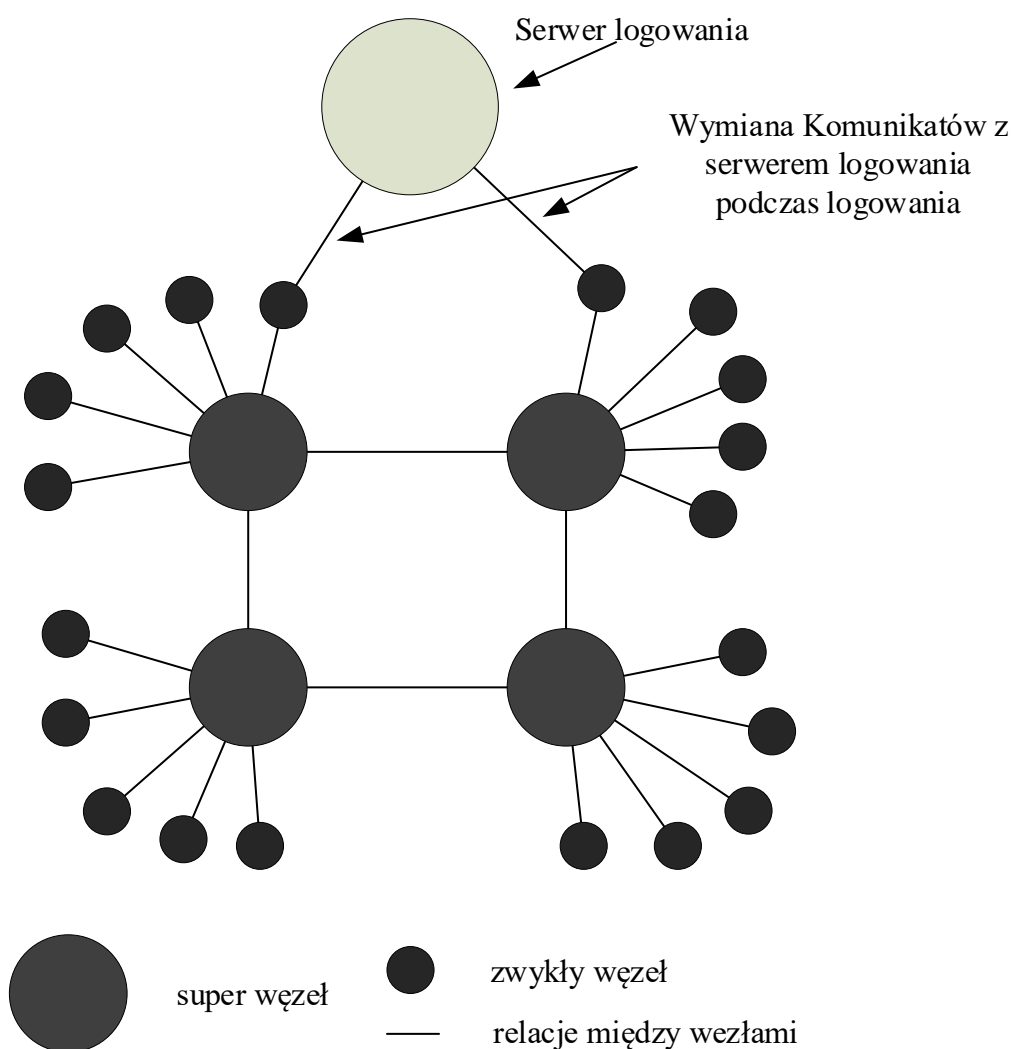
4. Protokoły sygnalizacyjne w sieciach multimedialnych

Wraz z rozwojem telekomunikacji powstało bardzo dużo rozwiązań pozwalających przesyłać dane w czasie rzeczywistym. Wykorzystanie sieci pakietowej spowodowało powstawanie opóźnień i uniemożliwiało kontrolę sprawdzenia kolejności pakietów przychodzących. Przed użyciem protokołu IP tworzone fizyczną ścieżkę pomiędzy nadawcą i odbiorcą. Obecnie pakiet, aby dotrzeć od nadawcy do odbiorcy, musi przebyć rozległą sieć Internetową i poprawnie zostać odtworzone po stronie odbiorcy.

4.1. Środowisko Skype

Skype jest to pierwszy klient VoIP oparty o technologię peer-to-peer. Został opracowany przez Szweda Niklas Zennström i Duńczyka Janus Friis w 2003 roku [13]. Bezproblemowo może pracować za NAT-em i firewallem. Szyfruje połączenia pomiędzy klientami a także przechowuje informacje o użytkownikach w kilku miejscach. Obsługuje rozmowy, wiadomości, wideo, konferencje, udostępnianie ekranu, wiadomości błyskawiczne a także przesyłanie plików.

W wykorzystywanej architekturze występują dwa typy węzłów – zwykłe i superwęzły (SN). Zwykłe węzły są to użytkownicy końcowi z uruchomioną aplikacją Skype. Super węzeł jest to punkt końcowy zwykłego użytkownika posiadający publiczny adres IP. Każdy zwykły węzeł może zostać SN, o ile posiada wystarczająco dużo zasobów komputera i ma szybkie łącze internetowe. Użytkownik, żeby mógł się zalogować, musi się połączyć do SN i zarejestrować się na serwerze logowania Skype. Serwer ten nie jest węzłem Skype, jednak jest bardzo ważnym elementem sieci.



Rysunek 4. Schemat sieci Skype [14]

Jedną z najważniejszych funkcji w sieci Skype jest możliwość logowania. Podczas tego etapu klient Skype uwierzytelnia się na serwerze logowania, informuje innym użytkownikom, że jest aktywny, określa typ NAT i dostaje informację o superwęzłach. Protokół TCP jest używany w celu inicjalizacji połączeń, zaś dane medialne wysyłane są za pomocą UDP. W przypadku, gdy protokół UDP jest blokowany przez zaporę ogniową wykorzystuje się protokół TCP [14].

Skype używa własnego protokołu do komunikacji [15], dlatego bardzo trudno jest go badać. Jedynymi sposobami, aby dowiedzieć się w jaki sposób ten protokół działa, jest inżynieria odwrotna.

4.2. WebRTC

WebRTC jest to darmowy projekt z otwartym kodem źródłowym. Jest wspierany między innymi przez Google, Mozilla i Opera. Zapewnia on przeglądarkom i aplikacjom mobilnym funkcje komunikacji w czasie rzeczywistym za pośrednictwem prostych interfejsów

API⁹. Pozwala to na komunikację za pośrednictwem wspólnego zestawu protokołów pomiędzy przeglądarkami, urządzeniami mobilnymi i urządzeniami IoT. Interfejsy te pozwalają na tworzenie aplikacji, które w czasie rzeczywistym będą potrafiły przesyłać dźwięk i obraz [16].

Interfejs API WebRTC zbudowano na bardzo wielu protokołach, m.in.: Interactive Connectivity Establishment (ICE), Session Traversal Utilities for NAT (STUN), Traversal Using Relays around NAT (TURN), SIP, itd.

ICE jest wykorzystywany, aby znaleźć sposób, w jaki dwa komputery mogą komunikować się ze sobą bezpośrednio, tak jak to jest w sieciach typu peer-to-peer. Inne protokoły są używane do próby połączenia, jeżeli nie uda się znaleźć takiego sposobu, na przykład przez zapory ogniowe i technologię NAT.

STUN to protokół służący do odkrywania publicznego adresu IP i określenia wszelkich ograniczeń w routerze, które uniemożliwiają bezpośrednie połączenie z docelowym użytkownikiem. Klient po wysłaniu żądania do serwera STUN, otrzymuje informacje pod jakim publicznym adresem IP jest widziany z Internetu i ten adres jest wysyłany do docelowego użytkownika.

W przypadku, gdy na routerze jest stosowane ograniczenie w postaci symetrycznego NATu¹⁰ używa się protokołu TURN. W celu ominięcia tego zabezpieczenia, otwiera się połączenie z serwerem TURN i przekazuje się wszystkie informacje za pośrednictwem tego serwera. Inni użytkownicy muszą wysyłać informacje na ten serwer. Takie przekazywanie informacji wiąże się z pewnym dodatkowym obciążeniem i jest używane, gdy nie ma innych alternatyw.

Negocjowanie, inicjalizowanie, zarządzanie i kończenie sesji, odbywa się za pomocą protokołu sygnalizacji lub protokołu komunikacji. Jednym z takich protokołów jest protokół SIP. Wykorzystywanie protokoły sygnalizacji lub komunikacji muszą działać z protokołem warstwy aplikacji o nazwie Session Description Protocol (SDP). Wszystkie metadane dotyczące multimediów są przekazywane za pomocą protokołu SDP [17].

⁹ API – z ang. Application Programming Interface – interfejs programowania aplikacji, zestaw reguł opisujących w jaki sposób programy komunikują się między sobą.

¹⁰ Symetryczny NAT – translacja NAT zmienia port źródłowy zapytania

4.3. Środowisko H.323

H.323 jest jednym z najstarszych standardów powszechnej stosowanym w telefonii VoIP i wideokonferencjach. Jest to system różnych protokołów i elementów, dzięki którym jest możliwy przesył dźwięku i obrazu w czasie rzeczywistym. Powstał w 1996 roku i umożliwił bezproblemowe łączenie się urządzeń produkowanych przez różnych dostawców. Aktualna wersja normy dla H.323 to wersja 7.

Standard H.323 jest oparty na czterech komponentach: terminale, bramy, gatekeeper i jednostki kontrolne MCU (Multipoint Control Unit). Terminale to urządzenia pozwalające przysyłać dane multimedialne. Muszą zawsze obsługiwać dane audio, opcjonalnie mogą obsługiwać obraz i dane. Bramy służą do łączenia terminali z różnych sieci, np. pomiędzy H.323 i ISDN. Gatekeeper jest centralnym punktem sieci H.323. Jest odpowiedzialny za adresowanie połączeń, zarządzanie przepustowością i potwierdzanie tożsamości terminali. Serwer MCU służy do łączenia więcej niż dwóch terminali podczas jednej sesji. Wszystkie terminale uczestniczące w konferencji najpierw muszą połączyć się z MCU, następnie MCU wysyła strumień audio i wideo do innych terminali. Serwer MCU często pełni rolę MCU, bramy i gatekeepera.

Transfer danych multimedialnych w środowisku H.323 wykorzystuje następujące protokoły do wymiany danych:

- TCP
 - H.225 – nawiązanie połączenia między terminalami H.323,
 - H.245 – wymiana informacji o możliwości terminala,
- UDP
 - RAS – odpowiada za rejestrację, połączenia i statusy terminali między terminalami, bramami i gatekeeperem,
 - RTP – transmisja mediów w czasie rzeczywistym,

W celu zakończenia połączenia terminale wysyłają wiadomość do gatekeepera. Następnie połączenie zostaje zakończone [18].

4.4. Środowisko SIP

Protokół SIP (Session Initiation Protocol) został opracowany przez Internet Engineering Task Force. Służy on do ustanawiania, konfigurowania i kończenia połączenia pomiędzy dwoma lub wieloma klientami. Swoim działaniem bazuje na dwóch innych protokołach internetowych: HTTP i SMTP. Jest to protokół typu żądanie-odpowiedź, czyli po wysłaniu

wiadomości czeka na odpowiedź ze strony serwera. Wyróżnia się niewielkim narzutem danych w porównaniu do swojego rozbudowanego poprzednika, czyli H.323. To właśnie dzięki swojej prostocie zyskał tak dużą popularność. Protokół SIP nie jest pełnym systemem komunikacyjnym. W celu obsługi połączeń głosowych w czasie rzeczywistym używa się innych protokołów IETF, czyli Real Time Transport Protocol (RTP) i Real Time Control Protocol (RTCP).

SIP jest protokołem, który dostarcza usługi użytkownikowi końcowemu, czyli jest to protokół warstwy aplikacji. Realizuje on usługę sesji. Sesję można postrzegać jako uporządkowaną wymianę danych pomiędzy użytkownikami, którzy uczestniczą w rozmowie. Protokół SIP musi realizować pięć aspektów komunikacji przedstawione w dokumencie RFC 3261 [1] [19], aby móc w pełni realizować tę usługę:

a) lokalizacja użytkownika

Wymagana jest umiejętność zamiany nazwy użytkownika na docelowy adres IP, z którym ma zostać nawiązana sesja. Użytkownik docelowy może korzystać z różnych adresów IP. Przy rejestracji zapisywany jest aktualny adres IP danego użytkownika, dzięki czemu inni użytkownicy są w stanie się z nim skomunikować.

b) dostępność użytkownika

Użytkownik posiada możliwość ustanowienia, czy można z nim skomunikować. Może ustawić, czy jest zajęty, nieobecny czy też dostępny. Posiada także możliwość ustawienia, że jest dostępny tylko na niektóre rodzaje połączeń.

c) funkcjonalność użytkownika

Określenie możliwości użytkowników - polega na ustaleniu jakie funkcje są dostępne w ich klientach, a następnie negocjowanie parametrów używanych podczas sesji. Jest to spowodowane ich multiplatformowością oraz funkcjonalnością klientów SIP.

d) konfiguracja sesji

Polega na ustanowieniu sesji pomiędzy dwoma użytkownikami. Użytkownik docelowy dostanie w swoim kliencie informację, że ktoś chce nawiązać z nim połączenie i dostanie możliwość odrzucenia lub zaakceptowania takiego zaproszenia. Parametry sesji zostaną ustalone, jeżeli zaakceptuje to.

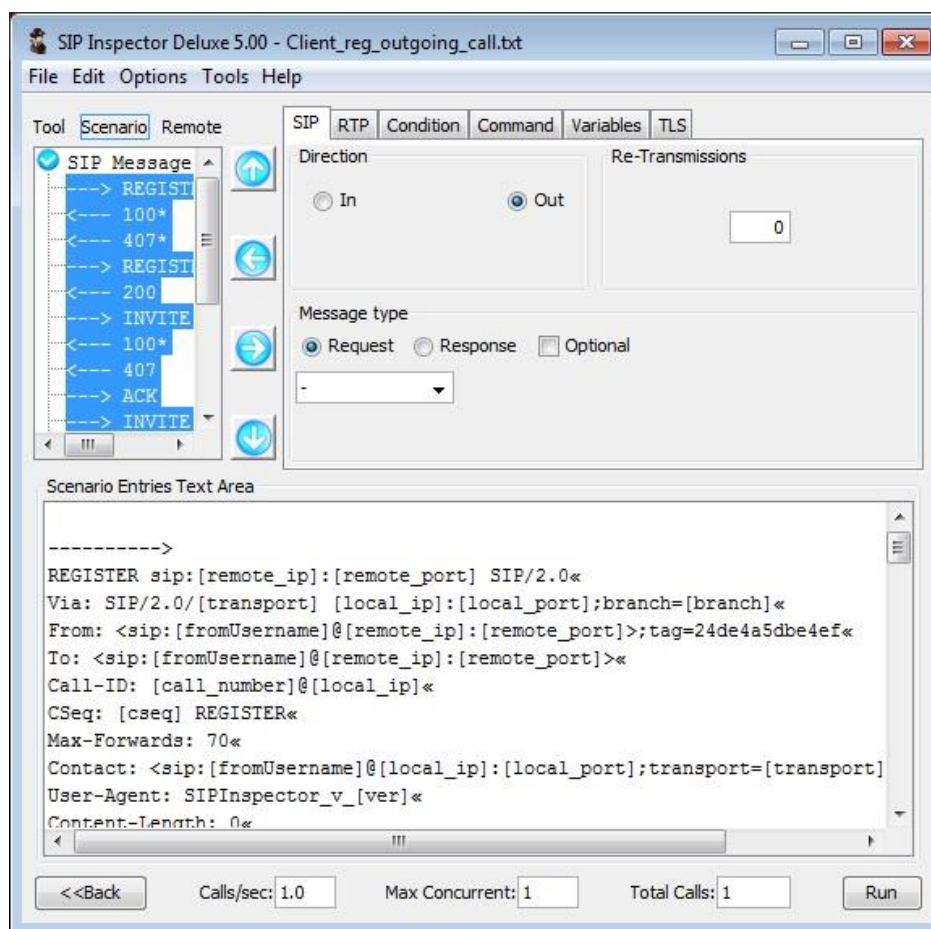
e) zarządzanie sesją

Służy do modyfikowania sesji w trakcie jej użytkowania. Można zmienić m.in. rodzaj mediów, dodaniu osób do sesji lub przekazaniu połączenia.

5. Istniejące programy prezentujące komunikację w środowisku SIP

5.1. SIP Inspector

Pierwsze wydanie programu SIP Inspector pojawiło się w 2008 roku. Został on stworzony przez kanadyjskiego programistę Zarko Coklin. Program może być wykorzystany do szczegółowej nauki protokołu sygnalizacyjnego SIP, a także do wykonywania testów obciążeniowych i automatycznych. Pozwala na zlokalizowanie problemów z sygnalizacją SIP, odtworzenie trudnych scenariuszy sygnalizacyjnych, emulowanie serwera SIP bądź klienta. Aplikacja ma także możliwość tworzenia scenariuszy w oparciu o sygnalizację SIP i protokół RTP. Narzędzie to jest stale rozwijane i udoskonalane dzięki społeczności, która się zebrała wokół tego programu [20]. Poza interfejsem graficznym program posiada także możliwość uruchomienia go za pomocą konsoli.



Rysunek 5. Przykładowy zrzut ekranu z aplikacji SIP Inspector [20]

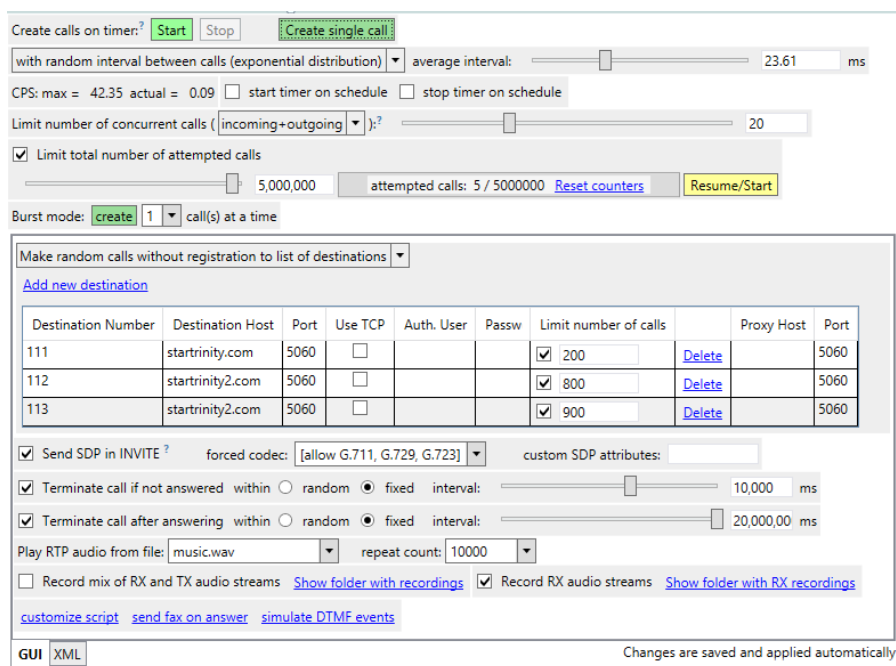
Ten program w wersji darmowej jest używany przez studentów w pracowni sieci telekomunikacyjnych. Ma on jedną podstawową wadę – poprawność formatu wpisywanego scenariusza nie jest weryfikowana. Bardzo łatwo popełnić błąd podczas edytowania scenariusza w polu tekstowym. Program sam w sobie posiada wiele scenariuszy użycia, jednak często bez zaawansowanej wiedzy nie można ich uruchomić. Oprogramowanie nie jest odpowiednie dla osób, które dopiero uczą się sygnalizacji SIP. Jest przeznaczony dla bardziej zaawansowanych użytkowników.

5.2. StarTrinity SIP Tester™

Program jest tworzony przez firmę deweloperską StarTrinity. Firma ta zajmuje się tworzeniem i rozwijaniem oprogramowania przeznaczonego dla telefonii VoIP. W swojej ofercie posiadają bardzo dużo programów i usług przeznaczonych dla VoIP.

SIP Tester jest narzędziem do testowania obciążenia VoIP. Umożliwia monitorowanie sieci, oprogramowania SIP lub sprzętu. Jest w stanie symulować i pasywnie monitorować sygnalizację SIP. Analizując media RTP jest w stanie określić jakość połączenia i budować raporty w czasie rzeczywistym. Skrypty pozwalają na symulowanie awarii komunikacji SIP.

Oprogramowanie działa na dowolnym komputerze z systemem Windows bez potrzeby dokupowania specjalnego sprzętu. Symuluje serwer aplikacji, serwer multimediiów, telefon SIP lub serwer rejestru [21].



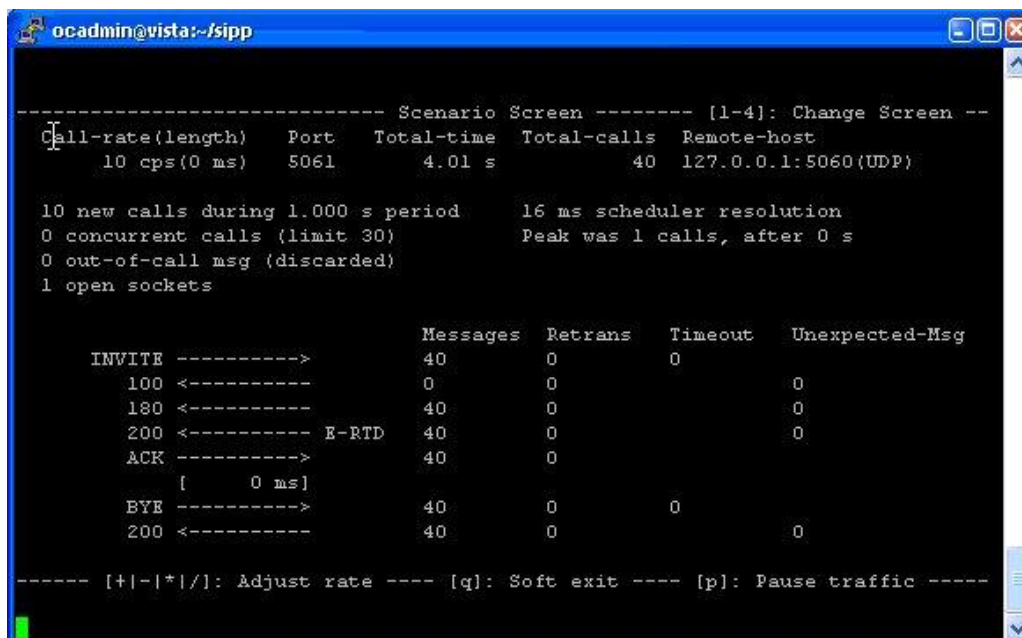
Rysunek 6. Przykładowe okno do generowania połączeń wychodzących w programie SIP Tester [21]

Program, pomimo bardzo dużej możliwości konfigurowania za pomocą GUI oraz możliwości konfigurowania symulacji za pomocą pliku XML¹¹, jest zbyt skomplikowanym programem do nauki sygnalizacji SIP. Wymagana jest wiedza na poziomie średniozaawansowanym odnośnie sygnalizacji, aby obsługiwać ten program jako początkujący użytkownik. Pola, które można edytować są prawie nieopisane, a jeżeli nawet, to w bardzo znikomej części. Program ten jest przeznaczony bardziej do wykonywania testów wydajnościowych i penetracyjnych.

5.3. SIPp

SIPp jest darmowym narzędziem testowym do generowania ruchu dla protokołu SIP. Program został stworzony przez Roberta Day. Zawiera w sobie kilka podstawowych scenariuszy połączeń, które można rozbudować za plików XML. Pozwala w czasie rzeczywistym wyświetlać statystyki dotyczące uruchomionych testów. SIPp może także służyć do testowania rzeczywistych urządzeń SIP, takich jak serwery proxy, serwery mediów, czy też bramki. Przy pomocy programu można emulować tysiące użytkowników, którzy wywołują serwer rejestracji SIP. Dodatkową zaletą aplikacji jest bardzo obszerna dokumentacja [22].

¹¹ XML – uniwersalny język znaczników dzięki którym można reprezentować dane w strukturalny sposób



```
ocadmin@vista:~/sipp
----- Scenario Screen ----- [1-4]: Change Screen --
Call-rate(length)  Port  Total-time  Total-calls  Remote-host
      10 cps(0 ms)  5061      4.01 s      40  127.0.0.1:5060(UDP)

10 new calls during 1.000 s period      16 ms scheduler resolution
0 concurrent calls (limit 30)           Peak was 1 calls, after 0 s
0 out-of-call msg (discarded)
1 open sockets

      Messages  Retrans  Timeout  Unexpected-Msg
INVITE ----->      40      0      0
      100 <-----      0      0      0
      180 <-----      40      0      0
      200 <----- E-RTD  40      0      0
      ACK ----->      40      0
      [      0 ms]
      BYE ----->      40      0      0
      200 <-----      40      0      0

----- [+|-|*|/|]: Adjust rate ---- [q]: Soft exit ---- [p]: Pause traffic -----
```

Rysunek 7. Przykładowy zrzut ekranu z uruchomionej aplikacji SIPp [22]

Program jest aplikacją konsolową, dlatego jego użytkowanie nie jest intuicyjne. Nie spodoba się użytkownikowi, który chce poznać podstawy sygnalizacji SIP. Istnieje graficzny program do edycji scenariuszy dla programu (SIP Test Studio), jednak główny program nadal jest uruchamiany w konsoli systemu Windows.

6. Projekt stanowiska laboratoryjnego i programu prezentującego komunikację w środowisku SIP

W tym rozdziale postaram się przybliżyć aspekty związane z tworzeniem oprogramowania edukacyjnego do nauki środowiska SIP. Omówię stanowisko laboratoryjne oraz etapy projektu i tworzenia aplikacji. Przedstawię zasady działania bardziej interesujących rozwiązań oraz krótką instrukcję obsługi programu.

6.1. Stanowisko laboratoryjne

W celu zniwelowania jakichkolwiek dodatkowych kosztów, program prezentujący komunikację w środowisku SIP będzie działał na aktualnej infrastrukturze w Pracowni Sieci Telekomunikacyjnych. Aktualne stanowiska są wyposażone w komputery PC, na których zostanie uruchomiona projektowana aplikacja oraz telefon sprzętowy. Komputery są połączone za pomocą infrastruktury sieciowej. Posiadają przypisane adresy IP i nazwy, które znajdują się na górnej części obudowy. Internet do komputerów PC jest doprowadzony za pomocą telefonu sprzętowego. Telefon jest ustawiony w tryb mostka sieciowego.

6.2. Program

6.2.1. Założenia programu

Tworzony w ramach pracy program ma spełniać następujące funkcjonalności:

- przedstawienie wiedzy odnośnie sygnalizacji SIP,
- przedstawienie prostych przepływów połączeń,
- możliwość definiowania własnych zapytań w sygnalizacji SIP,
- interfejs graficzny oparty o styl Material Design,
- łatwość użytkowania programu

Aktualnie istniejące programy, pozwalające tworzyć zapytania w sygnalizacji SIP nie są stworzone do nauki. Mają za zadanie wykonywać testy automatyczne i obciążeniowe. Dlatego posiadają zbyt dużo funkcji i możliwości. Osobie, która dopiero zaczyna poznawać sygnalizację SIP, należy przedstawić zagadnienia w jak najprostszym sposobie. Ponadto aplikacje przyjemne w użytkowaniu, które pozwalają definiować przepływ połączeń w sygnalizacji SIP posiadają płatną licencję oraz nie są dostępne dla studenta. Tworzony program ma na celu umożliwić naukę sygnalizacji SIP także poza czasem zajęć. Przykładowo można byłoby użyć darmowego serwera SIP i z jego pomocą testować przepływ komunikacji. Przykładowym serwerem, który umożliwiłby uzyskanie darmowego numeru SIP na jakimś serwerze jest strona onsip.com.

6.2.2. Wymagania i uruchomienie programu

Program do poprawnego działania wymaga, aby na uruchamianym systemie operacyjnym było zainstalowane oprogramowanie Java w wersji 8. Wymagane jest także, aby jednostka, na której będzie uruchamiany program, posiadała dostęp do Internetu. W celu uruchomienia aplikacji, można uruchomić plik .jar lub .exe wygenerowany z projektu lub skompilować kod źródłowy aplikacji.

6.2.2.1. Instalacja oprogramowania Java

W celu uruchomienia lub zbudowania aplikacji, należy na komputerze mieć zainstalowane środowisko Java. W tym celu należy z oficjalnej strony Java pobrać oprogramowanie w wersji 8 i zapisać plik na komputerze. Po pobraniu pliku należy uruchomić instalator i postępować zgodnie z instrukcjami na ekranie. Przed instalacją pokażą się jeszcze warunki licencji instalowanego oprogramowania. Dobrą praktyką jest przeczytanie warunków licencji i następnie należy zaakceptować jej warunki, jeżeli nie mamy zastrzeżeń. Podczas instalacji istnieje możliwość zainstalowania programów partnerów firmy Oracle, która jest

odpowiedzialna za oprogramowanie Java. Po poprawnej instalacji środowiska, pojawi się komunikat informujący o poprawnym zainstalowaniu [23].



Rysunek 8. Komunikat informujący o poprawnym zakończeniu instalacji oprogramowania Java [23]

6.2.2.2. Instalacja oprogramowania Maven

W przypadku, gdy trzeba skompilować kod źródłowy, możemy do tego użyć narzędzia Maven. Jest to narzędzie programistyczne, dzięki któremu można zarządzać i budować projekt napisany w języku programowania Java lub Kotlin. Projekt można zbudować za pomocą jednej komendy w konsoli. Dzięki Maven, w łatwy sposób możemy zarządzać zależnościami w aplikacji. Poprzez wbudowane wtyczki¹² możemy wykonywać w prosty sposób skomplikowane operacje na kodzie lub wrzucić naszą aplikację na serwer produkcyjny.

Pierwszym krokiem w instalacji jest pobranie spakowanego archiwum z tym oprogramowaniem z oficjalnej strony Maven. Wymagane jest, aby na komputerze było zainstalowane środowisko Java, żeby narzędzie działało. W celu finalizacji instalacji oprogramowania należy dodać do zmiennej systemowej PATH ścieżkę do folderu z narzędziem Maven oraz utworzyć zmienną środowiskową JAVA_HOME, która będzie wskazywać na folder, w którym znajduje się oprogramowanie Java. W celu weryfikacji poprawności instalacji należy uruchomić okno konsoli i wywołać komendę „mvn -version”. Powinna nam się pokazać aktualnie zainstalowana wersja narzędzia Maven [24].

W celu utworzenia wykonywalnego pliku z rozszerzeniem .jar z programem, należy w głównym folderze kodu źródłowego uruchomić polecenie „mvn clean compile package”.

¹² Wtyczka – dodatkowy moduł rozszerzający podstawowe możliwości programu komputerowego

Polecenie to w pierwszej kolejności wyczyści projekt, czyli usunie wszystkie pliki niezwiązane z kodem źródłowym aplikacji. Następnie skompiluje aplikację, czyli utworzy skompilowane pliki Javy. Ostatnia część polecenia służy do wygenerowania uruchamialnego pliku .jar wraz z wszystkimi niezbędnymi zależnościami. Po wykonaniu tego polecenia w folderze target znajdzie się plik o nazwie „sipprogram-0.0.1-SNAPSHOT-jar-with-dependencies.jar”, dzięki któremu będzie można uruchomić aplikację. Wystarczy dwukrotnie kliknąć przyciskiem myszy na ten plik.

```
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ sipprogram ---
[INFO] Building jar: C:\Users\ByJacob\Projects\Kotlin\sipprogram\target\sipprogram-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- maven-assembly-plugin:2.6:single (make-assembly) @ sipprogram ---
[INFO] Building jar: C:\Users\ByJacob\Projects\Kotlin\sipprogram\target\sipprogram-0.0.1-SNAPSHOT-jar-with-dependencies.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 02:01 min
[INFO] Finished at: 2018-12-12T06:40:14+01:00
[INFO] Final Memory: 89M/609M
[INFO] -----
Process finished with exit code 0
```

Rysunek 9. Komunikat poprawnej kompilacji programu i stworzenia wykonywalnego pliku .jar

6.2.3. Obsługa sygnalizacji SIP

Jak już wcześniej wspomniano, w celu obsługi ruchu wychodzącego i przychodzącego w środowisku SIP jest użyta biblioteka JAIN SIP API. To ona jest odpowiedzialna za tworzenie i przetwarzanie wszystkich zapytań. W celu zarządzania całym stosem protokołów w środowisku SIP tworzona jest instancja klasy singleton¹³ SipFactory. Pozwala ona uzyskać dostęp do funkcji niedostępnych bezpośrednio dla programisty oraz stworzyć instancje fabryk¹⁴ do tworzenia nagłówków, wiadomości i adresów. Bezpośrednia implementacja protokołu SIP nie jest dostępna do użycia, ponieważ gdy tworzono tę bibliotekę, sygnalizacja SIP bardzo szybko się rozwijała i chcąc zachować kompatybilność, wyprowadzono uniwersalne interfejsy dla podstawowych funkcjonalności. W niektórych miejscach w programie ta zasada jest łamana i obiekty są rzutowane¹⁵ na klasy z wewnętrznej implementacji sygnalizacji SIP. Ma to na celu uzyskanie dodatkowych informacji, które nie są oferowane przez interfejs. Najwięcej zabiegów tego typu zostało użytych podczas konwersji przychodzącej odpowiedzi z sygnalizacji SIP na odpowiednie obiekty zrozumiane dla programu.

¹³ Singleton – klasa jest tworzona jeden raz podczas życia programu, można uzyskać dostęp do tej klasy z każdego miejsca w programie.

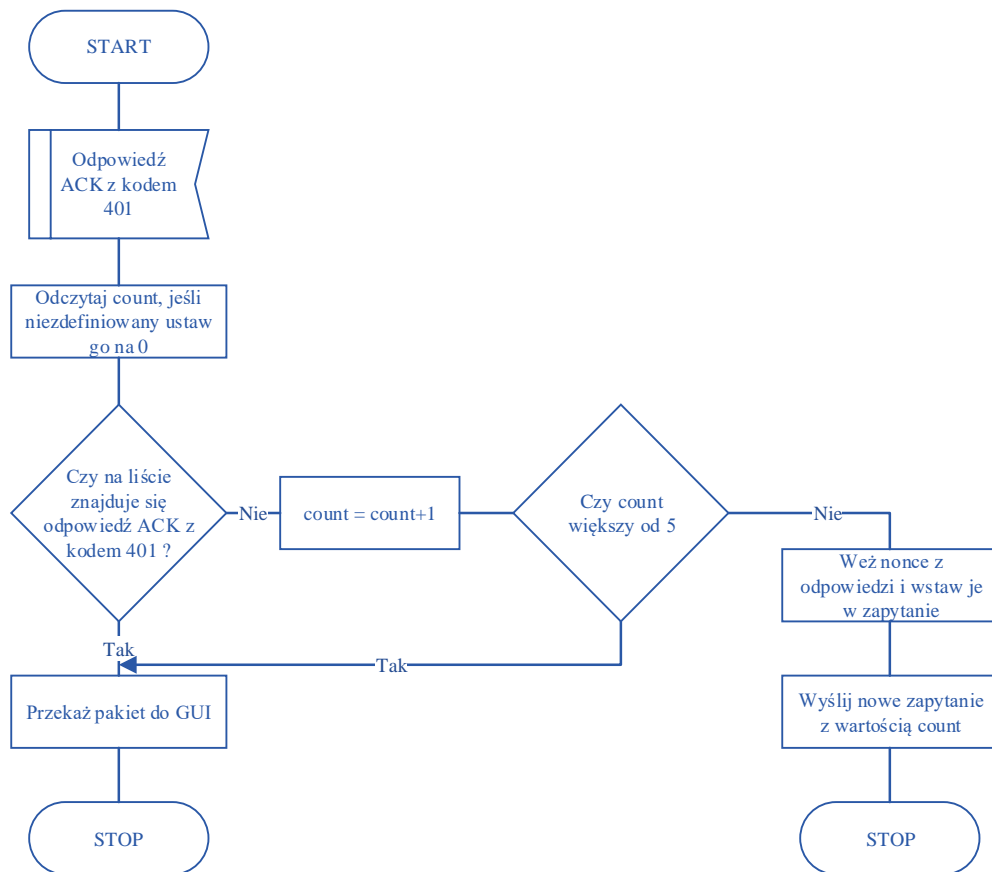
¹⁴ Fabryka – klasa, która dostarcza interfejs do tworzenia różnych obiektów tego samego typu, bez specyfikacji ich klas

¹⁵ Rzutowanie – jawna konwersja jednego typu klasy na inny

Biblioteka umożliwia także nasłuchiwanie wiadomości przychodzących. Do tego celu jest implementowany interfejs SipListener, a następnie ta implementacja jest dodawana podczas tworzenia zapytania sygnalizacji SIP. Dzięki temu można asynchronicznie przechwycić wiadomość przychodzącą, przetworzyć ją i jeżeli są jeszcze następne wiadomości do wysłania, to je wysłać. Problemem przy tworzeniu wiadomości do wysłania jest ustawienie odpowiedniego adresu IP w wiadomości i odpowiedniego portu do nasłuchiwania przez aplikację. Jeżeli pomylimy z tych rzeczy nie otrzymamy odpowiedzi od serwera. Użytkownikowi będzie podpowiadany pierwszy wolny port powyżej wartości 5060. Natomiast adres IP będzie uzyskiwany z połączenia do adresu publicznego DNS Google. Adres ten jest uzyskiwany w ten sposób, ponieważ podczas testów używany komputer posiadał dodatkową kartę sieciową, która nie miała połączenia z Internetem. Funkcja do uzyskania adresu z lokalnego komputera zwracała pierwszy napotkany adres IP. Nie było możliwości sprawdzenia bez wykonania połączenia, czy pobrany adres IP ma połączenie z Internetem.

Jednym z większych problemów było stworzenie mechanizmu autoryzacji podczas tworzenia własnego przepływu połączeń. Hasło jest wysyłane w postaci zakodowanej na serwer. Jest ono mieszane razem z innymi wartościami i wartością nonce¹⁶ za pomocą algorytmu MD5. To właśnie przez parametr nonce pojawiły się problemy z autoryzacją. Maksymalny czas ważności w używanym serwerze do testów (Asterisk) wynosił tylko 32 sekundy [25]. Praktycznie niemożliwe byłoby, aby użytkownik w tak krótkim czasie przygotował następne zapytanie do serwera. Podczas wysyłania błędnej odpowiedzi odnośnie autoryzacji dostajemy kod odpowiedzi 401 razem z nową wartością parametru nonce. Jeżeli chociaż jedna odpowiedź z serwera na liście wszystkich zapytań posiada kod odpowiedzi 401, to w takim przypadku pobierana jest nowa wartość nonce z odpowiedzi, następnie jest zamieniana stara wartość parametru nonce na nową i ponownie jest wysyłana na serwer. Maksymalnie program ponowi wysyłanie 5 razy, jeżeli za piątym razem się nie uda, oznacza to, że zapytanie jest błędne.

¹⁶ Nonce – w kryptografii dowolny ciąg znaków który może zostać użyty tylko raz w komunikacji



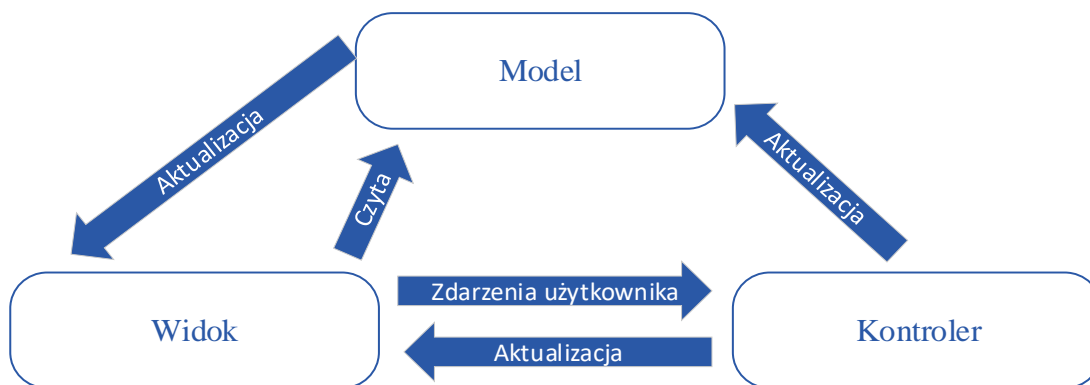
Rysunek 10. Schemat działania wysyłania pakietów posiadających autoryzację w projektowanej aplikacji

Ustawienie takiego zachowania programu ułatwi wysyłanie pakietów i zniweluje niepotrzebne duplikowanie tych samych zapytań.

6.2.4. MVC

Podczas tworzenia aplikacji został użyty wzorec projektowy Model-View-Controller (MVC). Wzorec ten ułatwia organizację aplikacji graficznej posiadającej graficzny interfejs użytkownika [26]. MVC dzieli się na trzy główne części:

- model – zawiera dane, które są wyświetlane bądź modyfikowane przez użytkownika,
- widok – opisuje w jaki sposób ma wyglądać graficzny interfejs użytkownika, jak ma być ułożony,
- kontroler – warstwa pośrednicząca pomiędzy modelem a widokiem. Przyjmuje zdarzenia generowane przez użytkownika i odpowiednio na nie reaguje



Rysunek 11. Zależności w wzorcu projektowym MVC [27]

Używanie takiej struktury aplikacji pozwala na odpowiednie rozdzielenie odpowiedzialności pomiędzy klasami, a także na ładny i czytelny kod programu. Zależności pomiędzy tymi częściami są wstrzykiwane za pomocą „Dependency Injection¹⁷” które jest dostarczone z modulem TornadoFx [9]. Pozwala to na nieuzależnienie się głównych modułów od siebie i oddzielenie logiki biznesowej¹⁸ od modeli i interfejsu użytkownika. Poprzez taki zabieg moduły mogą być niezależnie od siebie rozwijane.

6.2.5. Interfejs graficzny aplikacji

Główną klasą, która zarządza oknem aplikacji jest JFXDecorator z pakietu JFoenix. Ta klasa jest odpowiedzialna za to, aby aplikacja była utrzymywana w stylu Material Design. W interfejsie graficznym można się przełączać pomiędzy czterema oknami: ekran powitalny, strefa nauki, strefa symulacji i strefa pojęć. Pomiędzy oknami można się płynnie przełączać, ponieważ są umieszczone w obiekcie TabPane. Przy przełączeniu występuje animacja przesuwania okna. Widoczna jest tylko jedna karta na raz.

¹⁷ Dependency Injection – wzorec projektowy polegający na niwelowaniu bezpośrednich zależności pomiędzy klasami, przekazywane są gotowe instancje obiektów

¹⁸ Logika biznesowa – zawiera obiekty wykonujące zadanie funkcjonalności aplikacji

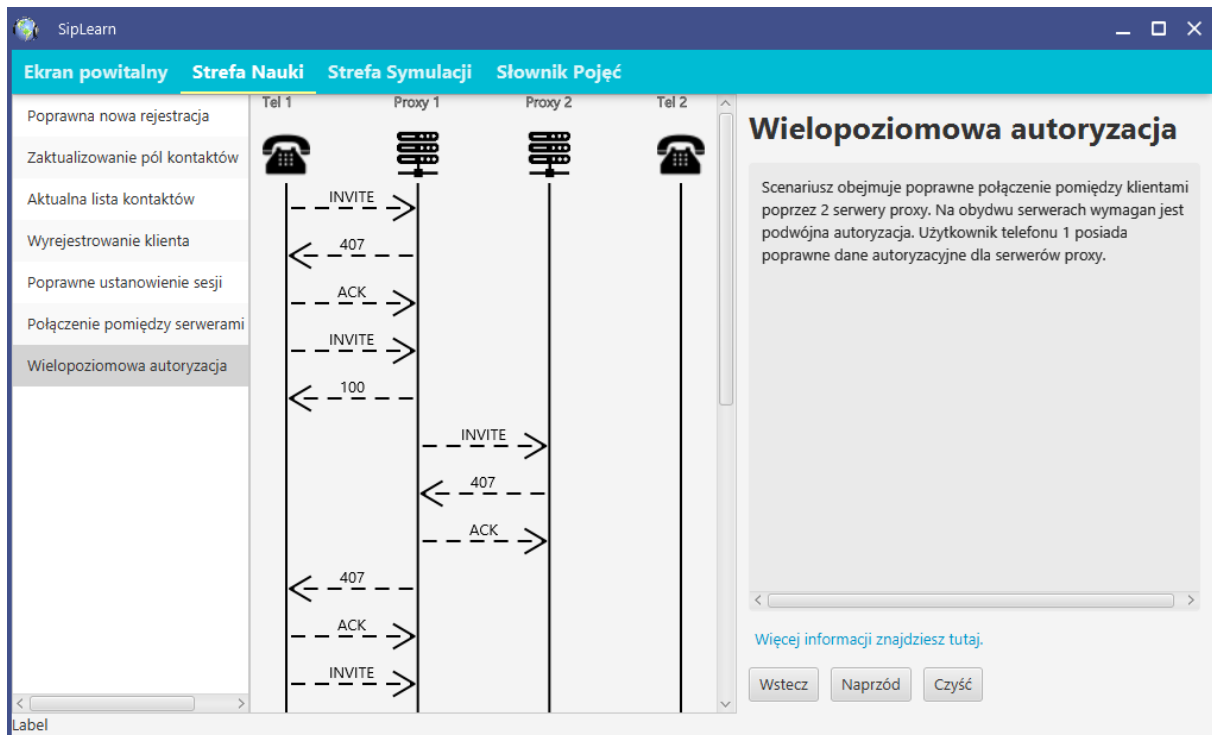
6.2.5.1. Ekran powitalny



Rysunek 12. Ekran powitalny projektowanej aplikacji

Na ekranie powitalnym znajduje się krótka informacja odnośnie programu, logo Politechniki Wrocławskiej oraz lista najważniejszych technologii jakie zostały wykorzystane przy tworzeniu tej aplikacji.

6.2.5.2. Strefa nauki

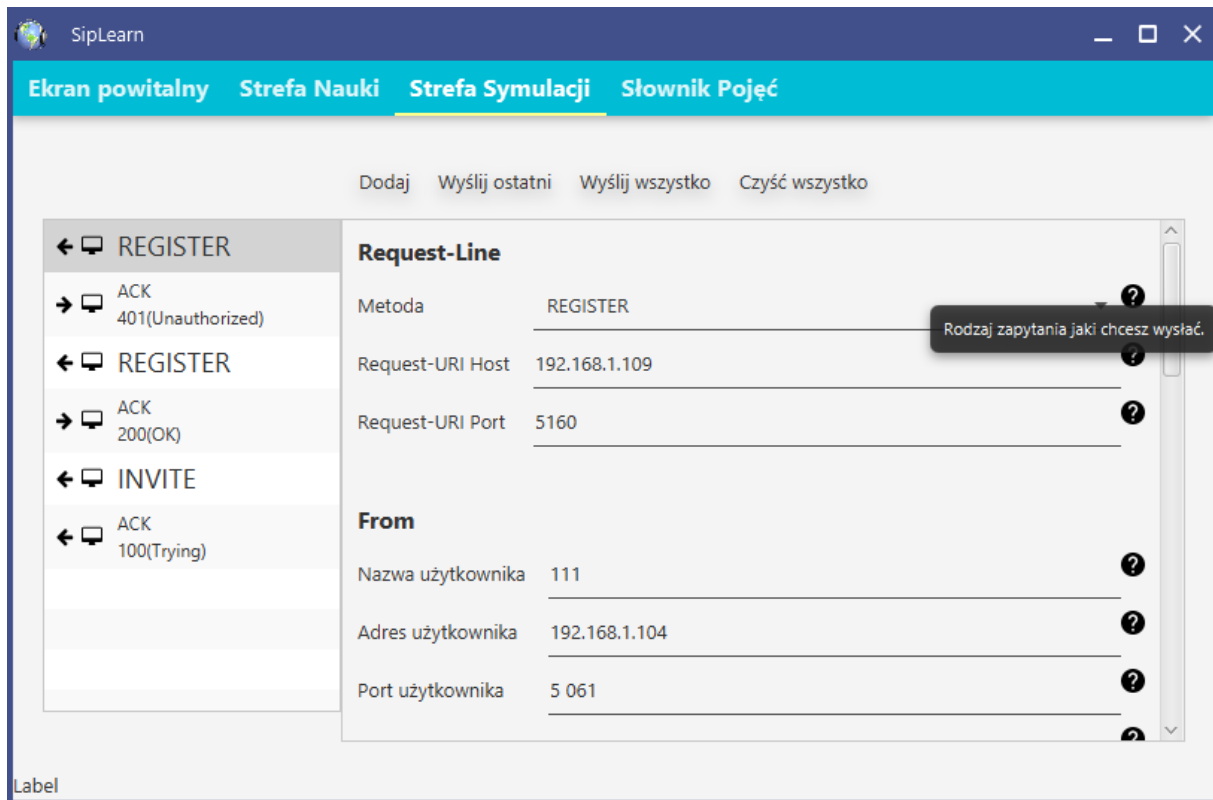


Rysunek 13. Ekran "Strefy Nauki" projektowanej aplikacji

Ekran o nazwie „Strefa Nauki” ma za zadanie przedstawić podstawowe rodzaje przepływów połączeń w sygnalizacji SIP. Użytkownik będzie miał do dyspozycji kilkanaście scenariuszy do wyboru. W każdym z nich znajduje się krótkie wprowadzenie do wybranego scenariusza. Klikając przyciski „Naprzód” i „Wstecz” można obserwować przepływ wiadomości w sygnalizacji SIP. Do każdego etapu transmisji danych jest dołączony tekstowy opis oraz przykładowa ramka transmisji z normy RFC3665 [28]. Z tego dokumentu będą wzięte przykładowe scenariusze w środowisku SIP. Program daje możliwość pogłębienia dalszej wiedzy poprzez link umieszczony nad przyciskami. Otwiera on przeglądarkę z odpowiednio dobraną częścią normy RFC odnoszącą się do aktualnie wybranego scenariusza.

Aktualnie zaprojektowany mechanizm pozwala na animację przepływu pomiędzy dwoma, trzema i czterema punktami. Przyszłościowo zostanie dodana także możliwość dodania scenariuszy za pomocą pliku tekstowego. Pozwoli to na dalszą rozbudowę modułu edukacyjnego bez potrzeby ingerowania w kod aplikacji.

6.2.5.3. Strefa Symulacji



Rysunek 14. Ekran Strefy Symulacji projektowanej aplikacji wraz z wygenerowanym przepływem połączeń

Ekran o nazwie „Strefa Symulacji” implementuje najciekawszą funkcję programu, czyli możliwość łatwego i intuicyjnego generowania wiadomości w środowisku SIP. Użytkownik będzie miał możliwość stworzenia wiadomości poprzez wypełnienie formularza. Przy każdym elemencie do wypełnienia znajduje się ikona pomocy, przy której po najechnaniu pojawi się krótka informacja odnośnie tego pola. Na początek użytkownik będzie musiał wypełnić wymagane nagłówki:

- Request-Line – odpowiada za rodzaj wysyłanej wiadomości,
- From – nagłówek informujący od kogo wiadomość jest wysyłana,
- To – nagłówek informujący do kogo wiadomość jest wysyłana, w przypadku zapytania REGISTER wartości powinny być takie same jak w nagłówku From,
- Via – nagłówek informujący z jakiego adresu i portu będzie wysyłana wiadomość, oraz jaki protokół zostanie do tego użyty,
- Call-ID – unikatowy identyfikator dla zapytania,
- CSeq – służy do identyfikowania zapytania i do retransmisji

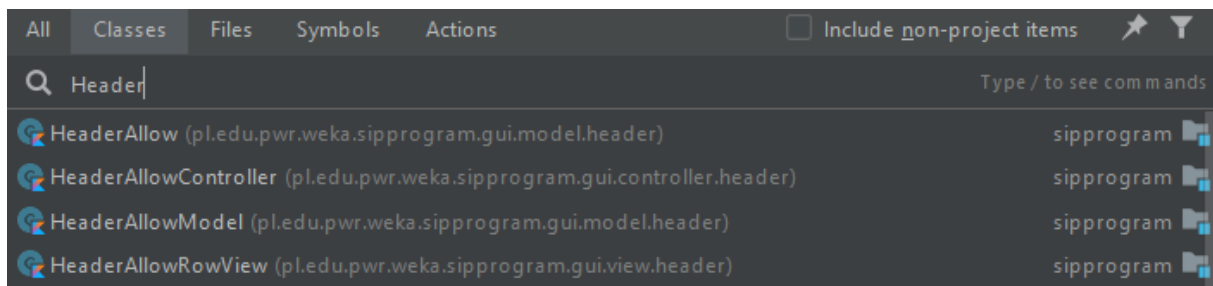
Wysyłać wiadomości można na dwa sposoby: tylko ostatni lub wszystko. Pierwszy sposób przed wysłaniem wiadomości usunie wiadomość przychodzącą z listy przepływu połączeń

umieszczonej z lewej strony. Następnie są tworzone nagłówki na podstawie danych wypełnionych przez użytkownika. Po utworzeniu danych jest tworzony tzw. „listener” który ma za zadanie nasłuchiwać wiadomości przychodzące z serwera do naszej aplikacji. Po utworzeniu listenera jest on dodawany do wysyłanej wiadomości, a następnie wiadomość jest wysyłana na serwer. Jeżeli wszystkie pola wypełniliśmy dobrze, na liście z lewej strony pojawi się wiadomość ACK z odpowiednim kodem statusu i wiadomością.

Moduł odnośnie symulacji był najcięższym elementem do stworzenia i to on pochłonął najwięcej czasu. Głównym problemem była mała ilość dokumentacji odnośnie pakietu JAIN SIP API. Musiałem się wzorować na innych projektach i poradnikach umieszczonych w Internecie. Bardzo dużą ilość czasu spędziłem także na przeglądanie kodu źródłowego wspomnianego pakietu. Nie jest on przygotowany na ręczne tworzenie i wysyłanie sygnalizacji SIP. Często musiałem szukać obejść jakichś problemów, ponieważ pakiet nie oferował zadowalającego rozwiązania dla mnie.

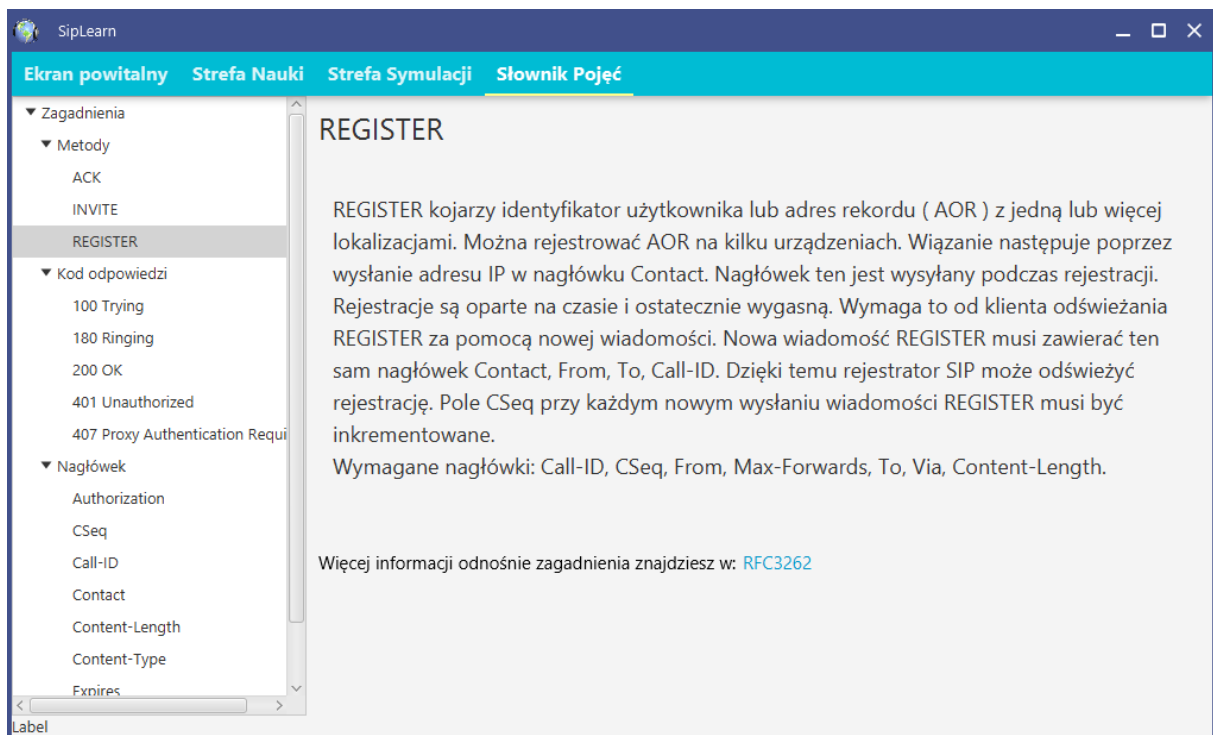
Drugim problemem jaki napotkałem, był sposób graficznej reprezentacji wiadomości sygnalizacji SIP. Musiałem znaleźć sposób, aby w dynamiczny sposób można było dodawać nagłówki do zapytania, oraz żeby dla użytkownika nie było to kłopotliwe. W pierwszej wersji zapytania REGISTER, INVITE itd. Miały posiadać swoje własne widoki formularzy, które byłyby wyświetlane w oknie. Jednak z tego pomysłu szybko zrezygnowałem, z racji na bardzo dużą ilość powtarzającego się kodu. Taki kod w późniejszej fazie projektu byłby trudny do utrzymania. Zdecydowałem się na stworzenie dla każdego nagłówka osobnego widoku, kontrolera i modelu. Decyzja ta sprawiła, że w programie pojawiło się bardzo dużo małych klas, jednak dzięki temu kod jest bardziej czytelny.

Ostatecznie moduł ten, z uwagi na liczne błędy i niedociągnięcia, zostanie wyłączony podczas wdrażania niniejszego programu w Pracowni Sieci Telekomunikacyjnej. Według mnie nie został on odpowiednio przygotowany, aby studenci mogli go używać. Podczas licznych testów stwierdziłem, że użytkowanie tego modułu jest uciążliwe, oraz w chwili obecnej jest pozbawione walorów edukacyjnych. W celu prezentacji implementacji tej części programu dodałem cztery przykładowe ramki wiadomości, które użytkownik może zobaczyć.



Rysunek 15. Przykładowy zestaw klas do obsługi nagłówka Allow w projektowanym programie

6.2.5.4. Słownik Pojęć



Rysunek 16. Ekran słownika pojęć projektowanej aplikacji

Ostatnim z prezentowanych ekranów jest ekran Słownika Pojęć. Zawiera on w sobie listę najważniejszych pojęć dotyczących środowiska SIP. Lista pojęć została wzięta ze strony organizacji IANA¹⁹ [29]. Zagadnienia są odpowiednio pogrupowane i wyświetlane w postaci drzewa. Przy zagadnieniu znajduje się krótki opis. W celu dalszego poszerzenia wiedzy pod opisem znajduje się link, pod którym można znaleźć więcej informacji odnośnie zagadnienia.

6.2.6. Scenariusze w „Strefie Nauki”

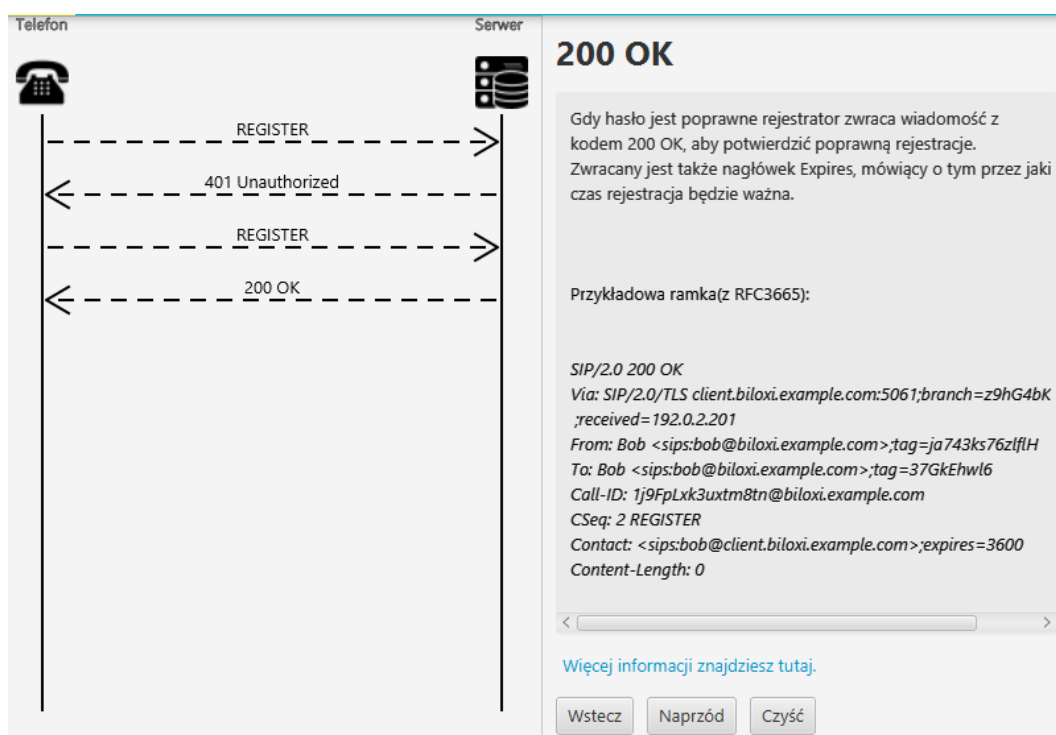
W chwili obecnej w programie jest zaimplementowane siedem scenariuszy związanych z sygnalizacją SIP. Każdy z nich posiada krótki opis poszczególnych etapów połączenia oraz

¹⁹ IANA (Internet Assigned Numbers Authority) – organizacja zajmująca się organizacją domen i adresów IP w Internecie

odnośnik do dokumentacji, gdzie użytkownik może dowiedzieć się więcej. Scenariusze wzorowano na przykładach z dokumentu RFC3665 [28].

6.2.6.1. Poprawna nowa rejestracja

Opis scenariusza: Bez rejestracji można się komunikować jedynie bezpośrednio do innego użytkownika, ale i w takim przypadku należy znać adres IP użytkownika docelowego. Proces rejestracji usuwa te problemy i bez przeszkód można się komunikować z innym użytkownikiem. REGISTER kojarzy identyfikator użytkownika – AOR (Address of Record) z jedną lub większą ilością lokalizacji. AOR można zarejestrować na wielu urządzeniach. Podczas rejestracji wysyłany jest nagłówek Contact, w którym zawarte jest AOR i adres IP. Nagłówek ten informuje, gdzie chcemy otrzymywać zapytania.



Rysunek 17. Graficzna reprezentacja przepływu sygnalizacji w scenariuszu "Poprawna nowa rejestracja"

Przepływ sygnałów:

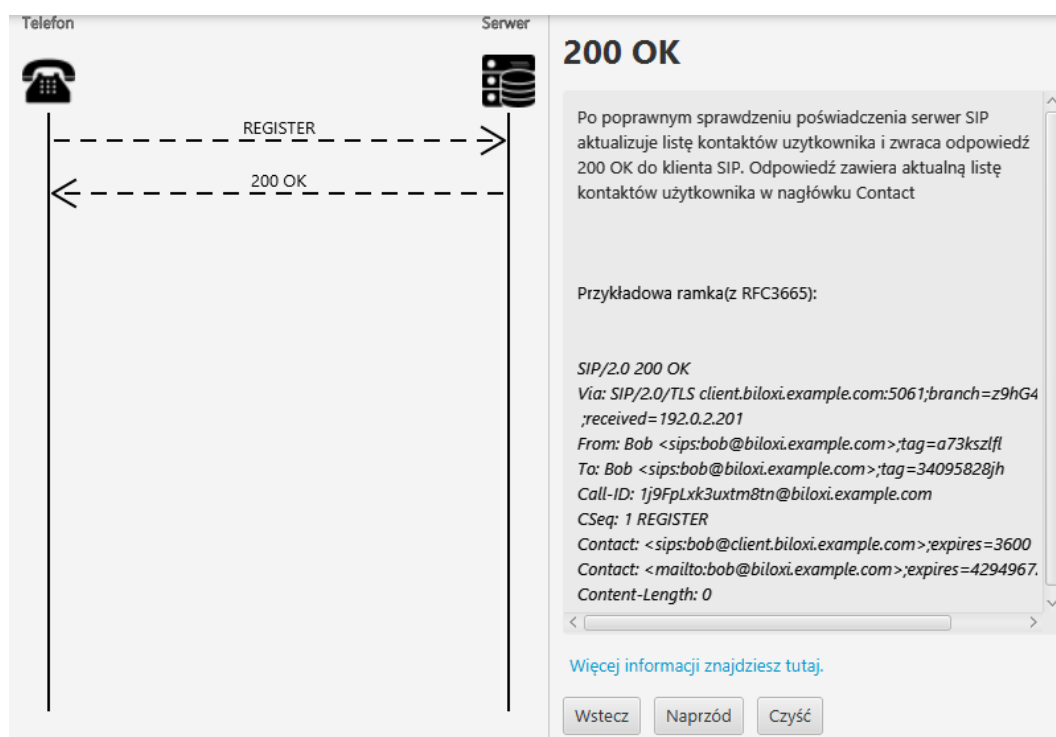
- REGISTER - użytkownik wysyła zapytanie REGISTER do rejestratora SIP. Nagłówki From i To powinny zawierać takie samo AOR użytkownika. Użytkownik dodatkowo może określić czas, przez który rejestracja powinna być ważna za pomocą nagłówka Expires. Wartość ta może być później zmodyfikowana przez rejestrator.
- 401 Unauthorized - rejestrator zwraca wiadomość z kodem 401 Unauthorized. W wiadomości tej znajduje się nagłówek WWW-Authenticate zawierający

dane, które muszą zostać użyte do zaszyfrowania hasła użytkownika. Najważniejszymi parametrami z tego nagłówka jest unikalny numer (nonce) i algorytm szyfrowania.

- REGISTER - Użytkownik wysyła zapytanie REGISTER do rejestratora SIP. Zapytanie to zawiera nagłówek Authorization. W tym nagłówku znajduje się zaszyfrowane hasło użytkownika.
- 200 OK - Gdy hasło jest poprawne rejestrator zwraca wiadomość z kodem 200 OK, aby potwierdzić poprawną rejestrację. Zwracany jest także nagłówek Expires, mówiący o tym przez jaki czas rejestracja będzie ważna.

6.2.6.2. Zaktualizowanie pól kontaktów

Opis scenariusza: W celu zaktualizowania listy adresów, pod które serwer SIP będzie przekierowywał lub przysyłał żądania INVITE, wysyła się odpowiednio przygotowaną wiadomość REGISTER. Użytkownik jest już uwierzytelniony na serwerze i posiada odpowiednie poświadczenie.



Rysunek 18. Graficzna reprezentacja przepływu sygnalizacji w scenariuszu "Zaktualizowanie pól kontaktów"

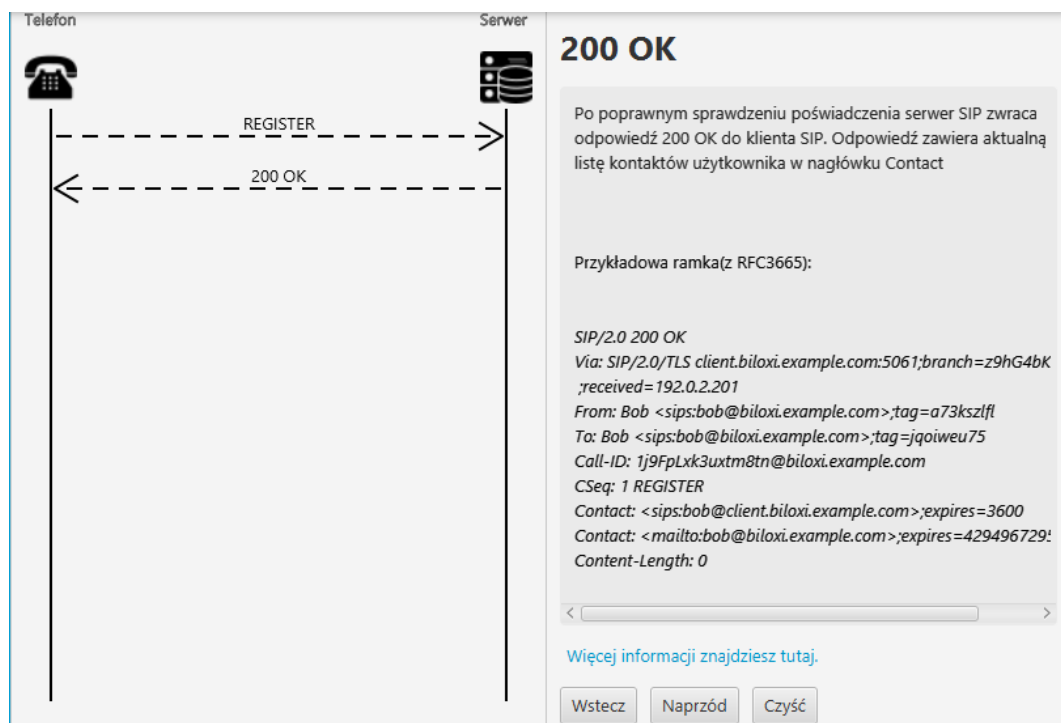
Przebieg sygnałów:

- REGISTER - Wysyłana jest wiadomość REGISTER do serwera. Żądanie to zawiera nowe adresy, pod które serwer SIP będzie kierował żądania INVITE.

- 200 OK - Po poprawnym sprawdzeniu poświadczenia serwer SIP aktualizuje listę kontaktów użytkownika i zwraca odpowiedź 200 OK do klienta SIP. Odpowiedź zawiera aktualną listę kontaktów użytkownika w nagłówku Contact

6.2.6.3. Aktualna lista kontaktów

Opis scenariusza: Użytkownik chce dostać informację odnośnie aktualnej listy kontaktów użytkownika. Użytkownik jest już uwierzytelniony na serwerze i posiada odpowiednie poświadczenie.



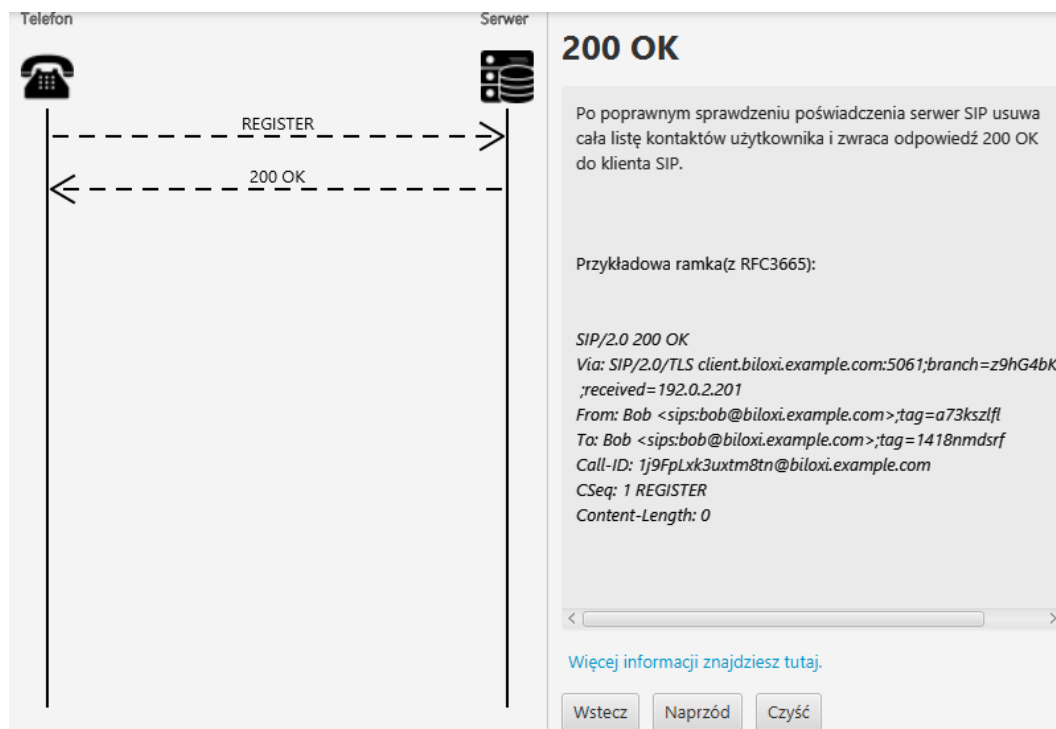
Rysunek 19. Graficzna reprezentacja przepływu sygnalizacji w scenariuszu "Aktualna lista kontaktów"

Przepływ sygnałów:

- REGISTER - Wysyłana jest wiadomość bez nagłówka Contact, ponieważ użytkownik chce wysłać zapytanie do serwera o aktualną listę kontaktów.
- 200 OK - Po poprawnym sprawdzeniu poświadczenia serwer SIP zwraca odpowiedź 200 OK do klienta SIP. Odpowiedź zawiera aktualną listę kontaktów użytkownika w nagłówku Contact.

6.2.6.4. Wyrejestrowanie klienta

Opis scenariusza: Użytkownik chce się wyrejestrować z serwera SIP. Jest on już uwierzytelniony na serwerze i posiada odpowiednie poświadczenie.



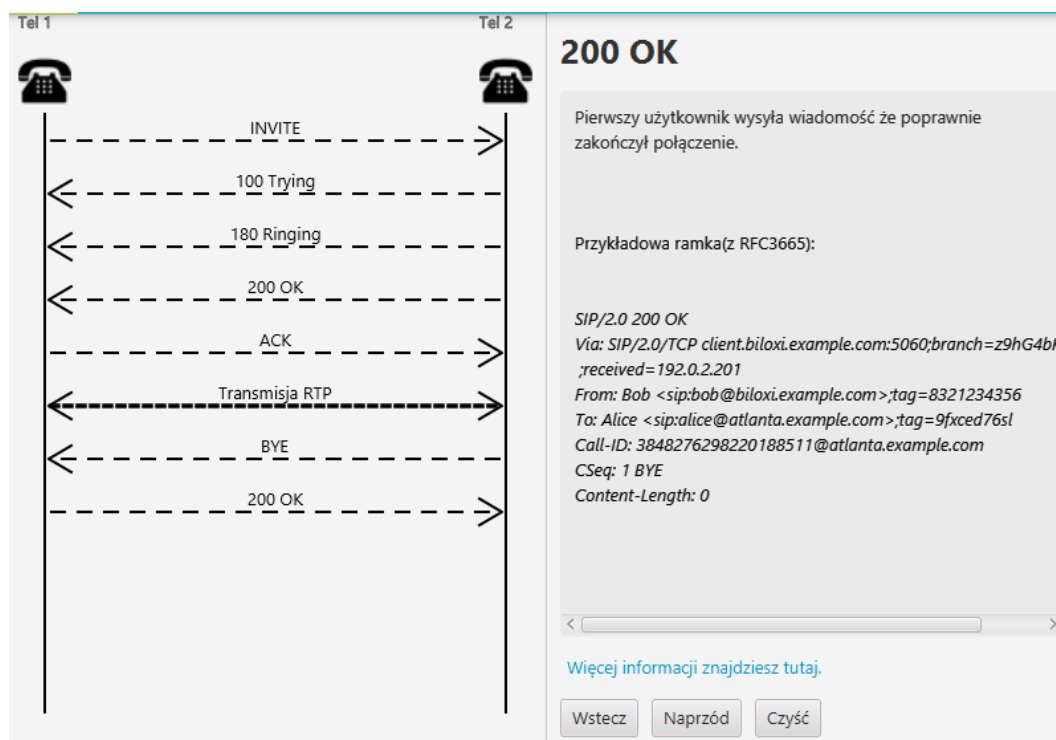
Rysunek 20. Graficzna reprezentacja przepływu sygnalizacji w scenariuszu "Wyrejestrowanie klienta"

Przepływ sygnałów:

- REGISTER - Wysyłana jest wiadomość REGISTER do serwera SIP z nagłówkiem Expires równym 0 dla wszystkich istniejących pól kontaktów. Nagłówek ten odpowiada za okres ważności danej operacji.
- 200 OK - Po poprawnym sprawdzeniu poświadczenia serwer SIP usuwa całą listę kontaktów użytkownika i zwraca odpowiedź 200 OK do klienta SIP.

6.2.6.5. Poprawne ustanowienie sesji

Opis scenariusza: Połączenia są ustalane za pomocą metody INVITE razem z protokołem Session Description Protocol. SDP zawiera niezbędne informacje do inicjalizacji mediów strumieniowych. Po wysłaniu wiadomości od odbiorcy jest otrzymywana wiadomość Ringing w celu wskazania, że docelowy terminal dzwoni. Wiadomość ta jest wysyłana, aby zapobiec niepotrzebnej retransmisji wiadomości od nadawcy. Osoba docelowa może zaakceptować nasze połączenie poprzez wysłanie wiadomości OK lub odrzucić je z błędem. W każdej chwili można przerwać połączenie wysyłając komunikat CANCEL. Użytkownik jest już uwierzytelniony na serwerze i posiada odpowiednie poświadczenie.



Rysunek 21. Graficzna reprezentacja przepływu sygnalizacji w scenariuszu "Poprawne ustanowienie sesji"

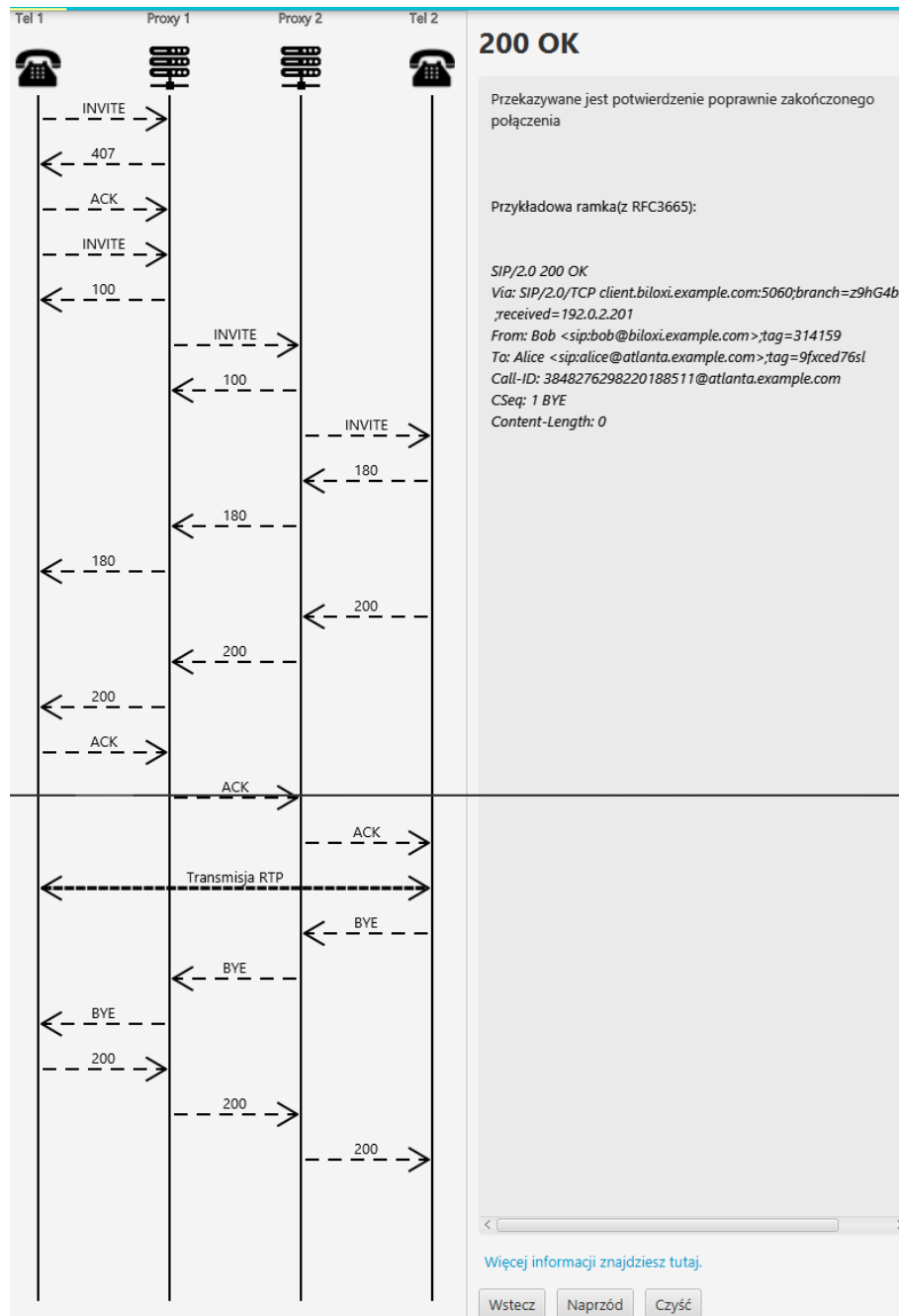
Przebieg sygnałów:

- INVITE - Wysyłana jest metoda INVITE wraz z wybieranym numerem. Wiadomość zawiera informację, że dodatkowo będą wysyłane parametry multimedialnych w postaci SDP, oraz jest podana ich długość.
- 100 Trying - Serwer natychmiast odpowiada za pomocą metody Trying, aby wskazać, że wiadomość została odebrana i uniknąć ponownych transmisji od klienta.
- 180 Ringing - Odbiorca wskazuje, że jego terminal dzwoni.
- 200 OK - Po zaakceptowaniu połączenia wysyłany jest komunikat 200 razem z informacjami odnośnie inicjalizacji multimedialnych.
- ACK - Wysyłana jest ramka potwierdzająca, że zakończono negocjacje odnośnie multimedialnych i połączenie zostało poprawnie skonfigurowane.
- Transmisja RTP - Tworzone są strumienie transmisji RTP między dwoma użytkownikami. Zawierają one informacje o typie przesyłanych danych, numery sekwencyjne oraz znaczniki czasu. RTP nie gwarantuje jakości usług (QoS).
- BYE - Drugi użytkownik wysła wiadomość zakończenia połączenia. Licznik CSeq ma wartość 1 a nie 2, ponieważ użytkownicy utrzymują swoje niezależne liczby CSeq.

- 200 OK - Pierwszy użytkownik wysyła wiadomość, że poprawnie zakończył połączenie.

6.2.6.6. Połączenie pomiędzy serwerami proxy

Opis scenariusza: Scenariusz obejmuje poprawne połączenie pomiędzy klientami poprzez dwa serwery proxy.



Rysunek 22. Graficzna reprezentacja przepływu sygnalizacji w scenariuszu "Połączenie pomiędzy serwerami proxy"

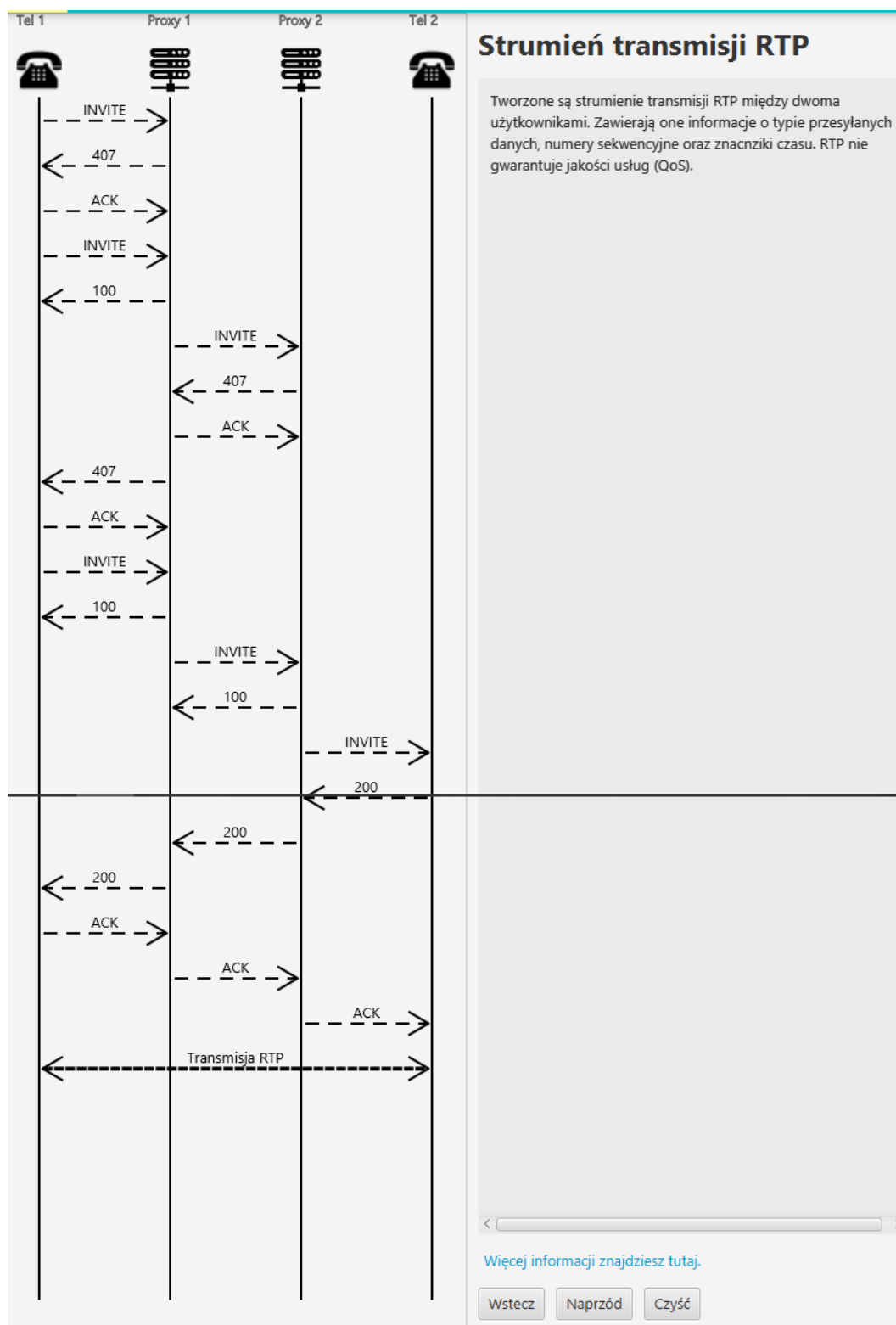
Przepływ sygnałów:

- INVITE - Z telefonu 1 wysyłana jest wiadomość INVITE do proxy 1. Proxy 1 jest skonfigurowany jako domyślny serwer proxy dla telefonu 1. Wysyłana wiadomość INVITE zawiera nagłówek trasy do proxy 1 i nie zawiera danych uwierzytelniających wymaganych przez serwer proxy 1
- 407 Proxy Authentication Required - Wymagana autoryzacja proxy. Do klienta wysyłane są dane potrzebne do autoryzacji użytkownika.
- ACK - Użytkownik wysyła, że potwierdza otrzymanie wiadomości odnośnie sposobu uwierzytelnienia na serwerze proxy 1.
- INVITE - Wysyłana jest ponownie wiadomość INVITE dodatkowo z danymi autoryzującymi na proxy 1.
- 100 Trying - Wysyłana jest wiadomość, że żądanie jest wykonywane. Ramka ta jest wysyłana, aby nie wykonano retransmisji poprzedniej wiadomości.
- INVITE - Proxy 1 zaakceptowało dane logowania i przekazuje ramkę INVITE do Proxy 2. Telefon 1 przygotowuje się do odbioru danych na porcie 49172.
- 100 Trying - Wysyłana jest wiadomość, że żądanie jest wykonywane. Ramka ta jest wysyłana, aby nie wykonano retransmisji poprzedniej wiadomości.
- INVITE - Wiadomość INVITE dociera do telefonu 2. Zawiera ona adresy wszystkich punktów przez które musiała przejść, oraz drogę, którą pokonała. Droga ta jest zawarta w nagłówku Record-Route.
- 180 Ringing - Przekazywany jest sygnał, że został uruchomiony sygnał dzwonienia w telefonie 2.
- 180 Ringing - Przekazywany jest sygnał, że został uruchomiony sygnał dzwonienia w telefonie 2.
- 180 Ringing - Telefon 1 otrzymuje sygnał, że został uruchomiony sygnał dzwonienia w telefonie 2.
- 200 OK - Przekazywana jest wiadomość, że użytkownik telefonu 2 zaakceptował połączenie (podniósł słuchawkę). Dodatkowo wysyłane są informacje odnośnie transmisji multimedialnej.
- 200 OK - Przekazywana jest wiadomość, że użytkownik telefonu 2 zaakceptował połączenie (podniósł słuchawkę). Dodatkowo wysyłane są informacje odnośnie transmisji multimedialnej.

- 200 OK - Użytkownik telefonu 1 dostaje informację, że użytkownik telefonu 2 podniósł słuchawkę, oraz otrzymuje parametry odnośnie transmisji multimedialnej.
- ACK - Do telefonu 2 jest wysyłane potwierdzenie negocjacji parametrów transmisji multimedialnej, oraz że sesja została poprawnie stworzona.
- ACK - Przekazywana jest wiadomość odnośnie potwierdzenia stworzenia sesji.
- ACK - Przekazywana jest wiadomość odnośnie potwierdzenia stworzenia sesji.
- Transmisja RTP - Tworzone są strumienie transmisji RTP między dwoma użytkownikami. Zawierają one informacje o typie przesyłanych danych, numery sekwencyjne oraz znaczniki czasu. RTP nie gwarantuje jakości usług (QoS).
- BYE - Użytkownik telefonu 2 wysyła informację o zakończeniu połączenia.
- BYE - Przekazywana jest informacja o zakończeniu połączenia.
- BYE - Przekazywana jest informacja o zakończeniu połączenia.
- 200 OK - Wysyłane jest potwierdzenie poprawnego zakończenia połączenia.
- 200 OK - Przekazywane jest potwierdzenie poprawnie zakończonego połączenia.
- 200 OK - Przekazywane jest potwierdzenie poprawnie zakończonego połączenia.

6.2.6.7. Wielopoziomowa autoryzacja

Opis scenariusza: Scenariusz obejmuje poprawne połączenie pomiędzy klientami poprzez dwa serwery proxy. Na obydwu serwerach wymagana jest podwójna autoryzacja. Użytkownik telefonu 1 posiada poprawne dane autoryzacyjne dla serwerów proxy.



Rysunek 23. Graficzna reprezentacja przepływu sygnalizacji w scenariuszu "Wielopoziomowa autoryzacja"

Przepływ sygnałów:

- INVITE - Wysyłana jest wiadomość INVITE z parametrami transmisji multimedialnej, bez danych autoryzacyjnych.

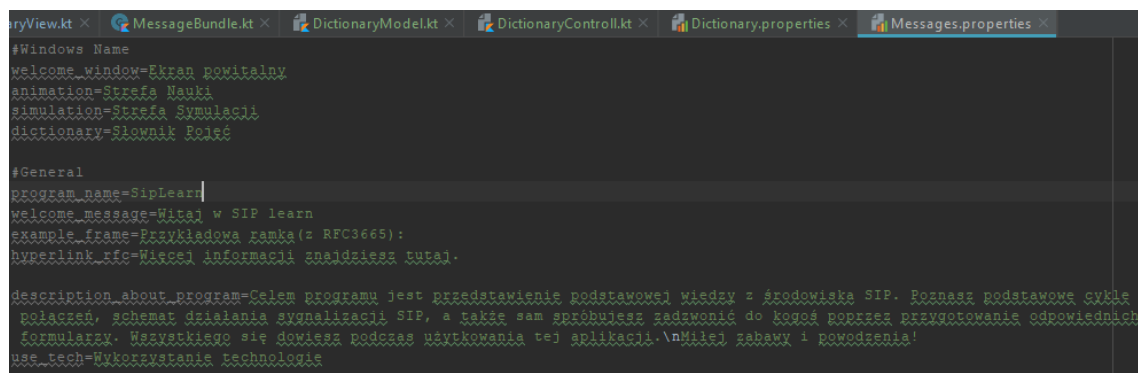
- 407 Proxy Authentication Required - Serwer proxy 1 wymaga autoryzacji i zwraca kod błędu 407 oraz dane niezbędne do autoryzacji.
- ACK - Wysyłana jest wiadomość potwierdzająca, że przyszła wiadomość z kodem błędu.
- INVITE - Ponownie jest wysyłana wiadomość INVITE, dodatkowo są dodane dane autoryzacyjne dla proxy 1.
- 100 Trying - Odsyłana jest ramka z kodem 100 w celu poinformowania, że żądanie jest aktualnie obsługiwane.
- INVITE - Wiadomość INVITE jest przesyłana dalej do proxy 2.
- 407 Proxy Authorization Required - Proxy 2 także wymaga autoryzacji i zwraca kod błędu 407 razem z informacjami niezbędnymi do autoryzacji
- ACK - Wysyłana jest wiadomość ACK w celu potwierdzenia, że kod błędu został odebrany.
- 407 Proxy Authorization Required - Wiadomość z kodem błędu 407 jest przesyłana do użytkownika początkowego.
- ACK - Wysyłana jest wiadomość potwierdzająca, że kod błędu został odebrany.
- INVITE - Ponownie jest wysyłana wiadomość INVITE z kolejnymi danymi autoryzującymi dla proxy 2.
- 100 Trying - Odsyłana jest ramka z kodem 100 w celu poinformowania, że żądanie jest aktualnie obsługiwane.
- INVITE - Wiadomość INVITE jest poprawnie autoryzowana na serwerze proxy 1 i jest przesyłana do serwera proxy 2.
- 100 Trying - Odsyłana jest ramka z kodem 100 w celu poinformowania, że żądanie jest aktualnie obsługiwane.
- INVITE - Wiadomość INVITE dociera do użytkownika końcowego.
- 200 OK - Użytkownik końcowy natychmiast odbiera połączenie i wysyłana jest wiadomość z kodem 200.
- 200 OK - Wiadomość z kodem 200 jest dalej przesyłana z serwera proxy 2 do serwera proxy 1.
- 200 OK - Wiadomość z kodem 200 jest dalej przesyłana z serwera proxy 1 do użytkownika początkowego.
- ACK - Wysyłana jest wiadomość ACK informująca o poprawnej negocjacji transmisji multimedialnej oraz o poprawnym stworzeniu sesji. Dodatkowo

wiadomość posiada wszystkie niezbędne dane do autoryzacji na serwerze proxy 1 i proxy 2.

- ACK - Wiadomość ACK jest przesyłana z proxy 1 do proxy 2.
- ACK - Wiadomość ACK jest przesyłana z proxy 2 do użytkownika docelowego.
- Transmisja RTP - Tworzone są strumienie transmisji RTP między dwoma użytkownikami. Zawierają one informacje o typie przesyłanych danych, numery sekwencyjne oraz znaczniki czasu. RTP nie gwarantuje jakości usług (QoS).

6.2.7. Internacjonalizacja aplikacji

W programie będzie możliwość translacji wyświetlanych opisów. Wszystkie pola tekstowe znajdują się w odpowiednich plikach o rozszerzeniu .properties, które są przypisane do odpowiednich języków i automatycznie ustawiane po wykryciu języka systemowego. Przykładowo plik Messages_pl.properties zostanie wczytany, gdy program wykryje, że na systemie Windows jest ustawiony język Polski. Jeśli nie będzie pliku dla języka, program wczyta teksty z pliku Messages.properties.



```
#Windows Name
welcome_window=Ekran powitalny
animation=Strefa Nauki
simulation=Strefa Symulacji
dictionary=Słownik Polac

#General
program_name=SipLearn
welcome_message=Witaj w SIP learn
example_frame=Przykładowa ramka (= RFC3665):
hyperlink_rfc=Więcej informacji znajdziesz tutaj.

description_about_program=Celem programu jest przedstawienie podstawowej wiedzy z środowiska SIP. Poznasz podstawowe cykle połączeń, schemat działania sygnalizacji SIP, a także sam spróbujesz zadzwonić do kogoś poprzez przygotowanie odpowiednich formularzy. Wszystkie się dowiesz podczas użytkowania tej aplikacji.\nWitaj znowu i powodzenia!
use_tech=Wykorzystanie technologii
```

Rysunek 24. Przykładowe wartości klucz=wartość w pliku Messages.properties

6.3. Przyszłość programu

Aktualna wersja programu, pomimo swojego zaawansowania, nie może zostać jeszcze oddana w celu używania jej w Pracowni Sieci Telekomunikacyjnych. Główne założenia programu zostały spełnione, tj. posiada walor edukacyjny i możliwość definiowania przepływu wiadomości w sygnalizacji SIP. Jednakże w programie jest jeszcze kilka rzeczy do udoskonalenia:

- w Strefie Nauki:
 - bardzo małą ilość zaimplementowanych scenariuszy,
- w Strefie Symulacji:
 - przetestowano jedynie działanie metody REGISTER,

- brak dużej ilości nagłówków,
- brak walidacji wprowadzanych danych,
- brak opisów dotyczących edytowanych pól,
- brak możliwości usuwania i klonowania zapytań i nagłówków,
- w Słowniku Pojęć:
 - małą ilość pojęć,

Powyższe niedociągnięcia wynikają z zbyt krótkiego okresu na napisanie aplikacji oraz trudności związane z użytymi technologiami. Wykorzystane technologie zostały przeze mnie użyte po raz pierwszy, dlatego miałem trudności ze zrozumieniem niektórych rzeczy. Dużą ilość czasu spędziłem na czytaniu dokumentacji oraz na testowaniu różnych rozwiązań. Nie znalazłem podobnej aplikacji, która służyłaby tylko do nauki sygnalizacji SIP, dlatego wymyślenie i zaimplementowanie niektórych rozwiązań zajęło mi sporą ilość czasu.

7. Wnioski

Zakres pracy obejmuje omówienie podstawowych protokołów sygnalizacyjnych oraz przedstawienie przykładowych programów służących do testowania sygnalizacji SIP. W pracy przybliżyłem także technologie, które użyłem do stworzenia programu edukacyjnego. Część praktyczna pracy obejmuje napisanie i uruchomienie aplikacji demonstrującej funkcjonowanie protokołów komunikacyjnych pracujących w środowisku SIP. Aplikacja pomimo licznych niedociągnięć, spełnia swoje podstawowe założenia. Została ona napisana w języku programowania Kotlin i w celu stworzenia interfejsu graficznego zostały użyte najnowsze technologie dostępne aktualnie na rynku. Celem moim było stworzenie aplikacji, którą studenci mogli by używać w Pracowni Sieci Telekomunikacyjnych podczas swoich laboratoriów. Idea ta trochę mnie przerosła, jednakże przy dopracowaniu aplikacji może spełniać zamierzoną funkcjonalność. Działanie aplikacji zostało przetestowane na komputerach w laboratorium Pracowni Sieci Telekomunikacyjnych.

Projektowanie aplikacji z interfejsem graficznym jest bardzo trudnym przedsięwzięciem. Największym problemem nie jest stworzenie wyobrażenia jak aplikacja ma wyglądać, lecz zaimplementowanie tego. Niekiedy musiałem iść na kompromis tworząc niektóre elementy GUI. Także mechanizm działania sygnalizacji SIP był na początku trudny do zrozumienia dla mnie. Aplikacje można w dalszym ciągu rozwijać, także upublicznienie kodu może bardzo pozytywnie przyczynić się do rozwoju.

8. Bibliografia

- [1] L. Chaffin, „SIP Architecture,” w *Building a VoIP Network with Nortel's Multimedia Communication Server 5100*, Syngress, 2006, pp. 345-386.
- [2] A. Tulibacka, „Czym jest Material Design?,” Grafmag, 29 Luty 2016. [Online]. Available: <https://grafmag.pl/artykuly/czym-jest-material-design-teoria-zasady-materialy-i-przyklady>. [Data uzyskania dostępu: 1 Grudzień 2018].
- [3] A. Kristensen, „SIP Servlet API Specification 1.0 Final Release,” Sun Microsystems, 5 Sierpień 2003. [Online]. Available: <https://jcp.org/aboutJava/communityprocess/final/jsr116/index.html>. [Data uzyskania dostępu: 1 Grudzień 2018].
- [4] J. K. Waters, „Kotlin Goes Open Source,” 22 Luty 2012. [Online]. Available: <https://adtmag.com/articles/2012/02/22/kotlin-goes-open-source.aspx>. [Data uzyskania dostępu: 04 Listopad 2018].
- [5] D. Jemerov i S. Isakova, *Kotlin in Action*, Manning Publication Co., 2017.
- [6] M. Debnath, „What is JavaFX? An Introduction,” 1 Kwiecień 2016. [Online]. Available: <https://www.developer.com/java/data/what-is-javafx-an-introduction.html>. [Data uzyskania dostępu: 04 Listopad 2018].
- [7] Tutorials Point, „JavaFX - Application,” [Online]. Available: https://www.tutorialspoint.com/javafx/javafx_application.htm. [Data uzyskania dostępu: 8 Grudnia 2018].
- [8] „Understanding the JavaFX Architecture,” Oracle, [Online]. Available: <https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-architecture.htm>. [Data uzyskania dostępu: 4 Listopad 2018].
- [9] E. Syse, „TornadoFX Guide,” 2016. [Online]. Available: <https://edvin.gitbooks.io/tornadofx-guide/content/>. [Data uzyskania dostępu: 04 Listopad 2018].

- [10] M. Sawicki, „Zmiana długości cyklu izochronicznego w szeregowych interfejsach komunikacyjnych USB 2.0 I IEEE 1394A,” *Napędy i Sterowanie*, p. 101, Listopad 2013.
- [11] Urząd Komunikacji Elektronicznej, „Raport o stanie rynku telekomunikacyjnego w Polsce w 2017 r.,” Warszawa, 2017.
- [12] M. Gartkiewicz, „Usługa VoIP – czynniki wpływające na jakość transmitowanego głosu,” *Telekomunikacja i Techniki Informacyjne*, nr 1-2, 2004.
- [13] W. L. Hosch, „Encyclopedia Britannica - Niklas Zennström,” 26 Styczeń 2011. [Online]. Available: <https://www.britannica.com/biography/Niklas-Zennstrom>. [Data uzyskania dostępu: 15 Listopada 2018].
- [14] H. Schulzrinne i S. A. Baset, „An Analysis of the Skype Peer-to-Peer Internet Telephony,” New York, 2004.
- [15] Skype, „Help for Skype – user guides, FAQs, customer support,” [Online]. Available: http://support.skype.com/en_US/faq/FA153/Which-protocols-does-Skype-use. [Data uzyskania dostępu: 03 Marzec 2009].
- [16] Google Chrome, „WebRTC,” [Online]. Available: <https://webrtc.org/>. [Data uzyskania dostępu: 17 Listopad 2018].
- [17] Mozilla, „Introduction to WebRTC protocols,” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Protocols. [Data uzyskania dostępu: 17 Listopad 2018].
- [18] TrueConf LLC., „H.323 Standard,” [Online]. Available: <https://trueconf.com/standart-h323.html>. [Data uzyskania dostępu: 17 Listopad 2018].
- [19] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley i E. Schooler, „RFC 3261,” Czerwiec 2002. [Online]. Available: <https://tools.ietf.org/html/rfc3261>. [Data uzyskania dostępu: 04 Listopada 2018].
- [20] Z. Coklin, „SIP Inspector,” [Online]. Available: <http://www.sipinspector.com>. [Data uzyskania dostępu: 29 listopad 2018].

- [21] StarTrinity, „StarTrinity SIP Tester™ (call generator, simulator) - VoIP monitoring and testing tool, VoIP recorder,” [Online]. Available: <http://startrinity.com/VoIP/SipTester/SipTester.aspx>. [Data uzyskania dostępu: 29 Listopad 2018].
- [22] R. Day, „Welcome to SIPp,” [Online]. Available: <http://sipp.sourceforge.net/>. [Data uzyskania dostępu: 29 Listopad 2018].
- [23] Oracle, „Jak ręcznie pobrać i zainstalować oprogramowanie Java na komputerze z systemem Windows?,” [Online]. Available: https://www.java.com/pl/download/help/windows_manual_download.xml. [Data uzyskania dostępu: 12 Grudzień 2018].
- [24] CyderCode Team, „Kurs Maven #1 Instalacja i konfiguracja,” Youtube, 29 Kwiecień 2016. [Online]. Available: <https://www.youtube.com/watch?v=DKX9tmkxYzI>. [Data uzyskania dostępu: 12 Grudnia 2018].
- [25] W. Bot, „Asterisk 13 Configuration_res_pjsip,” [Online]. Available: https://wiki.asterisk.org/wiki/display/AST/Asterisk+13+Configuration_res_pjsip. [Data uzyskania dostępu: 6 Grudzień 2018].
- [26] F. Buschmann, K. Henney i D. C. Schmidt, w *Pattern-oriented software architecture: On patterns and pattern languages Volume 5*, Wiley, 2007, pp. 178-179.
- [27] A. Almiray, „MVC Patterns,” 15 Grudzień 2015. [Online]. Available: <http://aalmiray.github.io/griffon-patterns/>. [Data uzyskania dostępu: 8 Grudzień 2018].
- [28] A. Johnston, S. Donovan, R. Sparks, C. Cunningham i K. Summers, „Session Initiation Protocol (SIP) Basic Call Flow Examples,” Grudzień 2003. [Online]. Available: <https://tools.ietf.org/html/rfc3665>. [Data uzyskania dostępu: 02 Styczeń 2019].
- [29] A. Roach, C. Jennings, F. Firmin, J. Peterson i M. Barnes, „Session Initiation Protocol (SIP) Parameters,” IANA, 30 Listopad 2018. [Online].

Available: www.iana.org/assignments/sip-parameters/sip-parameters.xhtml.

[Data uzyskania dostępu: 9 Grudzień 2018].