**POLITECHNIKA POZNAŃSKA**

**Telefonia IP**

**Projekt komunikatora głosowego**

**Damian Lewandowski**

**Dariusz Bernad**

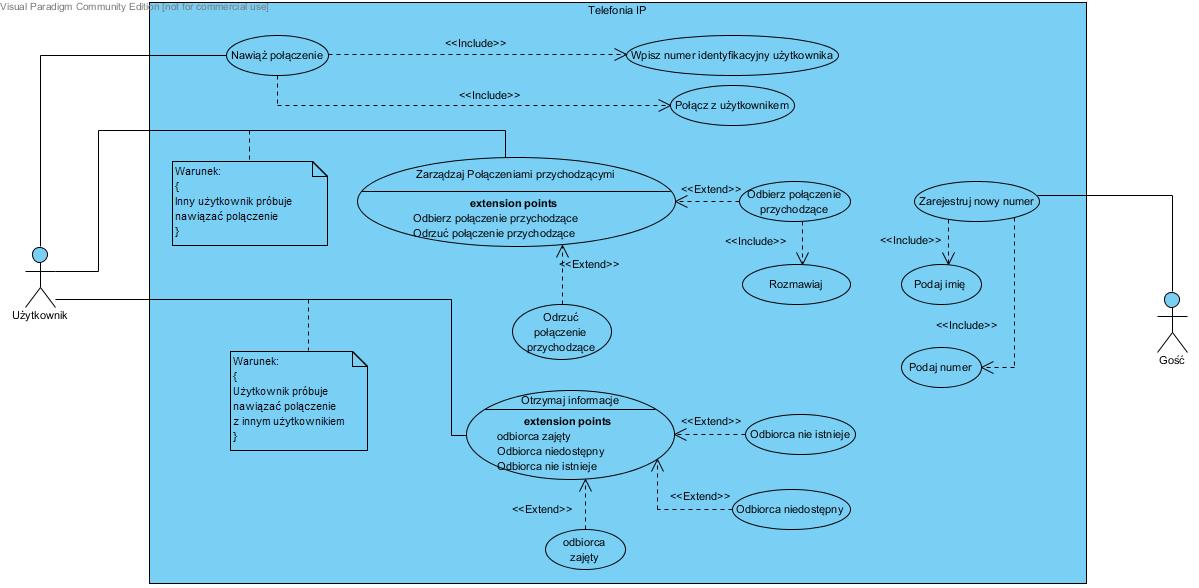
**1. Wstęp:**

Celem tego projektu jest zaprojektowanie oraz implementacja komunikatora głosowego służącego do komunikowania się między użytkownikami przy wykorzystaniu sieci komputerowych opartych o stos protokołów TCP/IP. Komunikator ten poza protokołem sterowania ma również przetwarzać, wysyłać , odtwarzać odbierać głos między użytkownikami.  
  
**2. Zasada działania systemu:**

Aby możliwa była w miarę swobodna komunikacja między użytkownikami przyjęta została zasada działania tego systemu która jest bardzo podobna do tego jak działają obecne telefony komórkowe oraz telefony stacjonarne. W tym celu wydzielone zostały następujące sytuacje:

* Użytkownik A ma zamiar nawiązać połączenie głosowe z użytkownikiem B, jednak użytkownik B nie jest podłączony (zalogowany) do systemu przez co nie będzie możliwości nawiązania połączenia. Użytkownik A otrzymuje zwrotną informację o tym że nie ma możliwości połączenia się z użytkownikiem B z powodu tego że użytkownik ten nie jest podłączony.
* Użytkownik A ma zamiar nawiązać połączenie głosowe z użytkownikiem B, jednak użyty numer będący identyfikatorem tego użytkownika nie istnieje. Użytkownik A analogicznie jak było w poprzednim przykładzie otrzymuje informację zwrotną o tym że nie ma takiego numeru.
* Użytkownik A ma zamiar nawiązać połączenie głosowe z użytkownikiem B, po pewnym czasie oczekiwania połączenie zostaje nawiązane oraz następuje komunikacja między tymi użytkownikami. Po naciśnięciu przycisku zakończenia połączenia z strony użytkownika A lub użytkownika B następuje zakończenie połączenia.
* Użytkownik A ma zamiar nawiązać połączenie głosowe z użytkownikiem B, następuje oczekiwanie na połączenie jednak użytkownik B z jakiegoś powodu odrzuca połączenie. Użytkownik A otrzymuje informację że użytkownik B jest zajęty.
* Użytkownik A ma zamiar nawiązać połączenie głosowe z użytkownikiem B, jednak okazuje się że użytkownik B nawiązał połączenie głosowe z innym użytkownikiem przez co nie ma możliwości nawiązania połączenia. Użytkownik A otrzymuje komunikat o tym że użytkownik B jest zajęty.

Przyjęte założenia przedstawione zostały w postaci poniższego diagramu przypadków użycia:



*il. 1: Diagram przypadków użycia systemu telefonii IP*

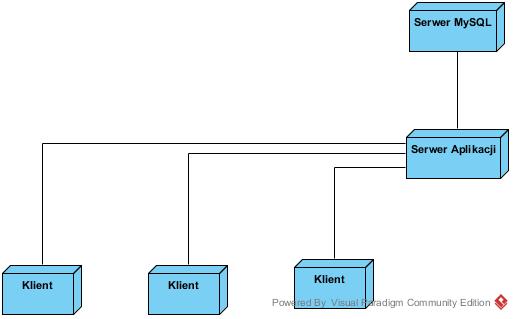
Na powyższym diagramie przypadków użycia można zauważyć aktora o nazwie gość. W tym wypadku gościem można nazwać użytkownika który nie posiada jeszcze numeru który umożliwiałby komunikację z innym użytkownikiem. Aby gość stał się użytkownikiem musi zarejestrować nowy numer identyfikacyjny, numer ten jest niezbędny by móc używać aplikacji. Przypadek "Podaj numer" jest przypadkiem w którym użytkownik sam definiuje sobie numer identyfikacyjny.

**3. Architektura działającego oprogramowania:**

Aby możliwe było działanie tego systemu na zasadzie takiej jakiej opisane zostało w punkcie poprzednim, przyjęta została architektura klient - serwer - klient. Dzięki takiej przyjętej architekturze możliwe jest sterowanie sesją połączeniową w taki sposób by możliwe były przypadki typu:

* zajęty,
* niedostępny,
* niepodłączony,
* podłączony,

Architektura została przedstawiona na poniższym diagramie:



Na powyższej ilustracji przedstawiona została architektura tego systemu dla trzech klientów.

Sytuacja ta jest przykładem gdyż w praktyce może być podłączona do serwera różna ilość klientów ze względu na to że serwer jest wielowątkowy.

Architektura ta uwzględnia następujące elementy:

* **Serwer aplikacji** - Aplikacja wielowątkowa pełniąca rolę centrali telefonicznej, zarządzająca protokołem sygnalizacyjnym
* **Klient -** Aplikacja posiadająca graficzny interfejs użytkownika, łącząca się z serwerem aplikacji, w celu realizacji połączenia z innym klientem,
* **Serwer MySQL -** Baza danych przechowywująca kontakty użytkowników, posiadająca zarejestrowane nazwy użytkowników aplikacji klienckich oraz przypisane do nich numery identyfikacyjne.

**4. Charakterystyka protokołu sygnalizacyjnego:**

By możliwa była sesja między dwoma użytkownikami zdefiniowany zostanie protokół komunikacyjny warstwy zastosowań wykorzystujący protokół transportowy TCP ze względu na swoje właściwości: Pakiety te będą przesyłane w sytuacjach gdy użytkownik będzie próbował nawiązać połączenie z innym użytkownikiem oraz w momencie przerwania transmisji. Transmitowane będą dane w postaci tekstowej do serwera oraz pod pewnymi warunkami przekazywane do drugiego użytkownika przez serwer. Dane będą następujące rodzaje komunikatów:

* INVITE - wysyłany do odbiorcy za pośrednictwem serwera kiedy odbiorca ten jest podłączony w celu zainicjalizowania sesji połączeniowej,
* TRANSFER - komunikat zwrotny od serwera informujący nadawcę o tym że komunikat "INVITE" został przekazany do odbiorcy. Komunikat ten informuje nadawcę o tym że nadawca odebrał żądanie odbioru połączenia głosowego,
* OK - wysyłane przez odbiorcę oraz nadawcę w celu akceptacji nawiązania połączenia z centralą, akceptacji zakończenia sesji podłączeniowej z centralą, akceptacji połączenia z odbiorcą (wykorzystywany również w protokole sprawdzającym stan połączenia).
* NEX - (Not EXists) - komunikat zwrotny od serwera w wypadku gdy nadawca próbuje się połączyć z odbiorcą który nie istnieje,
* NAC - (Not ACcess) - komunikat zwrotny od serwera w wypadku gdy nadawca próbuje połączyć się z odbiorcą który nie jest podłączony z serwerem
* BUSY - komunikat zwrotny od serwera w wypadku gdy nadawca próbuje się połączyć z użytkownikiem który odrzucił połączenie bądź jest zajęty,
* NOT - wysyłane przez serwer do klienta w wypadku gdy podczas podłączania się numer identyfikacyjny klienta jest już zarezerwowany bądź nie istnieje w Bazie danych,
* BYE - komunikat wysyłany do serwera w celu poinformowania go o tym że użytkownik odłącza się od serwera.

Do obsługi sesji dodawania użytkownika również będzie wykorzystana sesja sygnalizacyjna.

W tym celu wykorzystane zostanie zastosowany następujący komunikat:

* ADD - komunikat służący do dodawania nowego abonenta w momencie pierwszego uruchamiania aplikacji.
* OK - Potwierdzenie dodania nowego użytkownika,
* NOK - Odrzucenie żądania dodania nowego użytkownika spowodowane tym że użytkownik o takim numerze już istnieje.

Podobnie jak komunikaty sygnalizacyjne służące do sesji połączeniowej powyższy komunikat będzie wysyłany do serwera w postaci prefiksowej.

Sesja dodawania nowego użytkownika jest konieczna za pośrednictwem serwera gdyż w celu zapewnienia bezpieczeństwa systemu dostęp do bazy danych ma tylko i wyłącznie serwer pełniący rolę centrali telefonicznej.

Komunikaty protokołu sterowania będą miały przyjętą postać prefiksową.

Przykłady komunikatów protokołu sterującego:

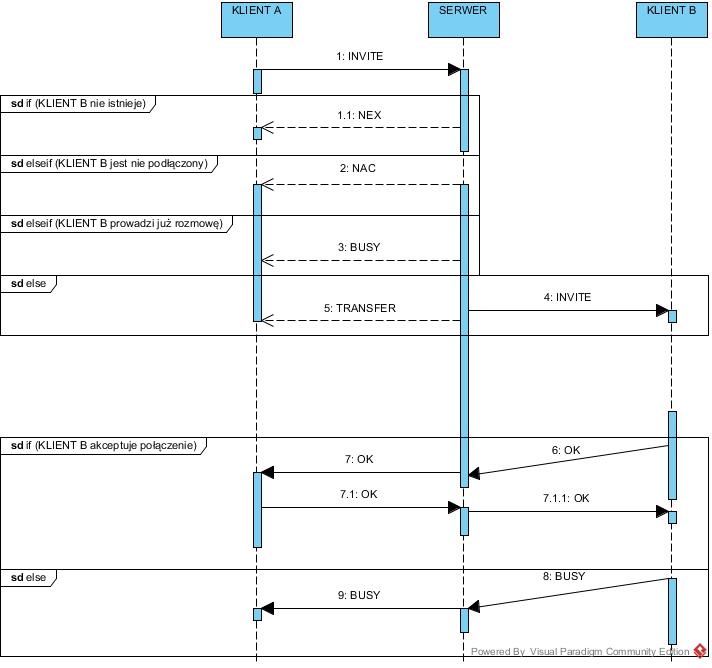
* INVITE 5550989 543432332 192.168.6.33 - Użytkownik o numerze 5550989 próbuje nawiązać połączenie z użytkownikiem 543432332, przekazany również zostaje adres IP nadawcy do odbiorcy w celu zastawienia połączenia w wypadku gdy odbiorca zaakceptuje połączenie,
* OK 5550989 543432332 192.168.6.34 - Użytkownik akceptuje połączenie nawiązane przez innego użytkownika będącego nadawcą,
* HELLO kamil 554343234 - użytkownik "kamil" o numerze identyfikacyjnym 554343234 informuje serwer o tym że chce się podłączyć do serwera,
* ADD jaś 123456789 - Klient wysyła żądanie do serwera w celu dodania nowego użytkownika o imieniu "jaś" oraz numerze "123456789".

Protokół ten będzie obsługiwał 2 rodzaje sesji:

* sesja komunikacyjna między klientami za pośrednictwem serwera.
* sesja dodawania nowego użytkownika do bazy danych.

Sesja komunikacyjna między klientami za pośrednictwem serwera pełniącym rolę centrali

została przedstawiona w sposób następujący:

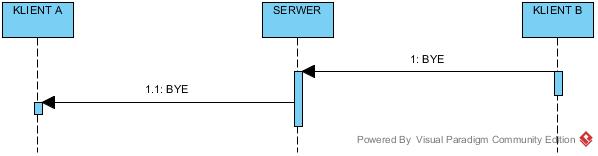


*il1: Sesja połączenia między klientami za pośrednictwem serwera pełniącego rolę centrali telefonicznej:*

W momencie gdy klient został podłączony do systemu, użytkownik klienta wybiera numer identyfikacyjny innego użytkownika z którym chce nawiązać połączenie. Wysyła do serwera polecenie "INVTE". Serwer odbiera ramkę zawierającą ten komunikat i w pierwszej kolejności sprawdza czy numer identyfikacyjny odbiorcy istnieje w Bazie danych. Jeżeli nie istnieje wysyłany jest komunikat zwrotny informujący klienta o tym że odbiorca o takim numerze identyfikacyjnym nie istnieje czyli nadawca odbiera komunikat "NEX", w przeciwnym wypadku sprawdza czy odbiorca który ma numer zarejestrowany w Bazie danych jest podłączony do centrali. W tym wypadku przeszukana zostanie baza klientów podłączonych do centrali. Jeżeli numer odbiorcy znajdzie się w tej liście a tym samym odbiorca zostanie podłączony do centrali, zostanie przekazana do niego komunikat "INVITE", w przeciwnym wypadku centrala wyśle komunikat "NAC" informujący nadawcę o tym ze "Abonent jest tymczasowo niedostępny". Odbiorca po odebraniu komunikatu "INVITE" informuje użytkownika o tym że nadawca próbuje się z nim skontaktować. Jeżeli użytkownik zaakceptuje połączenie, odbiorca wyśle do nadawcy za pośrednictwem serwera komunikat "OK" i rozpocznie transmisję pakietów danych zawierających głos. Serwer odbierający ten komunikat zarejestruje nadawcę oraz odbiorcę jako zajęte hosty by w przypadku gdy inny użytkownik będzie chciał się połączyć, otrzyma informację o tym że hosty o tych numerach są już w trakcie prowadzenia rozmowy. Gdy Nadawca odbierze "OK" rozpoczęte zostanie odbieranie danych głosowych do klienta. Aby poinformować odbiorcę że połączenie zostało już zastawione, odsyła spowrotem do odbiorcy ten sam pakiet zawierający komunikat "OK".

W przypadku gdy odbiorca odrzuci połączenie, wysyła do nadawcy komunikat "BUSY" nadawca zostanie poinformowany o tym że odbiorca odrzucił połączenie.

Sesja rozłączeniowa protokołu sygnalizacyjnego przedstawiono na poniższym diagramie sekwencji:



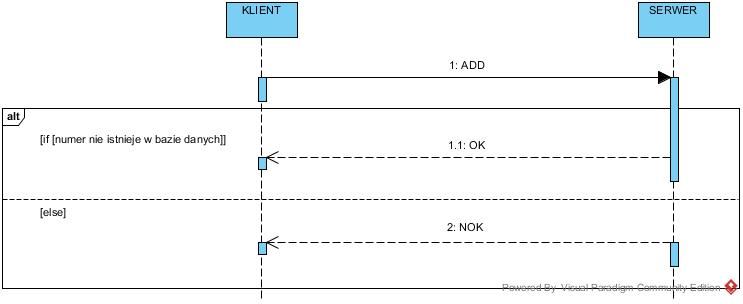
*il2: Sesja rozłączeniowa między klientami za pośrednictwem serwera pełniącego rolę centrali telefonicznej:*

Aby sesja zakończeniowa połączenie miało miejsce spełniony musi być jeden warunek:

* musi trwać połączenie między nadawcą a odbiorcą czyli dane głosu są transmitowane w sieci.

Sesja ta następuje w momencie gdy któryś ż użytkowników zamierza zakończyć połączenie głosowe. W momencie zakończenia połączenia przez użytkownika, użytkownik ten wysyła komunikat "BYE" informujący klienta o tym że połączenie jest już zakończone oraz informujący serwer o tym że użytkownicy o takich numerach zakończyli rozmowę więc od tego momentu inni użytkownicy będą mogli się skontaktować z użytkownikami którzy aktualnie prowadzili rozmowę. Nadawca po odebraniu komunikatu "BYE" kończy odbieranie oraz nadawanie pakietów dźwięku - połączenie zostaje zakończone.

Sesja dodawania użytkownika wyglądać będzie następująco:



*il3: Sesja dodawania nowego użytkownika*

*:*

**5. Charakterystyka protokołu stanu połączenia hostów:**

Podczas uruchamiania się aplikacji klienckiej bądź wyłączania, klient musi poinformować serwer o tym że jest połączony co wiąże się z tym że w serwerze odnotowana zostaje informacja o tym że dany użytkownik właśnie zostaje podłączony, bądź rozłączony.

Dzięki tej informacji możliwe jest wygodne zarządzanie oraz sprawdzanie czy dany użytkownik będzie miał szansę połączyć się z innym użytkownikiem. W tym celu zdefiniowane zostaną następujące sesje:

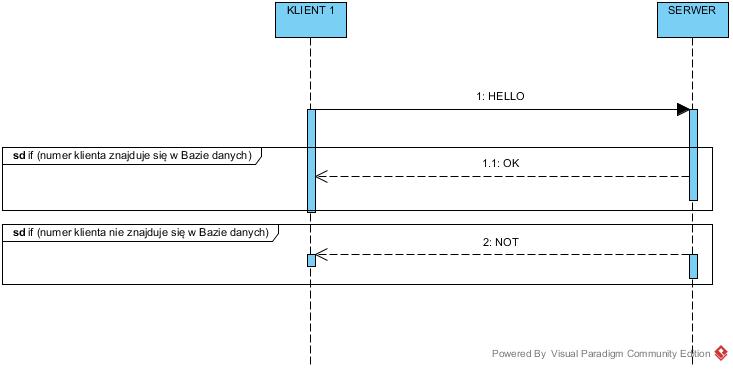
* sesja podłączeniowa klienta do centrali,
* sesja rozłączeniowa klienta z centralą,
* sesja sprawdzająca stan połączenia klienta z centralą.

Podczas trwania rozmowy telefonicznej między dwoma użytkownikami może wystąpić sytuacja w której serwer z jakiegoś powodu ulega awarii. Dane głosu przesyłane są między klientami bez pośrednictwa serwera. Jeżeli dojdzie do sytuacji w której centrala przestanie działać, wtedy nie możliwa będzie realizacja protokołu sygnalizacyjnego gdyż wszystkie komunikaty tego protokołu przepływają przez centralę. Aby zapobiec takiej sytuacji klient będzie wysyłał komunikat do centrali "ECHO" co odpowiedni czas by sprawdzić czy centrala nadal działa. W przypadku kiedy zabraknie odpowiedzi serwera, następować będzie rozłączanie rozmowy oraz wyświetlenie informacji o tym że nastąpił błąd krytyczny serwera.

Protokół ten podobnie jak protokół sygnalizacyjny będzie korzystał z warstwy protokołu TCP ze względu na fakt że dane będą przesyłane cylicznie. Komunikaty równie będą przesyłane w postaci tekstowej. Dla wyżej wymienionych przypadków komunikaty te będą posiadały następującą formę:

* HELLO - Komunikat wysyłany do serwera w celu poinformowania go o tym że użytkownik podłącza się do serwera,
* EXIT - Komunikat wysyłany do serwera informujący go o tym że klient chce się rozłączyć,
* ECHO - Komunikat wysyłany przez klienta do serwera w czasie cyklicznym w celu sprawdzenia czy serwer odpowiada na żądania, Komunikat ten ma zapobiegać niestabilności systemu w sytuacji awarii serwera a tym samym awarii protokołu sygnalizacyjnego.

Sesja podłączeniowa klienta do serwera przedstawiona jest na poniższym diagramie sekwencji:

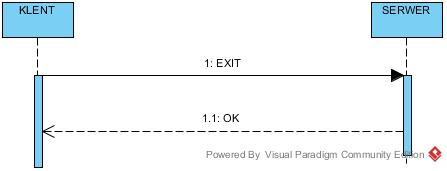


*il.4: Sesja przyłączeniowa klienta do centrali:*

Klient podczas uruchamiania sie wysyła do serwera pełniącej rolę centrali komunikat o tym że chce zostać podłączony. Serwer wysyła odpowiedź zwrotną "OK" by poinformować klienta o tym że klient został podłączony lub odpowiedź "NOT" w wypadku wystąpienia jakichkolwiek problemów. Problemami w tym przypadku mogą być następujące:

* Klient wysyła komunikat z numerem identyfikacyjnym który nie istnieje w Bazie danych
* Klient wysyła komunikat z numerem identyfikacyjnym który numer ten ma status już podłączonego - można to interpretować jako przejęcie numeru identyfikacyjnego przez innego użytkownika które jest nie pożądane dla działania systemu.

Sesja rozłączeniowa klienta z centralą przedstawiona została na poniższym diagramie:

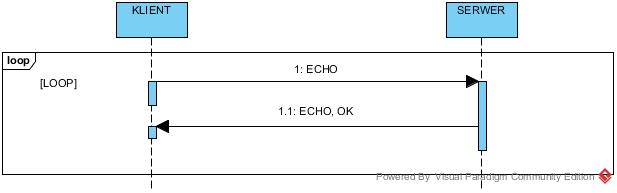


*il.5: Sesja rozłączeniowa klienta z centralą:*

Sesja ta jest realizowana w momencie zamknięcia aplikacji klienckiej. Informuje on serwer o tym że klient nie jest już podłączony, co jest przydatne przy informowaniu innych klientów o tym że nie można się z tym klientem połączyć ze względu na to że klient nie jest połączony.

Serwer wysyła odpowiedź zwrotną informującą klienta o tym że procedura odłączenia klienta od serwera zakończyła się pomyślnie.

Sesja sprawdzająca stan połączenia klienta z centralą został przedstawiony na poniższym diagramie sekwencji:



Sesja ta staje się realizowana przez klienta w momencie kiedy klient przyłącza się do centrali a przestaje być w momencie zakończenia działania klienta. Celem tej sesji jest zapewnienie przepływu informacji dla klienta o tym że centrala działa w sposób właściwy dzięki czemu możliwe jest realizowanie sygnalizacji przez protokół sygnalizacyjny. Sygnał "ECHO" Wysyłany przez klienta jest wysyłany w odpowiednim odstępie czasu. Jeżeli zajdzie sytuacja kiedy przez ten odstęp czasu klient nie otrzyma komunikatu "ECHO OK" dla klienta będzie to znak że centrala przestala odpowiadać lub został wyłączony lub doszło do awarii systemu. W tym wypadku klient przerywa działanie wykonywania połączenia z klientami, oraz połączenie z centralą dla wszystkich urządzeń zostaje zerwane. Mechanizm ten ma zabezpieczać przed prowadzeniem rozmowy bez możliwości działania protokołu sygnalizacyjnego. W praktyce gdyby doszło do błędu w protokole sygnalizacyjnym doszło by do sytuacji kiedy rozmowa z użytkownikami nadal była by realizowana, jednak nie możliwe było by rozłączenie się.

**6. Charakterystyka protokołu transportowego**

**czasu rzeczywistego:**

Do transmisji głosu między dwoma użytkownikami wykorzystany będzie inny protokół który oparty będzie na protokole UDP ze względu na to by pakiety te mogły dojść do hostów możliwe w jak najkrótszym czasie. Transmisja głosu odbywać się będzie za pośrednictwem jednego portu.

**7. Charakterystyka bezpiecznego Protokołu transportowego czasu rzeczywistego:**

Dopuszczalna jest w tym projekcie możliwość szyfrowanie transmisji głosowej w celu zapobiegnięcia podsłuchania przez osoby do tego nieupoważnione. Należy uwzględnić fakt że podczas transmisji głosu istotne jest aby czasy propagacji nie przekraczały odpowiednich wartości żeby możliwe było zapewnienie transmisji wysokiej jakości, dlatego algorytm szyfrujący musi być algorytmem szybkim, a przede wszystkim wydajnym (czasu rzeczywistego). Ze względu na swoje właściwości transmisja głosu będzie szyfrowana szyfrem strumieniowym z wykorzystaniem generatora liczb pseudolosowych - Szyfr Vernama. Zasada takiego szyfrowania jest następująca: Otóż każdy bajt transmisji będzie sumowany modulo z bajtem klucza wygenerowanego przez generator liczb pseudolosowych w wyniku czego zaszyfrowana zostaną oraz wysłane zostaną dane głosowe. Klucz będzie wysyłany poprzez inny port przy wykorzystaniu SSL.

**8. Specyfikacja wymagań:**

Na podstawie przyjętych wcześniej założeń oraz zasad działania tego oprogramowania przyjęte zostały następujące wymagania:

* **Wymagania funkcjonalne:**

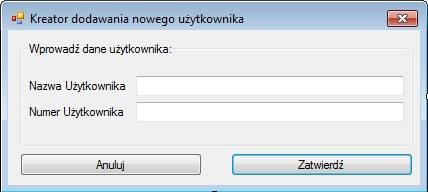
1. Aby usługi komunikatora były dostępne należy dodać Użytkownika do Bazy danych
2. Można połączyć się z odbiorcą pod warunkiem że odbiorca jest połączony z serwerem
3. Odbiorca ma czas nieograniczony na odbiór połączenia
4. Czas rozmowy między nadawcą a odbiorcą jest nieograniczony
5. Odbiorca może odrzucić połączenie nadawcy
6. Podczas wybierania połączenia słyszany jest dźwięk sygnalizujący użytkownika o tym połączenie jest nawiązywane
7. W wypadku odrzucenia połączenia bądź gdy odbiorca ma już nawiązane połączenie nadawca słyszy dźwięk sygnalizujący jego o tym że numer jest zajęty
8. W wypadku gdy odbiorca o danym numerze nie istnieje, nadawca otrzymuje komunikat o tym że nie ma takiego numeru!
9. W wypadku gdy odbiorca o danym numerze nie jest podłączony, nadawca otrzymuje komunikat o tym że użytkownik jest nie podłączony
10. Aplikacja klienta posiada interfejs graficzny
11. Aplikacja serwera ma postać konsolową
12. Aplikacja kliencka uruchamia kreatora tworzenia nowego konta w momencie pierwszego uruchamiania aplikacji
13. Odbiorca może w każdej chwili od momentu nawiązania połączenia rozłączyć się z nadawcą.
14. Dopuszczalna jest transmisja głosu zaszyfrowanego

* **Wymagania niefunkcjonalne:**

1. Serwer korzysta z Bazy danych
2. Baza danych zawiera informacje w postaci Nazwy oraz przypisany do niej numer identyfikacyjny
3. Aplikacja serwera generuje logi w sytuacjach wyjątkowych i zapisuje je do pliku tekstowego
4. W momencie uruchomienia aplikacji klienta, serwer jest informowany o tym że klient o danym numerze jest podłączony, analogicznie w momencie zakończenia
5. Klucz szyfrujący będzie generowany przez generator licz pseudolosowych

**9. Wygląd aplikacji (klient)**





**10. Omówienie implementacji**

1. **Serwer**
2. **Klient**
   1. **Sesja komunikacyjna**
   2. **Przesyłanie danych przy pomocy UDP**

|  |
| --- |
| **//Using UDP sockets**  clientSocket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);  EndPoint ourEP = new IPEndPoint(IPAddress.Any, 1450);  **//Listen asynchronously on port 1450 for coming messages //(Invite, Bye, etc).**  clientSocket.Bind(ourEP);  **//Receive data from any IP.**  EndPoint remoteEP = (EndPoint)(new IPEndPoint(IPAddress.Any, 0));    byteData = new byte[1024];  **//Receive data asynchornously.**  clientSocket.BeginReceiveFrom(byteData, 0, byteData.Length, SocketFlags.None, ref remoteEP, new AsyncCallback(OnReceive), null); |

***Konfiguracja obiektu klasy Socket do wysyłania oraz do nasłuchiwania na transmisję przychodzącą.***

|  |
| --- |
| if (vocoder == Vocoder.ALaw)  {  byte[] dataToWrite = ALawEncoder.ALawEncode(memStream.GetBuffer());  udpClient.Send(dataToWrite, dataToWrite.Length, otherPartyIP.Address.ToString (), 1550);  }  else if (vocoder == Vocoder.uLaw)  {  byte[] dataToWrite = MuLawEncoder.MuLawEncode(memStream.GetBuffer());  udpClient.Send(dataToWrite, dataToWrite.Length, otherPartyIP.Address.ToString(), 1550);  }  else  {  byte[] dataToWrite = memStream.GetBuffer();  udpClient.Send(dataToWrite, dataToWrite.Length, otherPartyIP.Address.ToString(), 1550);  } |

***Wysyłanie kodowanie danych w zależności od wybranego kodeka i wysyłanie ich do drugiego klienta.***

* 1. **Nagrywanie, kodowanie i odtwarzanie dźwięku**
* **Konfiguracja danych wejściowych**

|  |
| --- |
| **//Stereo.**  short channels = 1;  **//32Bits, 16Bits, 8Bits.**  short bitsPerSample = 32;  **//11KHz = 11025 , 22KHz = 22050, 44KHz = 44100**  int samplesPerSecond = 22050; |

*Dane wejściowe do nagrywania dźwięku.*

Powyższe wartości określają dane wejściowe do nagrywania dźwięku. Ich zmiana wpływa na jakość i szybkość uzyskiwania danych.

|  |
| --- |
| **//Set up the wave format to be captured.**  waveFormat = new WaveFormat();  waveFormat.Channels = channels;  waveFormat.FormatTag = WaveFormatTag.Pcm;  waveFormat.SamplesPerSecond = samplesPerSecond;  waveFormat.BitsPerSample = bitsPerSample;  waveFormat.BlockAlign =  (short)(channels \* (bitsPerSample / (short)8));  waveFormat.AverageBytesPerSecond =  waveFormat.BlockAlign \* samplesPerSecond; |

*Określenie formatu zapisywanego dźwięku.*

|  |
| --- |
| **//Set up captureBuffer**  captureBufferDescription = new CaptureBufferDescription();  captureBufferDescription.BufferBytes= waveFormat.AverageBytesPerSecond ;  captureBufferDescription.Format = waveFormat;  **//Set up playbackBuffer**  playbackBufferDescription = new BufferDescription();  playbackBufferDescription.BufferBytes= waveFormat.AverageBytesPerSecond ;  playbackBufferDescription.Format = waveFormat;  playbackBuffer= new SecondaryBuffer(playbackBufferDescription, device); |

*Definicja do utworzenia bufora nagrywania oraz odtwarzania danych.*

* **Kodowanie**

|  |
| --- |
| **//Get the vocoder to be used.**  if (cmbCodecs.SelectedText == "A-Law")  {  vocoder = Vocoder.ALaw;  }  else if (cmbCodecs.SelectedText == "u-Law")  {  vocoder = Vocoder.uLaw;  }  else if (cmbCodecs.SelectedText == "None")  {  vocoder = Vocoder.None;  } |

***Kodowanie G.711***

Do kodowania przesyłanego dźwięku wykorzystywane są algorytmy G.711 w zależności od preferencji klienta rozpoczynającego rozmowę :

1. Law (G.711U)
2. μ-law (G.711A)

Jest to podstawowa norma dla modulacji PCM w telefonii cyfrowej. Jest często stosowana w telefonii internetowej (VoIP) i zgodnie z G.711 można kodować sygnały o paśmie do 4kHz.

* **Odtwarzanie dźwięku**

|  |
| --- |
| **//G711 compresses the data by 50%, so we allocate a buffer of double the size to store the decompressed data.**  byte[] byteDecodedData = new byte[byteData.Length \* 2];  **//Decompress data using the proper vocoder.**  if (vocoder == Vocoder.ALaw)  {  ALawDecoder.ALawDecode(byteData, out byteDecodedData);  }  else if (vocoder == Vocoder.uLaw)  { MuLawDecoder.MuLawDecode(byteData, out byteDecodedData);  }  else  {  byteDecodedData = new byte[byteData.Length];  byteDecodedData **= byteData;**  **}** |

***Dekodowanie dźwięku.***

|  |
| --- |
| **//Play the data received to the user.**  playbackBuffer = new SecondaryBuffer(playbackBufferDescription, device);  playbackBuffer.Write(0, byteDecodedData, LockFlag.None);  playbackBuffer.Play(0, BufferPlayFlags.Default); |

***Odtwarzanie dźwięku przy pomocy DirectSound.***

Po dekodowaniu danych otrzymujemy dźwięk przesłany nam przez drugiego klienta po UDP.

**11. Środki techniczne wykorzystane w projekcie**

Do zaprojektowania oraz implementacji tego systemu wykorzystane zostaną następujące środki techniczne:

* **Visual Paradigm CE 12.0** - Aplikacja służąca do modelowania oraz projektowania w języku UML tego systemu. Wszystkie diagramy UML tworzone zostaną za pomocą tej aplikacji.
* **Github** - Repozytorium projektu w którym umieszczane zostaną prototypy aplikacji oraz cały projekt wraz z dokumentacją.
* **Serwer VPS** - Komputer klasy PC z zainstalowanym systemem operacyjnym Debian 7.6 Wheezy, posiadający publiczny adres IP oraz własną domenę. Na tym komputerze będzie działał serwer pełniący rolę centali oraz działała będzie Baza danych.
* **MySQL** - Baza danych działająca również na serwerze VPS, na której będzie przechowywana lista osób oraz przypisane do nich numery identyfikacyjne,
* **PhpMyAdmin**  - Narzędzie służące do zarządzania bazą danych MySQL,
* **MS Visual Studio 2013** - Srodowisko programistyczne na którym zaimplementowana zostanie aplikacja klienta
* **Eclipse Luna** - Środowisko programistyczne na którym zaimplementowana zostanie aplikacja serwera. Ze względu na wieloplatformowość oraz na fakt że serwer będzie działał na systemie operacyjnym Linux preferowanym językiem programowania do zaimplementowania tego systemu jest JAVA.
* **DirectSound**- część pakietu DirectX umożliwiająca szybki dostęp do karty dźwiękowej i m.in. odtwarzanie i nagrywanie dźwięku.