

Spis treści

| | |
|--|----|
| 1. Wstęp..... | 1 |
| 1.1 Wprowadzenie..... | 1 |
| 1.2 Cel i zakres pracy..... | 2 |
| 1.3 Zawartość pracy..... | 2 |
| 2. Podstawy teoretyczne pracy..... | 4 |
| 2.1 Platforma mobilna Android..... | 4 |
| 2.2 Protokół SIP..... | 5 |
| 2.3 Protokół SDP..... | 7 |
| 2.4 Protokół RTP..... | 8 |
| 2.5 Protokół RCTP..... | 8 |
| 2.6 Parametry wpływające na jakość przekazu głosu..... | 9 |
| 2.6.1 Utrata pakietów..... | 9 |
| 2.6.2 Opóźnienie..... | 9 |
| 2.6.3 Jitter..... | 10 |
| 2.6.4 Kodowanie..... | 10 |
| 2.7 Ocena jakości połączenia VoIP..... | 11 |
| 3. Technologie i narzędzia użyte w projekcie..... | 12 |
| 3.1 Opis stanowiska badawczego..... | 12 |
| 3.2 Klient SIP (softphone)..... | 13 |
| 3.3 Serwer SIP | 17 |
| 3.4 Serwer pośredniczący..... | 18 |
| 4. Przebieg badań..... | 22 |
| 4.1 Specyfikacja wykorzystanych urządzeń..... | 22 |
| 4.2 Konfiguracja środowiska badawczego..... | 23 |
| 4.2.1 Serwer SIP..... | 23 |
| 4.2.2 Serwer pośredniczący..... | 24 |
| 4.2.3 Klient SIP..... | 27 |
| 4.3 Zestawienie i analiza wyników badań..... | 28 |
| 4.4 Wnioski..... | 36 |
| 5. Podsumowanie..... | 37 |
| Bibliografia..... | 38 |
| Spis Ilustracji..... | 39 |
| Spis Tabel..... | 40 |

1. Wstęp

1.1 Wprowadzenie

Od czasu kiedy pojawiła się technologia telefonii internetowej VoIP, jej niskie koszty wdrażania oraz dostępność sprawiły, że stała się sporą konkurencją dla tradycyjnych sieci telefonicznych.

Ponadto intensywnie rozwijające się technologie mobilnego dostępu do internetu wywołały duże zainteresowanie wśród badaczy oraz poszerzyły horyzonty zastosowań telefonii internetowej.

Co raz większa liczba pojawiających się na rynku urządzeń mobilnych, laptopów, PDA, smartfonów oraz tabletów wyposażonych jest w co najmniej jeden z interfejsów sieciowych (802.11, Bluetooth, 2G/2.5G/3G). Sprawilo to, że rozwój mobilnych rozwiązań telefonii VoIP nabrał bardzo szybkiego tempa.

Jednak mimo tego, że technologia VoIP ma już niespełna 20 lat, jej największym problemem pozostaje jakość połączenia. Obecna infrastruktura nie zawsze umożliwia zapewnienie minimalnego stałego przesyłu danych, wymaganego dla idealnej rozmowy. W razie zbyt wolnego połączenia pojawiają się zniekształcenia, dźwięk staje się mechaniczny, a słowa urywane. W związku z tym kluczowe jest poszukiwanie rozwiązań, które pozwolą wyeliminować lub przynajmniej zredukować wyżej wymienione problemy.

1.2 Cel i zakres pracy

Celem pracy jest zbadanie zastosowań protokołu SIP w mobilnej telefonii VoIP oraz doświadczalne dobranie optymalnych parametrów rozmowy VoIP dla wybranej platformy. Na potrzeby realizacji projektu należało stworzyć stanowisko laboratoryjne, które umożliwi symulację parametrów sieciowych dla różnych mediów transmisyjnych wykorzystywanych przez platformy mobilne. Ponadto stanowisko badawcze powinno umożliwiać przeprowadzanie pomiarów jakości połączenia. Dodatkowo należy wybrać jedną z otwartych implementacji oprogramowania klienta oraz serwera protokołu SIP (ang. *Session Initiation Protocol*) w celu ich modyfikacji na potrzeby eksperymentów.

Przed przystąpieniem do implementacji stanowiska należało zapoznać się z wybraną platformą mobilną oraz przeprowadzić dokładną analizę działania protokołu inicjowania sesji SIP.

1.3 Zawartość pracy

Rozdział drugi ma na celu przedstawienie teoretycznych podstaw pracy. Na początku omówione zostaną cechy wybranej platformy mobilnej. W kolejnych podrozdziałach znajduje się opis bardzo popularnego i szeroko stosowanego protokołu SIP wraz z innymi protokołami wspierającymi, które również są używane przy nawiązywaniu połączeń w telefonii VoIP. Ponadto opisane zostaną czynniki wpływające na jakość mowy oraz problemy, z którymi należy się zmierzyć implementując usługi telefonii internetowej.

Kolejny rozdział zawiera przegląd technologii użytych w trakcie realizacji tej pracy. Na początku opisana zostanie wykorzystana w projekcie aplikacja kliencka protokołu SIP. Następnie zostanie przedstawiony wybrany do realizacji projektu serwer SIP. Ostatnim opisywanym w tym rozdziale elementem będzie autorska implementacja serwera SIP Proxy, niezbędna do badań jakości rozmów VoIP.

W czwartym rozdziale przedstawiony został autorski projekt stanowiska badawczego, służącego do przeprowadzania badań jakości połączenia VoIP w zmieniających się warunkach sieciowych. W dalszej części rozdziału, krok po kroku został przedstawiony proces konfiguracji sieci, serwerów SIP i Proxy oraz klientów SIP. Po etapie przygotowywania środowiska opisane zostały parametry, które były zmieniane w trakcie przeprowadzanych eksperymentów. Dalsza część rozdziału zawiera wyniki przeprowadzonych badań wraz z ich analizą.

Ostatnią częścią pracy jest podsumowanie, w którym na bazie analizy wyników z rozdziału czwartego zostały zaproponowane optymalne parametry ustawień klienta SIP dla różnych interfejsów sieciowych platform mobilnych.

2. Podstawy teoretyczne pracy

2.1 Platforma mobilna Android

Przez ostatnich kilka lat można zauważyć gwałtowny rozwój platform mobilnych. Jednak tworzenie aplikacji na platformy mobilne niesie za sobą konieczność uwzględnienia wielu krytycznych czynników wpływających na doświadczenia użytkownika. Należą do nich między innymi:

- potrzeba zapewnienia niskiego zużycia baterii.
- aplikacja musi generować jak najniższe zużycie danych przesyłanych przez internet.
- projektując aplikację trzeba uwzględnić różne się rozmiary ekranów urządzeń mobilnych (smartphone'y , tablety itd.).
- moc obliczeniowa urządzeń przenośnych jest o wiele mniejsza niż w urządzeniach klasy PC.

Na potrzeby realizacji projektu jako platformę mobilną użyto system Android. System wybrano z uwagi na to, że autor posiada zawodowe doświadczenie w implementacji aplikacji na tę platformę. Ponadto istnieje duża ilość aplikacji klienckich SIP, których kod jest udostępniony na licencji GPL.

Android jest to system operacyjny oparty na jądrze Linux oraz oprogramowaniu na licencji GNU. Początkowo był rozwijany przez firmę Android Inc. (kupioną później przez Google), następnie przeszedł pod skrzydła Open Handset Alliance. Najważniejsze cechy systemu to pełna wielozadaniowość oraz możliwość implementacji aplikacji w językach programowania C, C++ oraz Java. Stwarza to dogodne warunki do używania gotowych i udostępnionych rozwiązań istniejących już na innych platformach. Dodatkowo firma Google zapewniła dostęp do szerokiej gamy narzędzi wspomagających tworzenie aplikacji na system Android. Warty uwagi jest zbiór skryptów NDK (ang. Native Development Kit) wspomagających kompilację środowiska w językach C/C++. Zawiera on również zbiór plików nagłówkowych do bibliotek systemowych w standardzie POSIX.

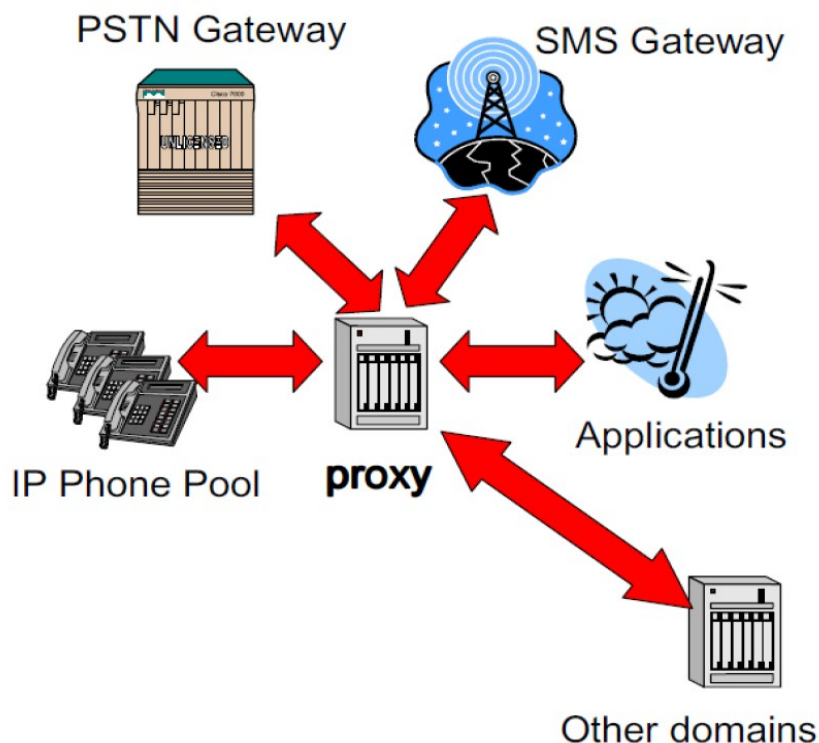
Spełnienie wymagań jakie niesie za sobą projektowanie aplikacji na platformy mobilne będzie jednym z najważniejszych czynników wpływających na dobór optymalnych parametrów dla mobilnej telefonii VoIP.

2.2 Protokół SIP

SIP (ang. Session Initiation Protocol) – protokół używany do inicjowania sesji, zaproponowany przez IETF standard dla zestawiania sesji pomiędzy jednym lub wieloma klientami. Jest obecnie dominującym protokołem sygnalizacyjnym dla Voice over IP i stopniowo zastępuje H.323.

Protokół SIP ma na celu dostarczać zestaw funkcji obsługi połączenia i innych cech obecnych w publicznej sieci telefonicznej (wybieranie numeru, dzwonek w telefonie, sygnał zajętości itp.). Jednakże ich implementacja i używana terminologia jest odmienna. SIP to protokół typu peer-to-peer. Wymaga jedynie prostej sieci szkieletowej. Inteligencja rozproszona jest na brzegach sieci – zaszyta w węzłach końcowych. Dodatkowo SIP współgra z kilkoma innymi protokołami i jest zaangażowany jedynie w część sygnalizacyjną sesji komunikacyjnej.

Protokół SIP działa na warstwie aplikacyjnej modelu OSI. Ponadto protokół SIP jest podobny do HTTP, używa zwykłego tekstu oraz oparty jest o bardzo prosty mechanizm żądanie-odpowiedź. Identyfikacja terminali końcowych odbywa się z wykorzystaniem adresów podobnych do adresów e-mail: *uzytkownik@domena:port*, domyślnym portem jest 5060.



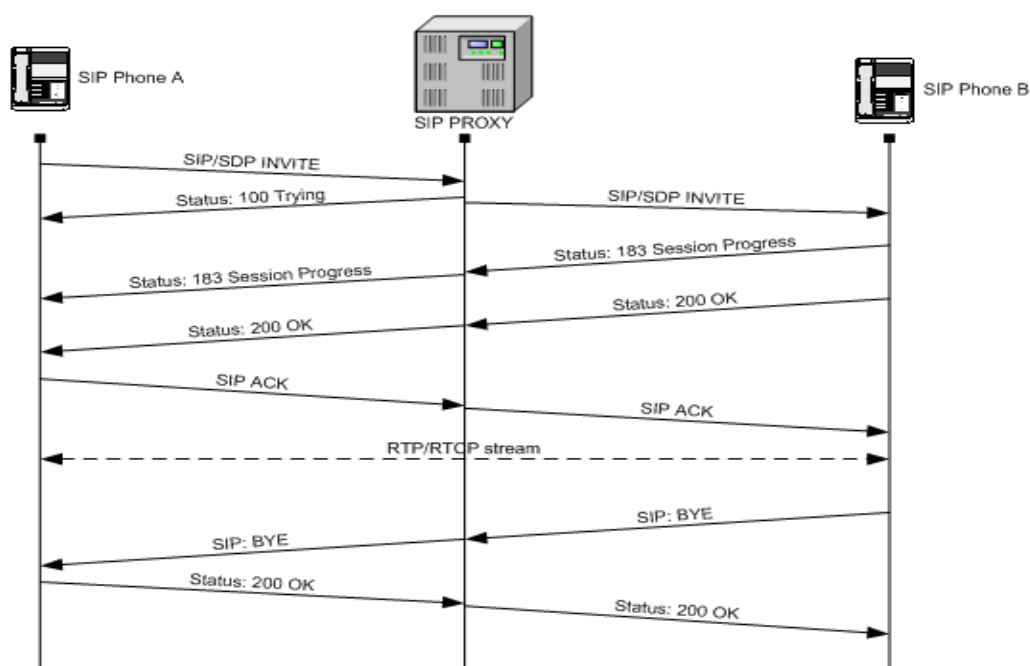
Rysunek 1: Przykładowa architektura infrastruktury opartej o protokół SIP [3].

Na rysunku nr 1 przedstawiono przykładową architekturę sieci opartej o protokół SIP, w której skład wchodzi elementy końcowe, mogą one być aplikacjami uruchamianymi na komputerze (softphone) lub urządzeniami podobnymi do telefonów.

Dodatkowo jednym z najważniejszych elementów sieci SIP jest rejestrator (registrar), który jest specjalną bazą danych, która komunikuje się z węzłami SIP w celu zbierania i archiwizacji informacji na temat użytkowników SIP. Dane te są używane w trakcie nawiązywania połączenia do odnajdywania w sieci węzłów docelowych.

Serwer proxy ma za zadanie przekazywać do odpowiedniej domeny, żądania połączenia. Proxy analizuje polecenie protokołu SIP i na podstawie zawartego w nim adresu kieruje je do innego węzła w sieci.

Na rysunku zamieszczonym poniżej przedstawiony jest proces nawiązywania sesji używając protokołu SIP. Strona inicjująca połączenie wysyła żądanie INVITE zawierające informacje m.in. o tym do kogo ma ono zostać przesłane oraz informacje o proponowanych strumieniach danych (protokół SDP). Serwer SIP proxy przesyła żądanie do drugiej strony. W odpowiedzi druga strona przesyła wiadomość potwierdzająca odebranie żądania (183 Session Progress). Jeśli użytkownik zdecyduje się odebrać połączenie, generowana jest odpowiedź 200 OK, zawierająca wybrane strumienie danych oraz informacje o kodeku, który będzie wykorzystywany w trakcie rozmowy. Po przesłaniu odpowiedzi do strony inicjującej sesję można przystąpić do przesyłania strumieni audio/video. W celu zakończenia sesji, jedna ze stron wysyła żądanie BYE.



Rysunek 2: Przykład nawiązywania rozmowy w infrastrukturze opartej o SIP [11].

2.3 Protokół SDP

Session Description Protocol jest formatem opisu parametrów do inicjacji mediów strumieniowych. Został opublikowany przez IETF jako dokument RFC 2327. Zasada działania opiera się o schemat oferta/odpowiedź. Wiadomość SDP składa się ze zbioru atrybutów zawierających opis strumieni mediów oferowanych przez jedną ze stron próbujących nawiązać komunikację wykorzystującą protokół RTP. Poniżej znajduje się przykładowa wiadomość SDP.

```
v=0
o=mhandley 2890844526 2890842807 IN IP4
  126.16.64.4
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e=mjh@isi.edu (Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 31
m=application 32416 udp wb
```

Rysunek 3: Przykładowa wiadomość SDP [3].

Analizując powyższy przykład można odczytać, że nadawcą wiadomości jest "mhandley" (parametr "o="), jego adres IPv4 to 224.2.27.12/127 (parametr "c=") oraz to, że nadawca wiadomości proponuje sesję audio, video oraz strumienia danych (application) na portach odpowiednio 49170, 51372 i 34416 (parametr "m="). Strona docelowa, która odbierze ten komunikat powinna sformułować odpowiedź wg tego samego schematu, wybierając interesujące ją strumienie mediów i odesłać ją do strony inicjującej połączenie.

Dodatkowo z podanej wyżej przykładowej wiadomości można odczytać, że strumienie mediów będą przesyłane z wykorzystaniem protokołu RTP, który zostanie opisany w kolejnym podrozdziale.

2.4 Protokół RTP

Próbki audio oraz video po kwantyzacji i kodowaniu są wysyłane jako pakiety UDP, proces ten standaryzuje protokół RTP (ang. Real-time Transport Protocol). Protokół RTP umożliwia przesyłanie danych w czasie rzeczywistym. Pakiet RTP zawiera dane o typie przesyłanych danych, numer seryjny oraz znacznik czasu. Ponadto informacje te pozwalają wykryć straty pakietów i zidentyfikować nadawcę.

| RTP packet header | | | | | | | |
|-------------------|--------------------------------------|---|---|-----|---|-------------------------|-----------------|
| bit offset | 0-1 | 2 | 3 | 4-7 | 8 | 9-15 | 16-31 |
| 0 | Version | P | X | CC | M | PT | Sequence Number |
| 32 | Timestamp | | | | | | |
| 64 | SSRC identifier | | | | | | |
| 96 | CSRC identifiers | | | | | | |
| | ... | | | | | | |
| 96+32×CC | Profile-specific extension header ID | | | | | Extension header length | |
| 128+32×CC | Extension header | | | | | | |
| | ... | | | | | | |

Rysunek 4: Struktura pakietu RTP [3].

2.5 Protokół RTCP

W celu usprawnienia oraz poprawienia działania protokołu RTP stworzono protokół RTCP (ang. *Real-time Transport Control Protocol*). Pozwala on prowadzić monitorowanie odbieranych danych, sterowanie oraz identyfikację źródła pakietów.

Poniżej wymienione zostały typy pakietów RTCP przenoszące informacje sterujące:

- SR – (ang. *Sender Report*) Raport nadawcy – transmisja i przyjmowanie statystyki od uczestników sesji – aktywnych nadawców,
- RR – (ang. *Receiver Report*) Raport odbiorcy – przyjmowanie statystyki od uczestników sesji nie będących aktywnymi nadawcami oraz jako kombinacja wraz z SR – raportowanie aktywnych nadawców przy liczbie źródeł sygnału powyżej 31,
- SDES – (ang. *Source Description*) Opis źródła – dane identyfikacyjne źródła sygnału łącznie z CNAME,
- BYE – zakończenie udziału w sesji uczestnika,

Protokół RTCP jest kluczowym narzędziem wykorzystywanym przy ocenianiu jakości połączenia, m.in. w algorytmie MOS, opisanym w dalszej części pracy.

2.6 Parametry wpływające na jakość przekazu głosu

W dziedzinie telefonii internetowej pojęcie jakości usług QoS (ang. Quality of Service) zostało zdefiniowane przez ITU (ang. International Telecommunication Union) w 1994r. Celem opracowania tej technologii było zapewnienie sieciom internetowym takiej samej jakości usług jaką posiadają sieci telefoniczne [10]. W dalszej części pracy zostaną opisane parametry wpływające na jakość połączenia w technologii VoIP.

2.6.1 Utrata pakietów

Jednym z najważniejszych czynników determinujących jakość rozmowy audio jest utrata pakietów. Określa się ją jako procentowy stosunek liczny utraconych pakietów do liczby pakietów wysłanych. Istnieje kilka przyczyn występowania tego zjawiska:

- przeciążenie łącza
- kolizje w węźle
- przekroczenie wyznaczonego czasu opóźnienia (pakiety, które zostały odebrane zbyt późno zostają odrzucone).

Istnieje kilka metod, które pozwalają rozwiązać problem utraty pakietów, jeśli straty nie są zbyt duże, tj. nie większe niż 5%. W tym celu stosuje się mechanizm PLC (ang. Packet loss concealment). Polega on na tym, że w miejscu utraconego pakietu powtórnie odgrywana jest poprzednia próbka lub luka zostaje wypełniona ciszą [7].

2.6.2 Opóźnienie

Do przesyłania dźwięku przez sieć pakietową konieczne jest jego przetworzenie do postaci cyfrowej. A następnie po stronie odbiorczej odwrotne przetworzenie sygnału cyfrowego do postaci analogowej. Na całkowite opóźnienie mają wpływ składniki wymienione poniżej:

- przygotowanie i transmisja pakietu IP
- próbkowanie i kodowanie
- dekodowanie i przetwarzanie do postaci analogowej
- eliminacja kolejności oraz opóźnienia pakietów (jitter)

Na podstawie badań stwierdzono, że opóźnienie transmisji mowy nie powinno przekraczać 150ms. Maksymalną dopuszczalną wartością jest opóźnienie nie przekraczające 250ms, natomiast wyższych wartościach rozmowa staje się uciążliwa dla uczestników i nie do zaakceptowania [9].

2.6.3 Jitter

Zjawisko Jitter'u to krótkookresowe odchylenie od ustalonych, okresowych charakterystyk sygnału. Fluktuacje strumienia pakietów znacząco wpływają na jakość połączenia w technologii VoIP. Z uwagi na zmienne warunki sieciowe pakiety mogą docierać w różnej kolejności lub o różnych opóźnieniach.

W celu zmniejszenia wpływu tego zjawiska na jakość rozmowy, stosuje się tzw. bufor fluktuacji (ang. Jitter Buffer). Na podstawie znaczników pakietów RTP wykonywane jest sortowanie pakietów wg kolejności ich nadania. Bufor posiada również swój maksymalny czas oczekiwania, pakiet przychodzący, który przekroczy ten czas zostanie odrzucony.

Wartość fluktuacji rzędu 40ms nie wpływają na jakość rozmowy, fluktuacja nie przekraczająca 75ms pozwala zachować dobrą jakość transmisji. Natomiast jitter przekraczający 75ms jest nie do zaakceptowania [10].

2.6.4 Kodowanie

Kodeki audio są stosowane w celu zmniejszenia zużycia łącza danych poprzez stosowanie skomplikowanych algorytmów. Poziom kompresji używany przez kodek wpływa na jakość transmitowanego głosu. Oznacza to, że wyższa jakość mowy wymusza większe zużycie łącza [11]. Kolejnym ważnym parametrem jest czas kompresji próbki mowy, czas ten różni się w zależności od szerokości pasma wykorzystanego do próbkowania sygnału, stopnia kompresji oraz skomplikowania zastosowanego kodeka. Stopień kompresji niesie za sobą również ryzyko dużego spadku jakości rozmowy w połączeniu ze zjawiskiem utraty pakietów. W tabeli nr 1 znajdują się wybrane parametry najczęściej stosowanych kodeków audio wraz ze współczynnikiem określającym jakość MOS, który zostanie opisany w kolejnym podrozdziale.

Tabela 1: Parametry najczęściej używanych kodeków audio w technologii VoIP [11].

| Kodek | Algorytm | Bitrate [kbps] | MOS |
|---------|----------|----------------|------|
| G.711 | PCM | 64 | 4,1 |
| G.726 | ADPCM | 32 | 3,85 |
| G.729 | CS-ACELP | 8 | 3,92 |
| G.723.1 | ACELP | 5,3 | 3,56 |

2.7 Ocena jakości połączenia VoIP

W trakcie realizacji projektu do oceny jakości rozmowy VoIP wykorzystano algorytm estymacji parametru MOS(ang. Mean Opinion Score). Takie rozwiązanie zostało opisane w [12], pozwala ono na implementacje uproszczonego modelu służącego do przeprowadzania pomiarów na warstwie transportowej.

Do zastosowania algorytmu oceny jakości potrzebne jest spełnienie kilku warunków:

- poprawna implementacja obsługi protokołu RTCP w implementacjach aplikacji klienckich SIP
- pomiar jest wykonywany co 5s.
- zbiór kodeków zbadanych w pracy [6] będzie rozszerzony o kodeki: G722, SILK, Speex oraz iLBC.
- wyniki pomiarów będą zapisywane do pliku dla każdej ze stron transmisji audio.

Algorytm obliczający parametr MOS został udostępniony autorowi dzięki uprzejmości firmy VoipSwitch. Firma od początku swojego istnienia wdrożyła ok. 15 tysięcy systemów VoIP własnej produkcji opartych o protokół SIP. Autor pracy uczestniczył w implementacji oferowanych przez firmę VoipSwitch mobilnych rozwiązań VoIP na platformie Android.

Funkcja obliczająca parametr MOS przyjmuje 3 parametry:

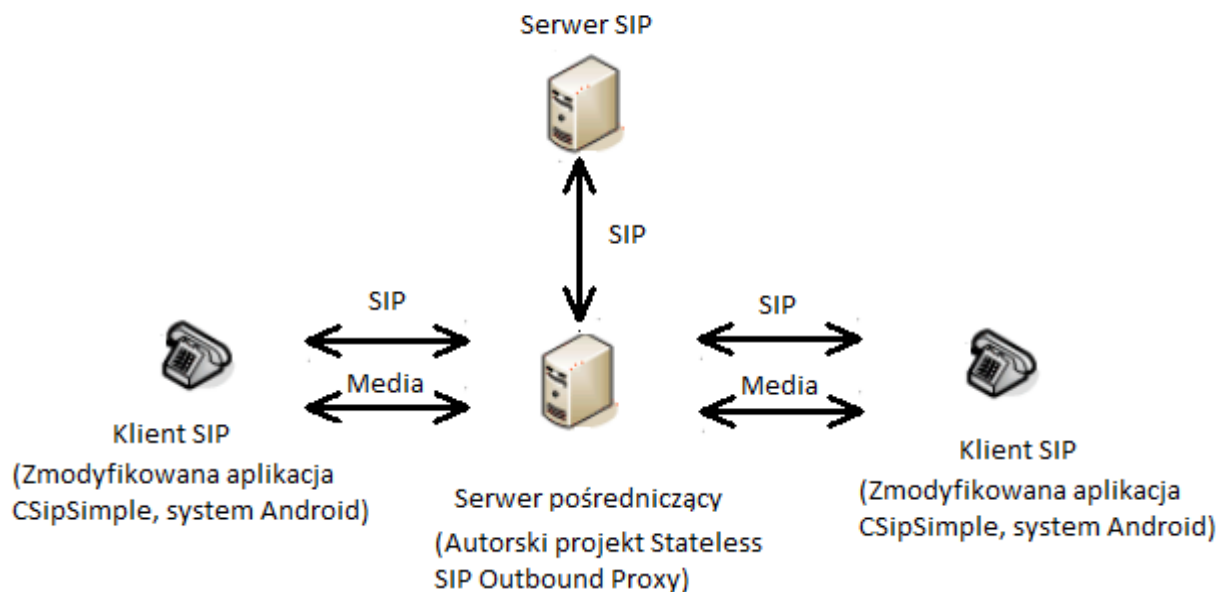
1. avg_loss_rate [%] - średni procent strat pakietów od ostatniego pomiaru.
2. rtt [ms] – round trip time obliczany na podstawie zawartości raportów RTCP
3. jitter [ms] – wielkość bufora fluktuacyjnego w chwili pomiaru

Dokładny opis implementacji algorytmu MOS wraz z opisem jego integracji z wybraną aplikacją kliencką SIP został zawarty w rozdziale trzecim.

3. Technologie i narzędzia użyte w projekcie

3.1 Opis stanowiska badawczego

W celu przeprowadzenia badań jakości połączenia VoIP w różnych warunkach sieciowych należało zaprojektować oraz zaimplementować środowisko badawcze. Powinno ono się składać z co najmniej dwóch urządzeń wyposażonych w aplikację kliencką SIP (tzw. softphone), działających na platformie Android oraz centralnego serwera SIP. Dodatkowo w celu zapewnienia możliwości manipulacji strumieniami przesyłanych danych należało zaimplementować serwer pośredniczący, który będzie filtrować oraz edytować przepływające przez niego wiadomości SIP. Ponadto serwer pośredniczący powinien mieć opcje zmiany parametrów manipulacji strumieniami przekazywanych danych oraz informować użytkownika o zmieniających się warunkach w infrastrukturze SIP (wykrywanie nowych połączeń, logowanie wprowadzanych w pakietach zmian). Na rysunku nr 5 znajduje się schemat architektury sieci zaprojektowanej do badań. W dalszej części rozdziału dokładnie opisane zostaną poszczególne elementy stanowiska badawczego.



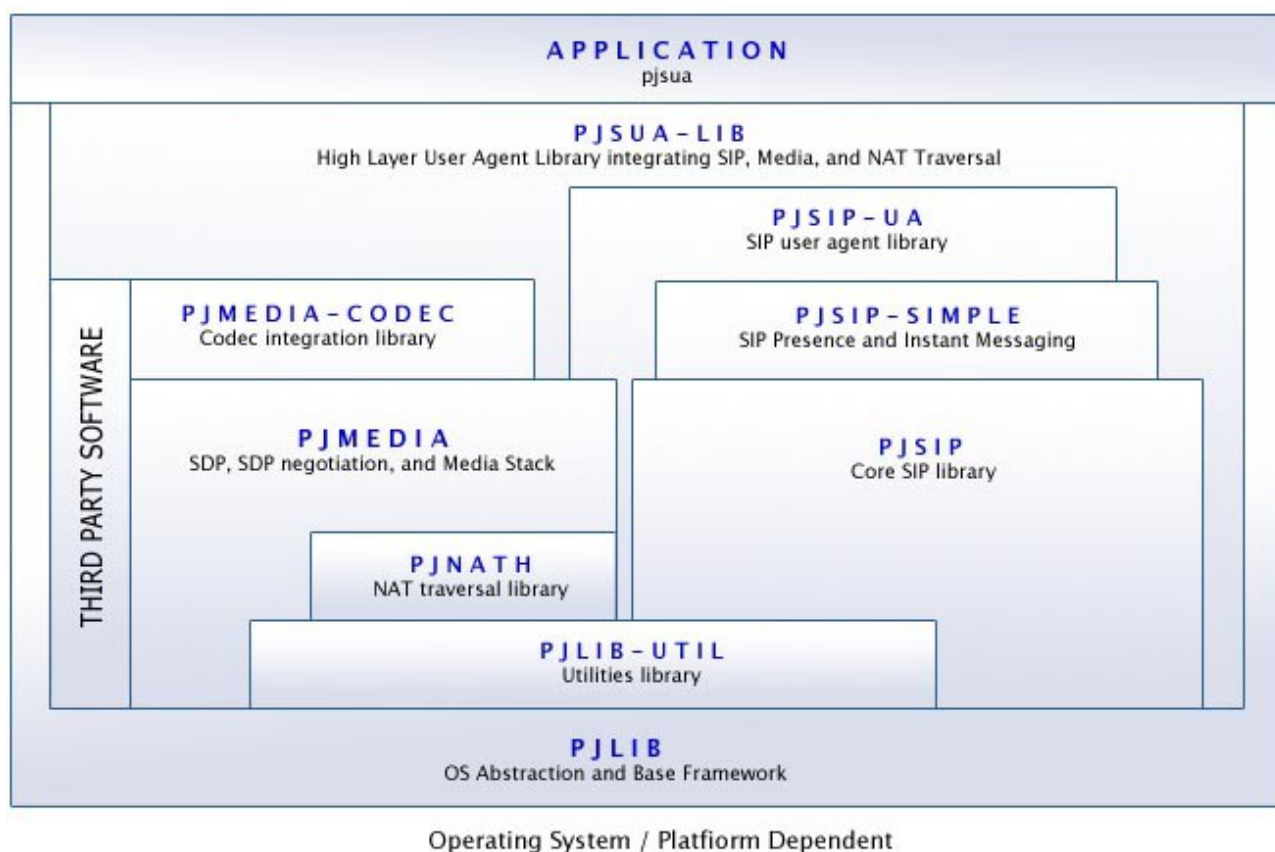
Rysunek 5: Architektura stanowiska badawczego.

3.2 Klient SIP (softphone)

W trakcie poszukiwań odpowiedniego klienta SIP, który spełniałby wszystkie wymagania dostarczonego przez firmę VoipSwitch algorytmu estymacji parametru oceny jakości MOS, natrafiono na wielo-platformową bibliotekę PJSIP.

Jest to darmowa i otwarta biblioteka implementująca większość funkcjonalności aplikacji klienckich SIP. Została stworzona przez firmę Teluu Ltd. i udostępnioną na licencji GPLv2.

Liczba zaimplementowanych funkcjonalności protokołu SIP jest przytłaczająca, skupiono się więc głównie na części biblioteki odpowiedzialnej za manipulację oraz przesyłanie mediów. Na rysunku nr 6 znajduje się schemat architektury biblioteki PJSIP.



Rysunek 6: Architektura biblioteki PJSIP [7].

Implementacja funkcjonalności pozwalającej na mierzenie parametru MOS, wymagała modyfikacji modułu PJMEDIA. W tym celu zmodyfikowano struktury odpowiedzialne za zarządzanie strumieniami mediów oraz rozszerzono dostępną funkcjonalność. Dodatkowo należało zmodyfikować warstwę aplikacji, aby udostępnić użytkownikowi nowe funkcje.

Kolejnym etapem przygotowania aplikacji klienckiej było skompilowanie biblioteki PJSIP na platformę Android. W trakcie studiowania platformy natrafiono na aplikację CSipSimple. Jest to darmowa oraz otwarta wersja klienta SIP na system Android oparta o bibliotekę PJSIP stworzona przez Régisa Montoye [15]. Aplikacja ta dodatkowo rozszerza możliwości biblioteki PJSIP o większą ilość kodeków audio oraz interfejs graficzny. Ponadto jest napisana w większości w języku Java. Natomiast część natywna (biblioteka PJSIP i kodeki audio/video) jest napisana w językach C/C++ i zintegrowana z częścią napisaną w języku Java z wykorzystaniem technologii JNI (ang. Java Native Interface). Na rysunku nr 7 znajduje się zrzut ekranu w trakcie rozmowy video oraz z ustawień aplikacji CSipSimple.



Rysunek 7: Interfejs aplikacji CSipSimple [8].

Biblioteka PJSIP posiada własną implementację adaptacyjnego bufora fluktuacji (ang. Jitter Buffer). Bufor może być konfigurowany przez zmianę parametrów oraz algorytmu działania.

Dostępne algorytmy:

- *PJMEDIA_JB_DISCARD_NONE* – bufor nie będzie odrzucał żadnej ramki z wyjątkiem sytuacji gdy bufor jest już pełny. W takiej sytuacji jedna ramka będzie porzucona w celu zapewnienia miejsca dla nowej ramki.
- *PJMEDIA_JB_DISCARD_STATIC* – odrzucane będą ramki których opóźnienie jest większe niż 200ms. Jeśli bufor jest już pełny jedna ramka będzie porzucona w celu zapewnienia miejsca dla nowej ramki.
- *PJMEDIA_JB_DISCARD_PROGRESSIVE* – ilość odrzucanych ramek będzie obliczana na podstawie parametrów takich jak jitter oraz opóźnienie. Jeśli bufor jest już pełny jedna ramka będzie porzucona w celu zapewnienia miejsca dla nowej ramki.

Parametry bufora fluktuacji:

- *jb_init* - początkowa wielkość bufora [ms].
- *jb_min_pre* - minimalna wielkość bufora [ms].
- *jb_max_pre* – maksymalna wielkość bufora [ms].
- *jb_max* – maksymalne opóźnienie pakietów które nie będą odrzucone [ms].

Opisane wyżej parametry będą wykorzystane i zmieniane w trakcie badań w celu dobrania optymalnych warunków dla różnych parametrów sieciowych. W tym celu należało zaimplementować możliwość ich zmiany z poziomu interfejsu użytkownika aplikacji CSipSimple.

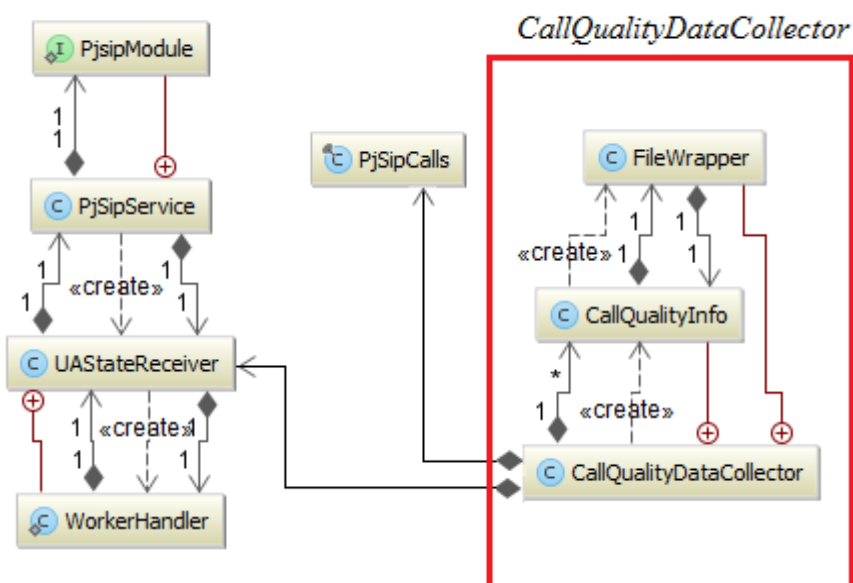
Biblioteka PJSIP posiada również funkcjonalność, która pozwala po zakończonej rozmowie pobrać statystyki strumienia danych, takie jak:

- średnie zużycie łącza danych
- straty pakietów
- użyty kodek audio
- jitter

Statystyki te będą zapisywane po zakończeniu każdego eksperymentu i poddawane analizie.

Po zapoznaniu się z kodem źródłowym biblioteki PJSIP oraz aplikacji CSipSimple, można było przystąpić do implementacji modułu odpowiedzialnego za akwizycję danych w trakcie eksperymentu. Wykorzystując istniejące już interfejsy aplikacji stworzono moduł *CallQualityDataCollector*, który wykrywał nawiązywanie połączenia z aplikacji klienckiej oraz pobierał wartość parametru jakości połączenia MOS co 5 sekund. Następnie po zakończonej rozmowie wyniki były zapisywane do pliku na karcie SD urządzenia. Moduł został zaimplementowany w języku Java oraz wykorzystywał funkcjonalności dodane do biblioteki PJSIP.

Na rysunki nr 8 przedstawiony został diagram klas modułu pomiarowego oraz jego relacje z klasami wchodzącymi w skład klienta CSipSimple. Klasa *CallQualityDataCollector* jest obserwatorem klasy *UASStateReceiver* i reaguje na wszystkie zmiany stanów rozmów w aplikacji klienckiej. Klasa pomocnicza *CallQualityInfo* przechowuje informacje o aktualnie prowadzonej rozmowie VoIP. Ponadto posiada uchwyt do klasy *FileWrapper*, która enkapsuluje dostęp do pliku, w którym są gromadzone wyniki badań. Klasa *PjSipCalls* udostępnia API pozwalające na pobieranie aktualnych wartości parametrów rozmowy z biblioteki PJSIP.



Rysunek 8: Diagram klas modułu służącego do akwizycji danych pomiarowych (klient SIP).

3.3 Serwer SIP

Do pełnienia roli serwera SIP wybrano otwartą i darmową dystrybucję firmy Digium, Asterisk. Serwer jest udostępniony na licencji GPL.

Cechy serwera Asterisk:

- działa pod kontrolą systemów operacyjnych MS Windows, Linux, BSD i Mac OS X
- obsługa protokołów SIP, IAX, H.323, ADSI, MGCP, SCCP
- obsługa kart produkowanych przez firmę Digium pracujących w sieciach PSTN i ISDN
- obsługa kart rozszerzeń innych producentów np. Junghanns.NET (sloty PCI)
- telekonferencje
- poczta głosowa
- obsługa IVR
- obsługa kolejkowania rozmów [14].

Na rysunku nr 9 została przedstawiona typowa architektura sieci SIP opartej o serwer Asterisk. W trakcie realizacji projektu serwer SIP będzie skompilowany i uruchomiony na systemie Ubuntu 12.10. Konfiguracja serwera na potrzeby badań będzie opisana w rozdziale czwartym.



Rysunek 9: Architektura sieci VoIP opartej o serwer Asterisk [14].

3.4 Serwer pośredniczący

Z uwagi na brak istniejących rozwiązań spełniających wymogi założeń projektowych, należało zaimplementować własne rozwiązanie serwera pośredniczącego. W tym celu została zaimplementowana aplikacja posiadająca funkcję Stateless Outbound SIP Proxy [5], tzn serwera przekierowującego pakiety SIP do odpowiedniego celu. Dodatkowo aplikacja powinna mieć możliwość pełnienia roli serwera pośredniczącego w wymianie mediów. Ponadto aplikacja powinna udostępniać konfigurowalną funkcjonalność manipulacji strumieniami danych.

Według RFC 3261 Stateless Outbound SIP Proxy powinno modyfikować wiadomości SIP jedynie manipulując nagłówkiem *Via*, który niesie informacje o adresie na którym nadawca spodziewa się otrzymania odpowiedzi. Jednak w związku z tym że protokół SIP jest protokołem peer2peer, należało również dodać do aplikacji funkcję serwera Registrar. Serwer registrar śledzi lokalizację zarejestrowanych klientów SIP, dzięki temu jest możliwe przesyłanie wiadomości między dwoma końcami sieci. Rozwiązanie to wymagało modyfikacji nagłówka *Contact* oraz dodanie funkcji śledzenia położenia zarejestrowanych klientów w implementacji serwera pośredniczącego. Niżej znajduje się przykładowa wiadomość SIP wraz z zaznaczonymi nagłówkami modyfikowanymi przez aplikację.

REGISTER sip:192.168.0.108 SIP/2.0

Via: SIP/2.0/UDP **ADRES_PROXY:PORT**;rport;branch=.TU.kbg

Via: SIP/2.0/UDP 192.168.0.100:39597;rport;branch=z9hG4bKPjngvSfC.TU.kbg-DfWGzEVEOZzfnQwTdM

Route: <sip:**ADRES_PROXY:PORT**;lr>

Max-Forwards: 70

From: <sip:1005@192.168.0.108>;tag=sGY2prg4ZZW5aZdy6HECAXxooXq-Xk9l

To: <sip:1005@192.168.0.108>

Call-ID: XWlZPrLKJA0hANP.kVHLj2JgHYomxYxj

CSeq: 43495 REGISTER

User-Agent: CSipSimple_GT-I9100-10/r2225

Contact: <sip:1005@**ADRES_PROXY:PORT**;ob>

Expires: 900

Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, INFO, SUBSCRIBE, NOTIFY, REFER, MESSAGE, OPTIONS

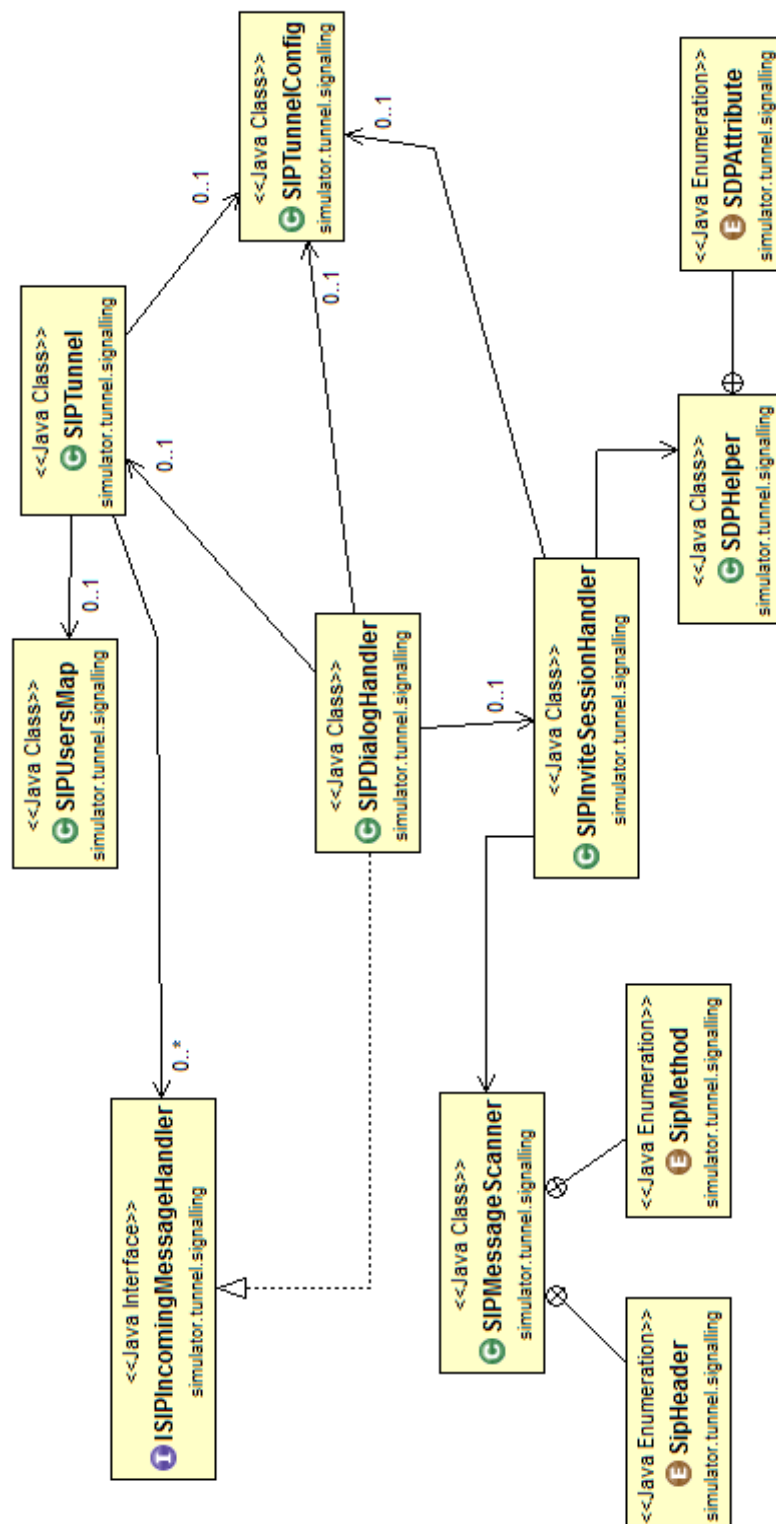
Content-Length: 0

Tak zmodyfikowana wiadomość przesyłana jest do serwera SIP. Odpowiedź otrzymana od serwera również podlega edycji, usuwany jest nagłówek *Via* wskazujący na adres serwera pośredniczącego. Następnie wiadomość przekazywana jest do klienta. Oczywiście aby takie rozwiązanie zadziałało klient SIP musi być skonfigurowany wraz z adresem Outbound Proxy.

Aplikacja została skonstruowana tak, aby była w stanie wykryć nową transakcję SIP i stworzyć dla niej nowy, odpowiadający jej obiekt, który będzie się zajmował filtracją i modyfikacją wiadomości przesyłanych między klientem a serwerem. Mechanizm ten pozwala na wykrycie nowej sesji INVITE inicjującej połączenie głosowe. Jeśli nowa sesja zostaje wykryta, wiadomość SDP nadana przez klienta zostaje zmieniona tak aby pakiety audio były przesyłane do media proxy zamiast bezpośrednio do klienta. Odpowiedź drugiej strony podlega analogicznej edycji. Każda wykrywa zmiana w wiadomości SDP jest rejestrowana w centralnym module zarządzającym strumieniami. Jeśli nowa rozmowa zostanie wykryta, otwierane jest nowe gniazdo na którym, nowo stworzona kolejka wejścia/wyjścia nasłuchuje na pakiety. Odebrane pakiety przekazywane są warstwie wyżej, która zajmuje się manipulacją strumieniem danych. Po zakończonych zmianach strumień przysyłany jest do drugiej strony połączenia. Niżej znajduje się przykładowa wiadomość SDP z zaznaczonymi zmianami wprowadzonymi przez zaimplementowane SIP Proxy.

```
v=0
o=- 3581249636 3581249636 IN IP4 ADRES_PROXY
c=IN IP4 ADRES_PROXY
t=0 0
m=audio POXY_PORT_RTP RTP/AVP 0 101
c=IN IP4 ADRES_PROXY
a=rtcp: POXY_PORT_RTCP IN IP4 ADRES_PROXY
a=sendrecv
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

Na rysunku nr 10 znajduje się diagram klas modułu odpowiedzialnego za filtrację i modyfikację wiadomości SIP. Natomiast na rysunku nr 11 przedstawiony został diagram klas modułu służącego do manipulacji strumieniami mediów w trakcie rozmowy VoIP.



Rysunek 10: Diagram klas serwera pośredniczącego – moduł SIP.

4. Przebieg badań

4.1 Specyfikacja wykorzystanych urządzeń

Zgodnie z rysunkiem nr 5 w skład stanowiska badawczego wchodzi cztery węzły, poniżej zostaną opisane parametry sprzętowe oraz wersja oprogramowania urządzeń wykorzystanych w trakcie przeprowadzania badań.

a) Serwer SIP:

System operacyjny: Ubuntu 12.10 (32 bit)

CPU: Intel Core Duo 2 x 1,7GHz

RAM: 2 GB

Połączenie: Ethernet 100 Mbps

Oprogramowanie: Serwer Asterisk w wersji 11.4.0

b) Serwer pośredniczący:

System operacyjny: Windows 7 (64 bit)

CPU: AMD FX-6100 6 x 4GHz

RAM: 8 GB

Połączenie: Ethernet 100 Mbps

Oprogramowanie: Autorska implementacja serwera pośredniczącego

c) Klienci SIP

1) Samsung Galaxy S2

System operacyjny: Android 4.1.2

CPU: Dual-core 1.2 GHz Cortex-A9

RAM: 1 GB

Połączenie: WiFi/3G

Oprogramowanie: Zmodyfikowany na potrzeby badań klient CSipSimple

2) ZTE Blade

System operacyjny: Android 4.2.1

CPU: 600 MHz ARM 11

RAM: 512 MB

Połączenie: WiFi/3G

Oprogramowanie: Zmodyfikowany na potrzeby badań klient CSpSimple

4.2 Konfiguracja środowiska badawczego.

4.2.1 Serwer SIP

W celu skonfigurowania serwera Asterisk należy odpowiednio zmodyfikować pliki konfiguracyjne. Pozwala to na stworzenie kont dla użytkowników oraz nadanie reguł dla rozmów VoIP. Na potrzeby badań dodano 2 konta, o nazwach 1004 oraz 1005. Na listingu zamieszczonym poniżej zawarty został zbiór parametrów opisujących jedno z kont. Aby dodać konto w serwerze Asterisk należy zmodyfikować plik **/etc/asterisk/sip.conf**, a następnie zrestartować usługę.

```
[1005]
type=friend
username=1005
secret=1005
context=users (kontekst dla reguł definiujących sposób nawiązywania połączeń)
directmedia=yes (ustawienie tej opcji umożliwia komunikację peer2peer między klientami)
allow=all (nadanie praw do używania dowolnego kodeka audio)
mohinterpret=none (Wyłączenie funkcji music on hold)
mohsuggest=none (Wyłączenie funkcji music on hold)
```

Aby umożliwić użytkownikom nawiązanie rozmowy należy również dodać odpowiednie reguły do pliku **/etc/asterisk/extensions.conf**.

```
[users]
exten => 1004,1,Dial(SIP/1004) (pozwala dodzwonić się do użytkownika 1004 na numerze 1004)
exten => 1005,1,Dial(SIP/1005) (pozwala dodzwonić się do użytkownika 1005 na numerze 1005)
```

Dodatkowo należy zdefiniować sposób działania serwera, na potrzeby badań do pliku konfiguracyjnego **/etc/asterisk/sip.conf** wprowadzono kilka modyfikacji:

```
[general]

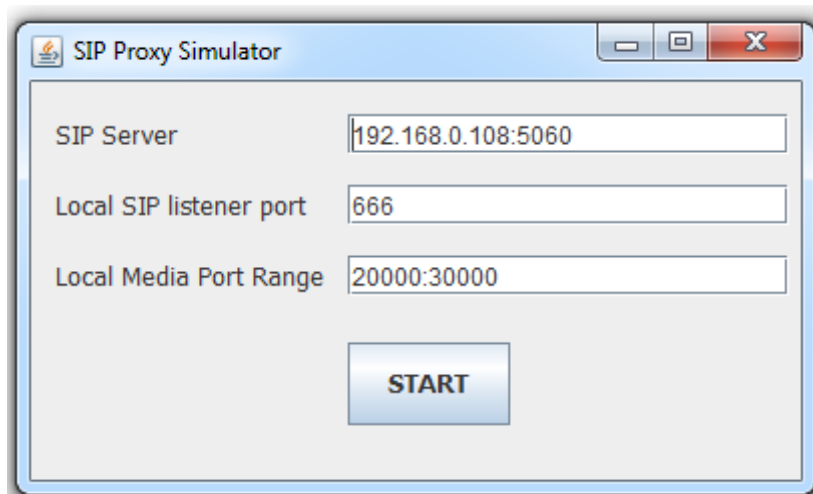
directmedia=yes   (globalne wyłączenie funkcji media proxy)
dtmfmode = info   (ustawienie metody SIP INFO dla przesyłania tonów DTMF)

outboundproxy=ADRES_I_PORT_SERWERA_PROXY
```

Najbardziej istotny jest parametr „outboundproxy”, umożliwia on uruchomienie serwera w trybie współpracy z Outbound Proxy, w tym przypadku jest nim serwer pośredniczący zaimplementowany na potrzeby badań. Po opisanym wyżej procesie konfiguracji serwer SIP jest gotowy do przeprowadzania eksperymentów.

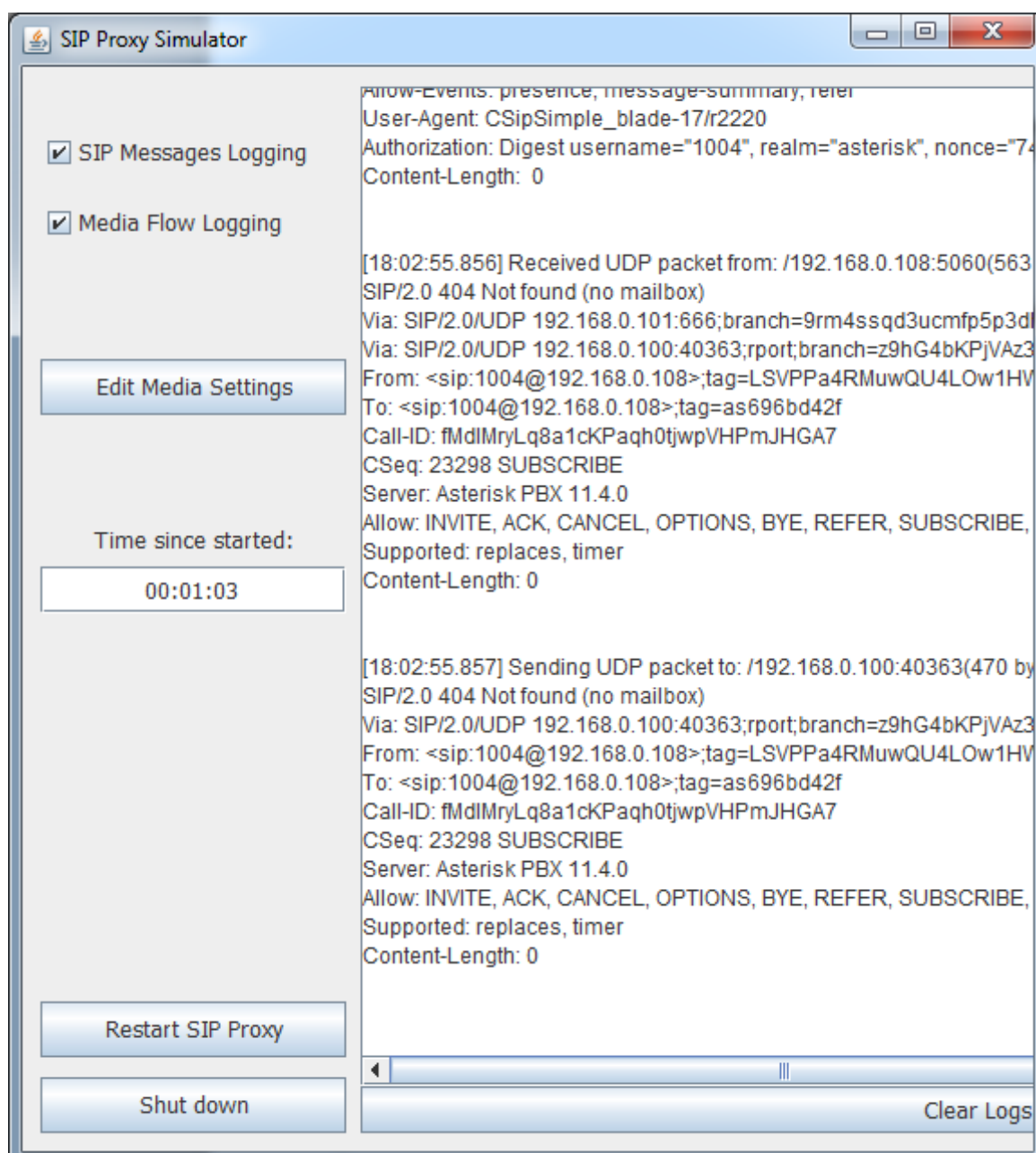
4.2.2 Serwer pośredniczący

Po uruchomieniu aplikacji do symulacji warunków sieciowych należy podać adres i port serwera SIP oraz lokalny port, na którym proxy będzie oczekiwać na wiadomości SIP. Dodatkowo należy podać zakres puli portów, z której rezerwowane będą nowe gniazda na potrzeby manipulacji strumieniami danych między rozmówcami VoIP.



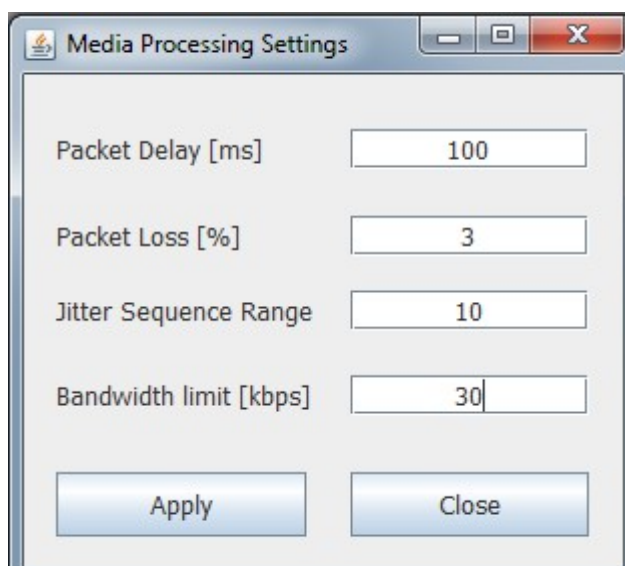
Rysunek 12: Startowe okno konfiguracyjne serwera pośredniczącego.

Po wprowadzeniu niezbędnych danych można uruchomić aplikację. Na rysunku nr 13 znajduje się zrzut ekranu z okna działającej aplikacji. Opcja "SIP Messages Logging" włącza lub wyłącza logowanie przychodzących i wychodzących wiadomości protokołu SIP. Pozwala to śledzić ruch między serwerem i klientami oraz monitorować zmiany wprowadzane w wiadomościach przez serwer proxy. Natomiast opcja "Media Flow Logging" pozwala na uruchomienie funkcji śledzenia przepływu strumieni danych. Jeśli zostanie wykryta nowa sesja INVITE otwarte zostanie oddzielne okno Loggera, w którym logowane będą najważniejsze informacje o połączeniu.



Rysunek 13: Interfejs graficzny serwera pośredniczącego.

Najważniejszą funkcją serwera pośredniczącego jest możliwość manipulacji strumieniami danych. W celu konfiguracji zmian, które będą przeprowadzane w trakcie przepływu danych przez serwer należy kliknąć na przycisk „Edit Media Settings”. Na rysunku nr 14 przedstawiony jest zrzut ekranu z widoku okna służącego do aplikowania opcji konfiguracyjnych.



Rysunek 14: Okno ustawień parametrów sieciowych serwera pośredniczącego.

Opcja „Packet Delay” pozwala na ustawienie opóźnienia jakie będzie nakładane na każdy otrzymany pakiet przed przesłaniem go drugiej stronie rozmowy audio.

Opcja „Packet Loss” pozwala na ustawienie procentowego udziału pakietów odrzucanych przez serwer względem liczby pakietów przesłanych drugiej stronie rozmowy.

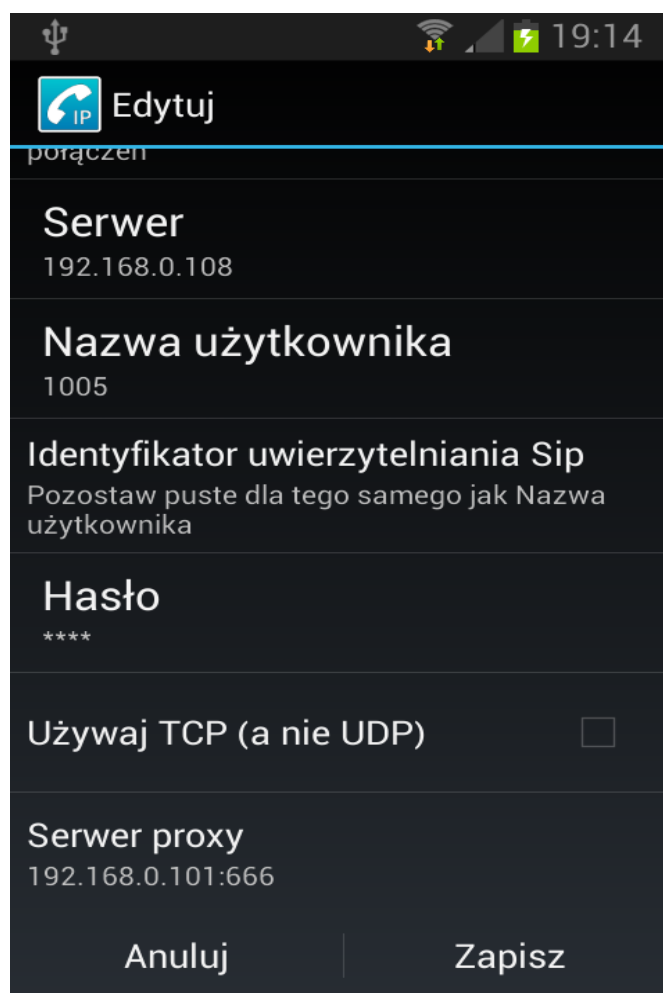
Opcja „Jitter Sequence Range” pozwala na skonfigurowanie emulowanego zjawiska jitter'u pakietów audio. Zastosowany algorytm symulacji wprowadza opóźnienie zależne od wartości parametru N. Spowodowane jest to tym, że losowa zmiana sekwencji pakietów wymaga wcześniejszego odebrania liczby N pakietów.

Opcja „Bandwidth Limit” pozwala na ustawienie limitu przesyłanych danych, umożliwia to symulację zmiennych warunków sieciowych.

Po wprowadzeniu zmian, należy kliknąć przycisk „Apply”, wprowadzona konfiguracja będzie obowiązywać w trakcie następnego nawiązanego połączenia VoIP.

4.2.3 Klient SIP

Wykorzystanym do badań klientem protokołu SIP, została zmodyfikowana na potrzeby badań aplikacja CSipSimple. Do kompilacji użyto zestawu udostępnionych przez firmę Google narzędzi developerskich platformy Android. Następnie wykorzystując narzędzie diagnostyczne ADB można było zainstalować aplikację kliencką na urządzeniu. Po uruchomieniu aplikacji niezbędne było skonfigurowanie nowego konta użytkownika. Na rysunku nr 15 przedstawiony jest zrzut ekranu z okna konfiguracyjnego aplikacji CSipSimple. W trakcie dodawania konta należało użyć loginu i hasła jednego z wcześniej stworzonych na serwerze kont SIP. Najważniejszym parametrem ustawień konta jest pole „Serwer proxy”. Podanie w tej sekcji adresu oraz portu serwera pośredniczącego zapewni odpowiednie przekierowanie wiadomości SIP.

The image is a screenshot of an Android application interface for configuring a SIP account. At the top, there is a status bar with icons for USB, Wi-Fi, signal strength, battery, and the time 19:14. Below the status bar is a header with a blue icon and the text "Edytuj". The main content area is divided into several sections: "Serwer" with the value "192.168.0.108", "Nazwa użytkownika" with the value "1005", "Identyfikator uwierzytelniania Sip" with the instruction "Pozostaw puste dla tego samego jak Nazwa użytkownika", "Hasło" with four asterisks "****", "Używaj TCP (a nie UDP)" with an unchecked checkbox, and "Serwer proxy" with the value "192.168.0.101:666". At the bottom, there are two buttons: "Anuluj" and "Zapisz".

| |
|--|
| Edytuj |
| porączeń |
| Serwer 192.168.0.108 |
| Nazwa użytkownika 1005 |
| Identyfikator uwierzytelniania Sip Pozostaw puste dla tego samego jak Nazwa użytkownika |
| Hasło **** |
| Używaj TCP (a nie UDP) <input type="checkbox"/> |
| Serwer proxy 192.168.0.101:666 |
| Anuluj Zapisz |

Rysunek 15: Opcje konfiguracji konta użytkownika w aplikacji CSipSimple.

Po zakończeniu procesu konfiguracji stanowiska badawczego można było przystąpić do przeprowadzania eksperymentów i zbierania wyników.

4.3 Zestawienie i analiza wyników badań

Pierwszym przeprowadzonym eksperymentem było sprawdzenie zużycia procesora w zależności od zastosowanej częstotliwości próbkowania audio na mobilnych urządzeniach wyposażonych w system Android. Pomiaru te są istotne z uwagi potrzebę optymalizacji zużycia baterii, czynnika bardzo istotnego z punktu widzenia projektowania aplikacji przeznaczonych na platformy mobilne. Pomiaru zostały przeprowadzone z wykorzystaniem kodeka audio G.711 ulaw. Do akwizycji danych wykorzystano aplikację ADB wchodzącą w skład zestawu narzędzi developerskich dla platformy Android. ADB pozwala na połączenie się z powłoką Shell urządzenia i wywoływanie poleceń systemowych. W ten sposób udało się uruchomić narzędzie systemowe „top”. Pozwala ono na śledzenie zużycia procesora w czasie rzeczywistym.

Tabela 2: Wpływ częstotliwości próbkowania audio na zużycie procesora.

| Częstotliwość próbkowania [kHz] | Kodek | Zużycie CPU [%] | |
|---------------------------------|--------|-------------------|-----------|
| | | Samsung Galaxy S2 | ZTE Blade |
| 8 | G.711u | 9,00% | 20,00% |
| 16 | G.711u | 14,00% | 26,00% |
| 32 | G.711u | 18,00% | 31,00% |
| 44,1 | G.711u | 21,00% | 34,00% |

Z analizy zebranych pomiarów wynika, że częstotliwość próbkowania sygnału audio ma liniowy wpływ na zużycie jednostki CPU. Dodatkowo wg subiektywnej oceny autora częstotliwość 16kHz cechuje się o wiele lepszą odczuwalną jakością mowy w porównaniu do 8kHz, słyszany głos jest zdecydowanie o wiele mniej metaliczny. Mimo tego, że kodek G711u po zdekodowaniu strumienia na wyjściu zwraca próbki o częstotliwości 8kHz, moduł *resampler* używany w bibliotece PJSIP umożliwia upsampling częstotliwości sygnału wyjściowego do 16kHz. Z punktu widzenia niesionej przez strumień informacji jest to proces wydawało by się zbędny i niepotrzebnie obciążający procesor. Jednak w związku z tym, że celem pracy jest dobranie optymalnych warunków zapewniających wysoką jakość rozmowy do dalszych badań zostanie wykorzystana częstotliwość próbkowania 16kHz.

Kolejnym eksperymentem jest zbadanie wpływu użytego kodeka na zużycie procesora klienta SIP. W trakcie badań wykonano przegląd znaczącej liczby kodeków. Skupiono się głównie na kodekach, które na wejściu i wyjściu używają pasma audio o szerokości 16kHz. Ma to na celu dobranie najlepszego kodeka względem wybranej w poprzednim eksperymencie częstotliwości próbkowania.

Tabela 3: Wpływ kodeka audio na zużycie procesora.

| Kodek | Częstotliwość we/wyj kodeka [kHz] | Zużycie CPU [%] | |
|--------|--------------------------------------|-------------------|-----------|
| | | Samsung Galaxy S2 | ZTE Blade |
| G.711u | 8 | 14,00% | 26,00% |
| G.711a | 8 | 15,00% | 25,00% |
| Speex | 8 | 22,00% | 33,00% |
| Speex | 16 | 29,00% | 42,00% |
| GSM.FR | 8 | 22,00% | 31,00% |
| G722 | 16 | 24,00% | 31,00% |
| iLBC | 8 | 35,00% | 49,00% |
| SILK | 8 | 21,00% | 48,00% |
| SILK | 16 | 28,00% | 59,00% |

Z analizy zebranych pomiarów wynika, że najlepszymi parametrami cechuje się kodek G722. Natomiast kodek audio SILK jest ciekawym przykładem jak architektura CPU wpływa na zużycie zasobów na platformach opartych o architekturę ARM. Procesor urządzenia Samsung Galaxy S2 jest wyposażony w jednostkę SIMD (NEON), która umożliwia przeprowadzanie w jednym cyklu kilku operacji na strumieniu danych, co znacząco poprawia wydajność. Natomiast urządzenie ZTE Blade jest wyposażone jedynie w jednostkę wektoryzującą VFP do obliczeń zmiennoprzecinkowych, co w efekcie powoduje o wiele słabsza wydajność.

W Tabeli nr 3 zostały przedstawione wyniki eksperymentu polegającego na porównaniu kodeków audio względem zużycia łącza danych oraz ocenie jakości z wykorzystaniem estymacji parametru MOS. W trakcie eksperymentu wykorzystano stworzone stanowisko badawcze. Analiza wyników będzie uzupełnieniem poprzedniego eksperymentu i ma na celu dobranie kodeka audio najlepiej spełniającego wymagania stawiane aplikacjom projektowanym na platformy mobilne.

Ocena wykorzystująca estymację parametru MOS(ang. Mean Opinion Score) umożliwia ocenę subiektywnej jakości połączenia VoIP. W tabeli nr 4 zawarto opis znaczenia wartości parametru MOS. Algorytm zintegrowany z klientem SIP został udostępniony autorowi pracy dzięki uprzejmości firmy VoipSwitch. W trakcie estymacji parametru MOS pod uwagę brane są wartości:

- opóźnienia pakietów
- jitter
- straty pakietów
- opóźnienia wprowadzane przez kompresję

Tabela 4: Charakterystyka wartości parametru MOS.

| MOS | Jakość | Degradacja sygnału |
|-----|--------------|--|
| 5 | Znakomita | Niezauważalna |
| 4 | Dobra | Zauważalna ale nie powoduje większego wysiłku słuchowego |
| 3 | Dostateczna | Lekko zwiększony wysiłek słuchowy |
| 2 | Słaba | Zwiększony wysiłek słuchowy |
| 1 | Bardzo słaba | Nieakceptowalna |

W tabeli nr 5 zamieszczono zestawienie pomiarów zebranych w trakcie badania wartości parametru MOS dla różnych kodeków audio. W trakcie eksperymentu, nie użyto żadnej z metod manipulacji strumieniem danych.

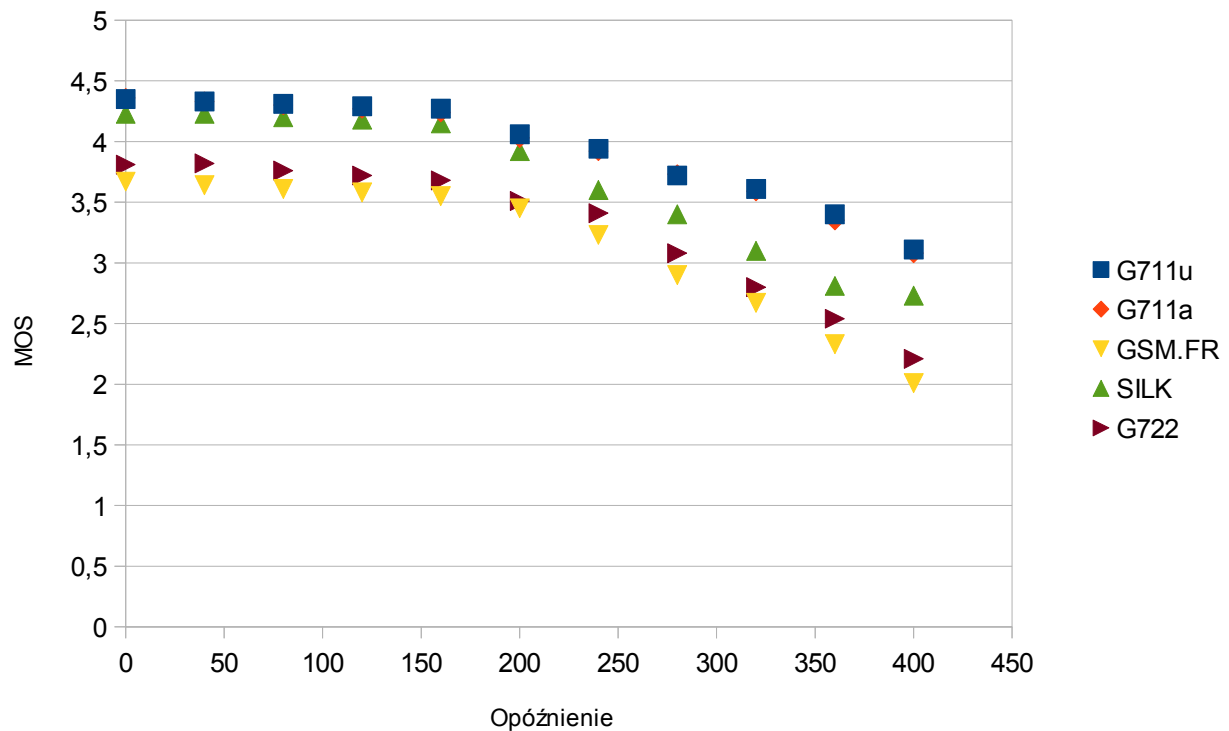
Tabela 5: Ocena MOS kodeków audio.

| Kodek | Częstotliwość we/wyj kodeka [kHz] | Zużycie łącza [kbps] | MOS |
|--------|--------------------------------------|-------------------------|------|
| G.711u | 8 | 64 | 4,36 |
| G.711a | 8 | 64 | 4,31 |
| Speex | 8 | 2,15 – 44,2 | 3,81 |
| Speex | 16 | 2,15 – 44,2 | 3,92 |
| GSM.FR | 8 | 12,2 | 3,67 |
| G722 | 16 | 48 - 56 | 3,95 |
| iLBC | 8 | 15,2 | 4,11 |
| SILK | 8 | 6 - 40 | 4,23 |
| SILK | 16 | 6 - 40 | 4,45 |

Z analizy zebranych pomiarów wynika, że dla częstotliwości próbkowania najlepszym współczynnikiem wartości parametru MOS do zużycia łącza danych charakteryzuje się kodek SILK 16kHz. Jednak biorąc również pod uwagę zużycie procesora lepiej wypada kodek G722.

W przypadku kodeków, które na wejściu i wyjściu wymagają zastosowania kodowania liniowego PCM o częstotliwości próbkowania 8kHz najlepiej wypadł kodek GSM.FR. Cechuje się on niskim zużyciem łącza oraz stosunkowo niewielkim zużyciem CPU. Ponadto mimo graniczących z dopuszczalnymi normami wartościami współczynnika MOS, jakość głosu przy wykorzystaniu tego kodeka jest zaskakująco dobra. Niska wartość oceny jakości ma najprawdopodobniej związek z zaniżonymi parametrami oceny użytego algorytmu kodowania.

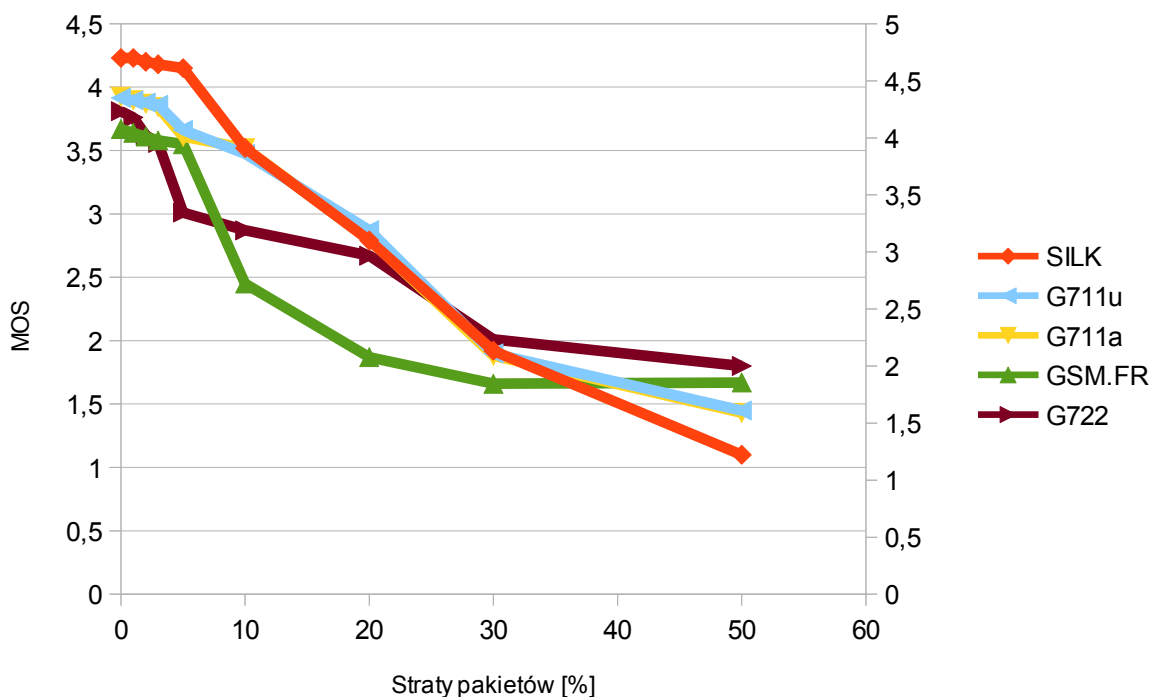
Kolejnym eksperymentem jest zbadanie wpływu opóźnienia pakietów audio na jakość rozmowy VoIP. W trakcie przeprowadzania eksperymentu wykorzystano autorską aplikację do symulacji warunków sieciowych. Na rysunku nr 16 znajduje się wykres ilustrujący wyniki badań.



Rysunek 16: Wpływ opóźnienia na wartość parametru MOS.

W trakcie analizy wyników badań zauważono, że charakterystyka parametru MOS jest podobna dla każdego z kodeków audio. Przy opóźnieniu większym niż 250ms jakość rozmowy przestała być akceptowalna dla kodeków SILK, Speex oraz GSM.FR. Co ciekawe dla kodeków G711a i G711u nawet opóźnienia rzędu 350ms pozwalały zachować dostateczną jakość połączenia.

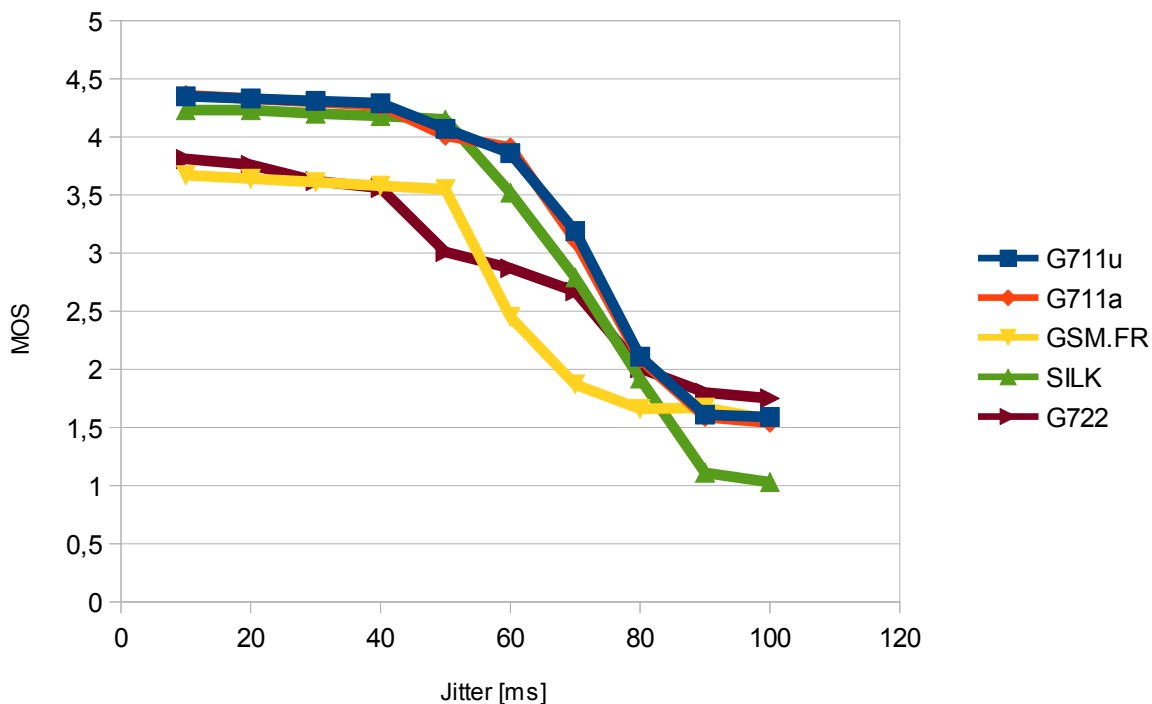
Celem następnego eksperymentu jest sprawdzenie wpływu strat pakietów na współczynnik jakości parametru MOS. Do przeprowadzenia badań wykorzystano autorską aplikację do symulacji warunków sieciowych. Na rysunku nr 17 znajduje się wykres ilustrujący wyniki badań.



Rysunek 17: Wpływ strat pakietów na wartość parametru MOS.

W trakcie analizy wyników badań zauważono, że na straty pakietów najbardziej odporne są kodeki SILK oraz G711. Zastosowanie mechanizmów PLC(ang. Packet loss cancelment), może znacząco poprawić jakość połączenia w przypadku kiedy mamy do czynienia ze stratami na sieci. Ponadto część kodeków audio jest wyposażona we wbudowany moduł PLC, należą do nich m.in. kodeki SILK, Opus, iLBC.

Na rysunku nr 18 zamieszczono wizualizację wyników pomiarów zebranych w trakcie badania wpływu zjawiska jitter'u na wartość parametru MOS dla różnych kodeków audio. W trakcie eksperymentu zastosowano ramki audio o długości 20ms oraz autorską aplikację do symulacji parametrów zjawiska jitter'u.

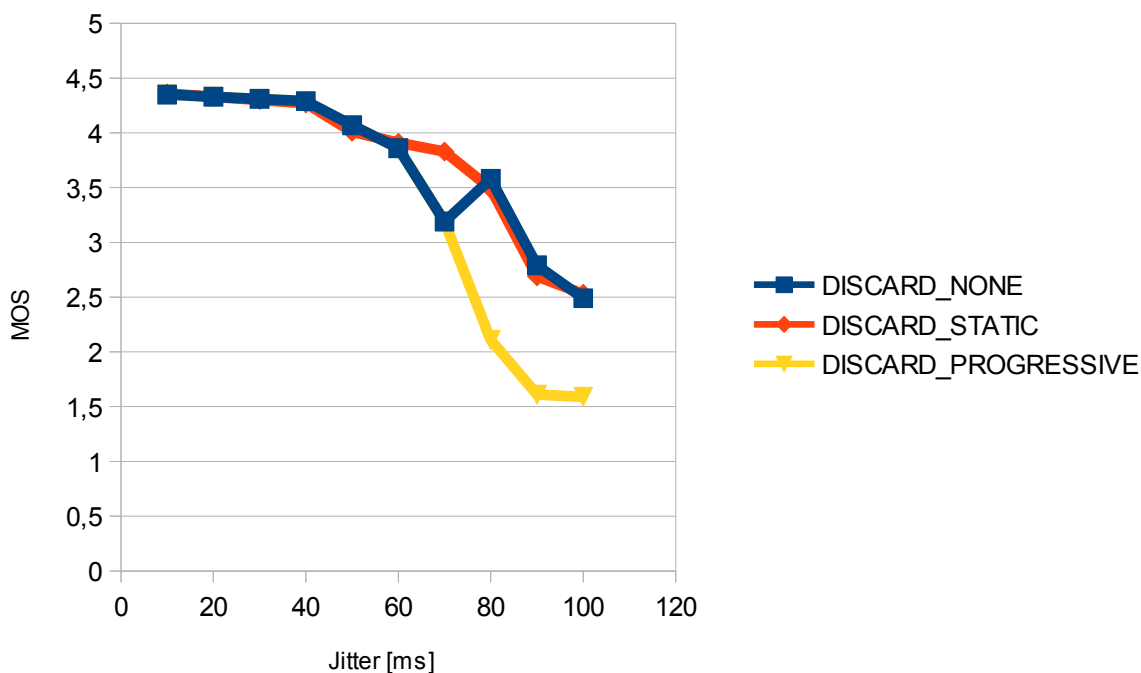


Rysunek 18: Wpływ zjawiska jitter'u pakietów na wartość parametru MOS.

Analizując zebrane pomiary można stwierdzić, że kodekiem najbardziej wrażliwym na występowanie zjawiska Jitter'u jest GSM.FR. Natomiast najbardziej odpornym kodekiem jest G711.

W trakcie przeprowadzania badań użyto bufora fluktuacji o długości 75ms. W kolejnym eksperymencie zbadany zostanie wpływ różnych algorytmów działania jitter buffer'a w bibliotece PJSIP.

Na rysunku nr 19 przedstawiono wyniki pomiarów przeprowadzonych dla różnych strategii działania bufora fluktuacji w bibliotece PJSIP. Do przeprowadzenia eksperymentu użyto kodeka G711u oraz ramki audio o długości 20ms.



Rysunek 19: Wpływ algorytmu bufora fluktuacji na zjawisko jitter'u oraz wartość MOS.

W trakcie analizy wyników badań zauważono, że algorytm DISCARD_NONE, który nie odrzuca żadnych ramek oraz algorytm statyczny DISCARD_STATIC odrzucający ramki opóźnione o więcej niż 200ms, dają lepsze wyniki w warunkach panujących w trakcie eksperymentu. Jednak zastosowanie tych algorytmów niesie za sobą pewne konsekwencje. Duża długość bufora fluktuacji powoduje większe opóźnienia oraz odgrywanie często nieaktualnych już informacji. Może to prowadzić do tego, że rozmowa będzie uciążliwa dla użytkownika.

4.4 Wnioski

Przeprowadzone badania pozwoliły na dobranie optymalnej konfiguracji klienta SIP dla wybranych interfejsów sieciowych. Poniżej zostaną opisane wybrane parametry, które powinny być użyte w trakcie nawiązywania rozmowy VoIP.

Platforma Android udostępnia zestaw funkcji pozwalających pobrać informacje o aktualnie wykorzystywanym interfejsie sieciowym. Pozwala to zdeterminować czy używana jest sieć WiFi czy sieć internetu mobilnego (2/3/4 G). Posiadając te istotne informacje można zdefiniować zestaw parametrów, z którymi powinna zostać uruchomiona aplikacja kliencka SIP.

W przypadku wykorzystywania bezprzewodowego połączenia WiFi, kodekiem, który najlepiej spełnia wymogi niskiego zużycia procesora oraz pozwala na zachowanie wysokiej jakości połączenia jest G722. Poziom wykorzystania łącza w przypadku tego algorytmu kompresji nie jest tak istotny z uwagi na stosunkowo dużą przepustowość medium transmisyjnego. Ponadto w związku z tym, że technologia WiFi posiada mniejsze opóźnienia przy przesyłaniu pakietów, bufor fluktuacji biblioteki PJSIP powinien działać w oparciu o algorytm progresywny (DISCARD_PROGRESSIVE). Pozwoli to zapewnić niższe wartości opóźnień w transmisji audio a przez to lepszą jakość rozmowy.

W przypadku połączenia z wykorzystaniem internetu mobilnego, kodek GSM.FR wydaje się być najlepszym kandydatem z uwagi na niskie zużycie łącza danych. Dodatkową zaletą tego kodeka jest stosunkowo niskie zużycie procesora. Technologie mobilnego dostępu do internetu cechują się większymi opóźnieniami, w związku z tym bufor fluktuacji biblioteki PJSIP powinien działać w oparciu o algorytm progresywny (DISCARD_STATIC). Zmniejszy to liczbę odrzucanych pakietów, a jednocześnie długość bufora rzędu 200ms nie wpłynie negatywnie na jakość rozmowy.

Naturalną rzeczą związaną z platformami mobilnymi, jest fakt, że użytkownik w trakcie prowadzenia rozmowy VoIP może się przemieszczać. Co za tym idzie zmieniać się mogą używane interfejsy sieciowe. Aby temu zapobiec należy zaimplementować moduł, który takie zmiany wykryje i zaktualizuje konfigurację aktywnej rozmowy. W tym celu należy przeprowadzić ponowną negocjację mediów. Po zaaplikowaniu odpowiedniej konfiguracji, należy wysłać uaktualnioną informację o sesji SDP. Nośnikiem tej informacji są wiadomości protokołu SIP, INVITE lub UPDATE. Po skończonej negocjacji optymalne warunki rozmowy zostaną zapewnione.

5. Podsumowanie

Celem niniejszej pracy dyplomowej było zbadanie zastosowań protokołu SIP w mobilnej telefonii VoIP oraz doświadczalne dobranie optymalnych parametrów rozmowy VoIP dla wybranej platformy mobilnej. Przeprowadzone badania pozwoliły na dobranie optymalnej konfiguracji klienta SIP dla różnych interfejsów sieciowych dostępnych na urządzeniach przenośnych. Na potrzeby realizacji pracy zaimplementowany został serwer pośredniczący umożliwiający zmianę parametrów sieciowych w trakcie rozmowy VoIP.

Zaimplementowany w kliencie SIP moduł służący do akwizycji danych o jakości połączenia mógłby być również wykorzystany do prezentacji informacji użytkownikowi aplikacji.

Ocena jakości połączenia audio nie jest zagadnieniem trywialnym, szereg parametrów wpływających na proces przesyłania mowy jest często ściśle zależny od istniejącej infrastruktury sieci IP. Jednak zastosowanie odpowiednich mechanizmów pozwala zmniejszyć degradujący wpływ tych zjawisk na jakość rozmowy audio. Efektem realizacji pracy jest zbiór zaproponowanych parametrów opisujących konfigurację aplikacji klienckiej SIP w zależności od wykorzystywanej technologii połączenia internetowego.

Rozwinięciem przeprowadzonych badań mogłoby być zbadanie wpływu poziomu sygnału WiFi i połączenia internetu mobilnego na jakość rozmowy audio. W celu zmniejszenia wpływu tego zjawiska konieczny byłby przegląd kodeków audio posiadających możliwość zmiany wykorzystania łączy danych w trakcie działania. Dodatkowo należałoby zaimplementować specjalny moduł, który pozwoliłby na śledzenie poziomu sygnału oraz dostosowywanie do tych warunków odpowiednio parametry strumienia danych.

Bibliografia

- [1] Shane Conder, Lauren Darcey "Android. Programowanie aplikacji na urządzenia przenośne", Wydawnictwo HELION, Gliwice 2011.
- [2] IETF "RTP: A Transport Protocol for Real-Time Applications", <http://www.ietf.org/rfc/rfc3550.txt> [1.07.2013]
- [3] Jiri Kuthan, Dorgham Sisalem, "SIP: More Than You Ever Wanted To Know About", Tekelec, Marzec 2007.
- [4] IETF "SDP: Session Description Protocol", <http://tools.ietf.org/html/rfc4566> [1.07.2013]
- [5] IETF "SIP: Session Initiation Protocol", <http://www.ietf.org/rfc/rfc3261.txt> [1.07.2013]
- [6] Łukasz Pokojowy "Badanie Jakości usługi VoIP w oparciu o E-model", Instytut Telekomunikacji, Teleinformatyki i Akustyki Politechniki Wrocławskiej. 2013
- [7] PJSIP Documentation, <http://trac.pjsip.org/repos/> [1.07.2013]
- [8] CSipSimple Documentation & Source Code <http://code.google.com/p/csipsimple/> [1.07.2013]
- [9] M.J. Trzaskowska, B. Mucha "Metody obiektywnej oceny jakości usługi głosowe QoS w sieciach łączności elektronicznej oraz urządzenia do takiej oceny i do badania dostępności 'usług' poprzez numery alarmowe - etap 1", Instytut Łączności, Warszawa 2006
- [10] Ferguson P., Huston G.: Quality of Service in the Internet: Fact, Fiction, or Compromise, INET98', Genewa 1998
- [11] Jan Mastalir "Understanding SIP-Based VoIP", http://www.packetizer.com/ipmc/sip/papers/understanding_sip_voip [1.07.2013]
- [12] ITU-T Recommendation G.107 (12/11) The E-model: a computational model for use in transmission planning
- [13] ITU-T Recommendation P.800. Methods for subjective determination of transmission quality, Sierpień 1996.
- [14] Digium, The Asterisk Company "Asterisk Quick User Guide", 2012

Spis Ilustracji

| | |
|--|----|
| Rysunek 1: Przykładowa architektura infrastruktury opartej o protokół SIP [3]..... | 5 |
| Rysunek 2: Przykład nawiązywania rozmowy w infrastrukturze opartej o SIP [11]..... | 6 |
| Rysunek 3: Przykładowa wiadomość SDP [3]..... | 7 |
| Rysunek 4: Struktura pakietu RTP [3]..... | 8 |
| Rysunek 5: Architektura stanowiska badawczego..... | 12 |
| Rysunek 6: Architektura biblioteki PJSIP [7]..... | 13 |
| Rysunek 7: Interfejs aplikacji CSipSimple [8]..... | 14 |
| Rysunek 8: Diagram klas modułu służącego do akwizycji danych pomiarowych (klient SIP)..... | 16 |
| Rysunek 9: Architektura sieci VoIP opartej o serwer Asterisk [14]..... | 17 |
| Rysunek 10: Diagram klas serwera pośredniczącego – moduł SIP..... | 20 |
| Rysunek 11: Diagram klas serwera pośredniczącego – moduł manipulacji mediami..... | 21 |
| Rysunek 12: Startowe okno konfiguracyjne serwera pośredniczącego..... | 24 |
| Rysunek 13: Interfejs graficzny serwera pośredniczącego..... | 25 |
| Rysunek 14: Okno ustawień parametrów sieciowych serwera pośredniczącego..... | 26 |
| Rysunek 15: Opcje konfiguracji konta użytkownika w aplikacji CSipSimple..... | 27 |
| Rysunek 16: Wpływ opóźnienia na wartość parametru MOS..... | 32 |
| Rysunek 17: Wpływ straty pakietów na wartość parametru MOS..... | 33 |
| Rysunek 18: Wpływ zjawiska jitter'u pakietów na wartość parametru MOS..... | 34 |
| Rysunek 19: Wpływ algorytmu bufora fluktuacji na zjawisko jitter'u oraz wartość MOS..... | 35 |

Spis Tabel

| | |
|--|----|
| Tabela 1: Parametry najczęściej używanych kodeków audio w technologii VoIP [11]..... | 10 |
| Tabela 2: Wpływ częstotliwości próbkowania audio na zużycie procesora..... | 28 |
| Tabela 3: Wpływ kodeka audio na zużycie procesora..... | 29 |
| Tabela 4: Charakterystyka wartości parametru MOS..... | 30 |
| Tabela 5: Ocena MOS kodeków audio..... | 31 |