POLITECHNIKA ŁÓDZKA
Samodzielny Zakład Sieci Komputerowych
Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej
90-924 Łódź, ulica Stefanowskiego 18/22, tel./ fax (48)(42) 636 03 00, e-mail: szsk@zsku.p.lodz.pl

Michał Słoma

Telefonia IP

praca dyplomowa magisterska

Promotor:
dr inż. Michał Morawski

Dyplomant:
Michał Słoma
nr albumu 104920

Łódź, czerwiec 2005 r.

Spis Treści

Akronimy

AAL	ATM Adaptation Layer
ATM	Asynchronous Transfer Mode
DNS	Domain Name Server
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
ISP	Internet Services Provider
ITU-T	International Telecommunications Union – Telecommunications
JMF	Java Media Framework
MGCP	Media Gateway Control Protocol
NAT	Network Address Translator
POTS	Public Old Telephone Service
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RAS	Registration Admission and Status
RSVP	Resource Reservation Protocol
RTCP	Real Time Transport Control Protocol
RTSP	Real Time Streaming Protocol
RTP	Real Time Protocol
SAP	Session Announcement Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SCTP	Stream Control Transmission Protocol
STUN	Simple Traversal of UDP Through NAT
TURN	Traversal Using Relay NAT
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server

Rozdział 1. Wstęp

Rozwój Internetu w ostatnich latach XX wieku przyniósł mnóstwo nowych możliwości. Podstawowym motorem rozwoju wszystkich technologii, jest działalność gospodarcza, która stawia informatykom coraz większe wymagania. Jedną z dziedzin, która ma wyjątkowo duże znaczenie dla każdego rodzaju biznesu jest telefonia. Obecnie większość dużych firm posiada komputerowe stanowiska pracy połączone siecią lokalną w każdej swojej filii. Każda taka filia jest wyposażona zazwyczaj w szerokopasmowe łącze internetowe. Komunikacja w firmie odbywa się oczywiście drogą elektroniczną, ale i w tradycyjny sposób poprzez sieć telefonii komutowanej PSTN (Public Switched Telephone Network). Nowoczesne rozwiązania polegają jednak na zastąpieniu komunikacji analogowej telefonią związaną z siecią pakietową nazywaną Telefonią IP. Dzieje się tak głównie ze względów finansowych. Tradycyjna sieć telefoniczna potrzebuje własnego okablowania oraz centrali telefonicznej zarządzającej siecią. Rozwiązania Telefonii IP wykorzystują sieć komputerową a funkcję centrali pełnić może darmowy programowy serwer zainstalowany na istniejącym już sprzęcie.

Wiek XXI przynosi nam sprawdzone już rozwiązania dotyczące telefonii wspieranej przez sieć Internet. Coraz więcej korporacji z branży IT dostrzega potrzeby rozwijania tej właśnie dziedziny informatyki. Dostrzec można obecnie znaczny rozwój oprogramowania oraz inwestycje dotyczące sprzętu telefonicznego współpracującego z sieciami pakietowymi. Jest to zjawisko bardzo pozytywne nie tylko dla rozproszonych lokalizacyjnie firm, ale także dla klientów indywidualnych.

Firmy takie jak omawiana wyżej, posiadające wiele placówek z dostępem do sieci globalnej, coraz częściej decydują się na rozwiązania Telefonii IP. Stosowanie takiej metody głosowej komunikacji przynosi wymierne korzyści finansowe. Oddziały firm, których pracownicy porozumiewają się bardzo często w czasie pracy między sobą, redukują koszty w postaci mniejszych rachunków telefonicznych od providerów telefonii PSTN. Klienci indywidualni także mogą ograniczać wydatki na rozmowy telefoniczne. Szczególnie korzystne wydaje się korzystanie z rozmów o dalekim zasięgu, które są znacznie droższe w telefonii tradycyjnej.

Dzisiejsze czasy nie pozwalają nam już żyć bez telefonu. Porozumiewanie się telefonicznie to podstawa szybkiego załatwienia wszelakich spraw, zatem Telefonia IP ma przed sobą duże

perspektywy. Jedyną przeszkodą szybkiego rozwoju jest jak dotąd brak ujednoliconych standardów. Telefonia IP jak każda nowa dziedzina rozwijała się w kilku środowiskach projektowych. Obecnie obserwujemy dwa liczące się standardy, które zdobywają rynek. Sytuacja szybko stabilizuje się co powoduje osiąganie głównego celu jakim jest integracja transmisji głosu i danych oraz stworzenie uniwersalnej sieci mogącej przenosić każdy rodzaj ruchu.

Rozdział 2. Cel i zawartość pracy

2.1 Cel pracy

Celem pracy było przybliżenie podstaw działania telefonii IP. Zakres pracy obejmuje omówienie protokołów komunikacyjnych używanych do komunikacji w telefonii IP oraz implementacja wybranych elementów tej technologii. Praca zawiera opis aktualnych rozwiązań opracowywanych na potrzeby rozwijania jeszcze młodej dziedziny informatyki, jaką jest właśnie telefonia IP. Część praktyczna pracy obejmuje implementację podstawowych protokołów stosowanych w tego typu aplikacjach, co stanowić będzie podstawę do zbudowania docelowo aplikacji oprogramowania klienckiego. Aplikacja spełnia zadania uproszczonego terminala programowego (urządzenia klienckiego dla telefonii IP) obsługującego wybrany standard stosowany w telefonii IP. Działanie oprogramowania powinno być przetestowane także dla urządzeń dedykowanych, produkowanych obecnie i występujących na rynku. Urządzeniami mogą być telefony IP obsługujące określony standard technologii. Wnioski wysnute podczas testów będą stanowiły ocenę działania oprogramowania, przydatność i wydajność technologii.

2.2 Streszczenie rozdziałów

Rozdział pierwszy omawia ogólnie standardy telefonii IP oraz przedstawia, czym jest telefonia IP i do czego może służyć.
Rozdział drugi omawia dokładniej jeden z protokołów sygnalizacyjnych – H.323.
Rozdział trzeci omawia najpopularniejszy w czasie pisania pracy dyplomowej standard sygnalizacyjny – SIP. W tym rozdziale zawarte są teoretyczne przykłady sesji w SIP a także innowacje wprowadzane do standardu do maja 2005 roku.
Rozdział czwarty zawiera omówienie praktycznej części pracy. Przybliża budowę wykorzystywanego projektu, omawia napotkane problemy oraz poprawki wprowadzone do projektu. W końcowej części rozdziału zawarte są wyniki przeprowadzonych testów projektu.

2.3 Aktualność tematu

Temat, jakim zająłem się w mojej pracy magisterskiej jest w fazie intensywnego rozwoju i dopracowywania szczegółów standardów. Telefonia IP jest przyszłością komunikacji w Internecie oraz Intranecie. Obecnie istnieje już wiele implementacji urządzeń telefonii IP. Są one cały czas udoskonalane i powoli komercjalizują się stanowiąc nową usługę w Internecie.

Skomputeryzowane firmy liczące kilkadziesiąt i więcej pracowników coraz częściej rozważają wprowadzenie telefonii IP jako podstawy komunikacji w firmie.
Podczas pisania mojej pracy powstawały nowe dokumenty zawierające innowacje dla standardów Telefonii IP, co pozwala myśleć, że jest to aktualnie jedna z najdynamiczniej rozwijających się dziedzin informatyki.

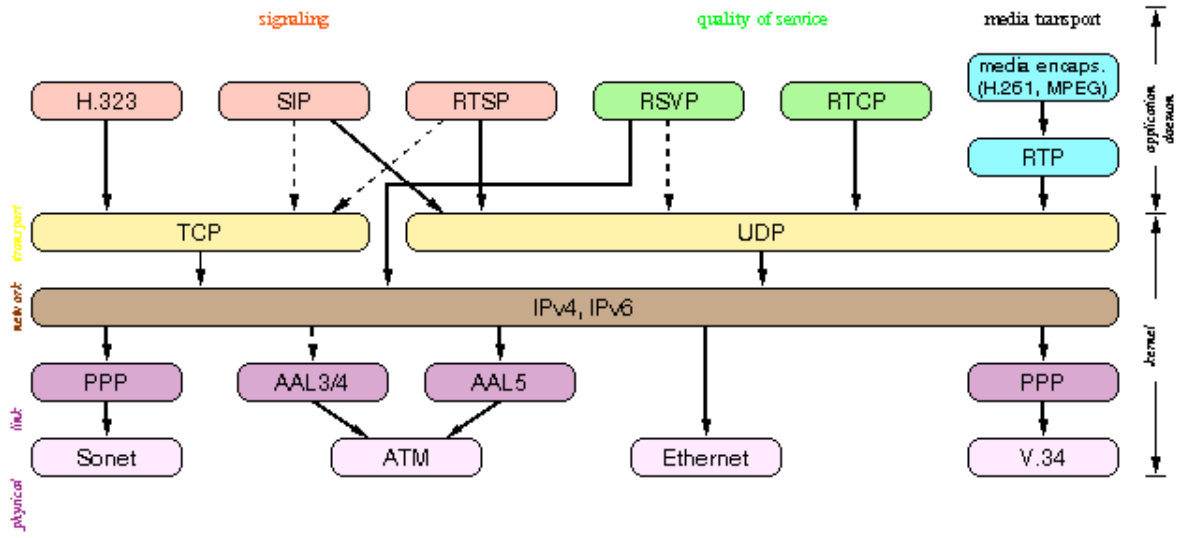
Rozdział 3. Telefonia IP i jej standardy

3.1 Pojęcie Telefonii IP

Telefonia IP (Internet Telephony) oznaczana także IPT jest to przenoszenie rozmów telefonicznych przez Internet. Nieważne przy tym jest, czy rozmowy przenoszone są całkowicie, czy też częściowo przez sieć pakietową oraz jakie urządzenia są do tego celu wykorzystywane (tradycyjne, komputery PC, urządzenia dedykowane).[1]

3.2 Usługi telekomunikacyjne w sieciach IP

Telefonia IP składa się z wielu elementów składowych. Połączenie ich w całość daje pozytywne rezultaty w komunikacji przez Internet. Stosowany jest cały stos protokołów, z których każdy pełni ważną rolę. Rysunek 3.1 przedstawia stos protokołów wykorzystywanych w telekomunikacji pakietowej.



Rys.3.1 Stos protokołów potrzebnych do telefonii IP[3]

Wyróżniamy protokoły należące do kilku grup, z których każda grupa pełni określone funkcje. Podział przedstawia się następująco:

sygnalizacja

- SIP(Session Initiation Protocol)[35],

- H.323[15],
- MGCP (Media Gateway Control Protocol)[16],
- H.248/Megaco[17],
- RTSP (Real Time Streaming Protocol)[12],

zarządzanie jakością (Quality of Service)

- RSVP (Reservation Protocol)[9],
- RTCP (Real Time Transport Control Protocol)[13],

transport w czasie rzeczywistym

- RTP[4],
- Kodeki[22],

transport pakietowy

- TCP[2],
- UDP[34],

połączenie internetowe

- warstwa sieci modelu ISO/OSI: IPv4, IPv6[2],
- warstwa łącza danych modelu ISO/OSI: PPP, ATM, Ethernet itd.

inne

- DNS[19].

3.3 Protokoły sygnalizacyjne

3.3.1 Zadania protokołów sygnalizacyjnych

Najwyżej w hierarchii stoją protokoły sygnalizacyjne. Stanowią element logiczny sieci odpowiedzialny za sterowanie przebiegiem sesji w bezpołączeniowym środowisku pakietowym[1]. Kontrolują one wszelkie parametry połączenia, a dodatkowo mogą zajmować się bezpieczeństwem oraz bilingiem, czyli naliczaniem opłat. Do konkretnych zadań realizowanych przez protokoły sygnalizacyjne należą[1]:

- **ustalanie adresów IP oraz numerów portów wykorzystywanych przez wymieniane dane**,
  pakiety danych kierowane są zupełnie inną drogą niż pakiety protokołów sygnalizacyjnych, choć zdarza się, że niewielkie ilości danych użytkowych są przesyłane tą samą drogą co informacje sygnalizacyjne, ma to miejsce np. przy wysyłaniu wizytówek czy zdjęć

- **lokalizacja użytkowników,**
  protokół musi posiadać mechanizmy ustalania, gdzie jest zalogowany użytkownik konta, jest to realizowane najczęściej podczas komunikacji serwerów (obsługujących dany standard protokołu). Kolejne przekierowania zapytania o użytkownika owocują jego lokalizacją, protokół sygnalizacyjny powinien więc mieć odpowiednie do tego celu typy komunikatów.
- **zestawianie połączeń,**
  Jeżeli użytkownik A (dzwoniący) chce nawiązać połączenie z użytkownikiem B, protokół ma obowiązek zasygnalizować użytkownikowi B nadejście połączenia, realizowane jest to poprzez wysłanie tak zwanego „zaproszenia". Zaproszenie ma różną postać dla poszczególnych protokołów. Po zaakceptowaniu zaproszenia przez użytkownika odbierającego B, ustalane są parametry transmisji i otwierane odpowiednie porty dla protokołów transportowych.
- **dostępność użytkownika,**
  Protokoły sygnalizacyjne muszą posiadać w swojej implementacji komunikaty stanowiące o dostępności użytkownika w danej chwili, jest to ważne o tyle, gdyż użytkownik niedostępny nie jest w stanie odebrać zaproszenia do rozmowy.
- **możliwości użytkownika,**
  bardzo ważnym jest ustalenie parametrów, jakie mogą być osiągnięte są przez obu użytkowników chcących się skomunikować, protokół sygnalizacyjny dba o ustalenie jak najlepszych parametrów możliwych do osiągnięcia podczas transportu pakietów z właściwymi danymi.
- **kończenie sesji,**
  rozłączenie rozmowy między użytkownikami także niesie za sobą pewne obowiązki, które muszą być spełnione przez protokół sygnalizacyjny. Są to czynności „sprzątające" a należą do nich między innymi zamykanie portów protokołów transportowych oraz wysyłanie komunikatów zakończeniowych.

3.3.2 Przegląd istniejących rozwiązań protokołów sygnalizacyjnych.

Obecnie rozwijanych jest wiele standardów sygnalizacyjnych, a w związku z tym producenci rozwiązań Telefonii IP wahają się, który standard poprzeć i implementować na swoich urządzeniach. Sytuacja stanie się klarowna, gdy dominować będzie jeden standard, który implementowany będzie na wszystkich urządzeniach. Doprowadzi to na pewno do

rozpowszechnienia urządzeń, które będą wtedy tanimi terminalami produkowanymi bez ryzyka nietrafnej inwestycji.

RTSP

RTSP jest sygnalizacyjnym protokołem pomocniczym Nie dokonuje on działań kluczowych dla telefonii IP. RTSP (Real-Time Streaming Protocol) kontroluje inicjację i przesyłanie strumienia multimedialnego z serwerów multimedialnych. Jest to protokół warstwy aplikacji do sterowania prezentacją multimedialną w transmisjach strumieniowych, przebiegających w czasie rzeczywistym. Zapewnia synchronizację transmisji wielu strumieni danych, a dla każdego strumienia dokonuje wyboru odpowiedniego protokołu transportowego UDP, TCP czy też RTP[11]. Podstawowe komendy RTSP to [12]:

- SETUP - zestawienie połączenia,
- PLAY - wysyłanie danych,
- RECORD - odbieranie danych multimedialnych,
- PAUSE - przerwa w transmisji bez zwolnienia zasobów,
- TEARDOWN - zakończenie transmisji i zwolnienie zasobów.

SIP

SIP jest to standard stworzony przez Multiparty Multimedia Session Control (MMUSIC), który jest zespołem stworzonym i pracującym dla IETF. Jest to protokół tekstowy. Ma wiele cech wspólnych z protokołem HTTP, np. komunikaty zwrotne. Protokół SIP jest ciągle doskonalony i zdobywa coraz większe poparcie poprzez implementacje w urządzeniach dedykowanych telefonii IP znanych firm telekomunikacyjnych[1]. Architekturę protokołu SIP tworzą dodatkowo jeszcze dwa inne protokoły sygnalizacyjne: Session Description Protocol (SDP)[48] i Session Announcement Protocol (SAP)[49]. SDP używany jest do opisu możliwości i typu przesyłanych mediów zaimplementowanych w terminalu SIP, natomiast SAP do informowania o sesjach multicast'owych takich jak publiczne konferencjach czy radio i telewizja internetowa[1].

H.323

H.323 jest standardem starszym niż SIP. Prace nad serią protokołów należących do standardu H.323 rozpoczęły się około roku 1996[15]. Jeszcze niedawno posiadał on całkiem dobrą sytuację na rynku telekomunikacyjnym. Niestety dla organizacji ITU-T zostaje on zastępowany coraz częściej protokołem SIP[1]. H.323 jest częścią serii, która obejmuje protokoły dla różnych typów sieci.

Są to:

- H.320 dla wąskopasmowych sieci ISDN,
- H.321 dla szerokopasmowych cyfrowych sieci ISDN i ATM,
- H.322 dla sieci pakietowych z gwarantowanym pasmem,
- H.323 dla sieci pakietowych z nie gwarantowanym pasmem,
- H.324 dla sieci analogowych POTS (Public Old Telephone Servise),

Standard H.323 jest dosyć skomplikowany w implementacji, stosuje wiele innych standardów ITU. Zawiera w sobie usługi dotyczące procedur kontrolnych, rodzaju medium oraz sygnalizacji zgłoszeń. Bazuje na protokołach TCP i UDP. Pierwszy z nich wykorzystywany jest do sygnalizacji i wymiany danych kontrolnych, natomiast drugi do przesyłania danych audio, video oraz typu RAS (Registration Admission Status)[15].

MGCP

Protokół sygnalizacyjny MGCP (Media Gateway Control Protocol) zajmuje się sterowaniem bram medialnych. Bramy są to urządzenia na pograniczu sieci pakietowych oraz tradycyjnych sieci POST (Public Old Telephone Service). W przeciwieństwie do SIP i H.323 nie jest protokołem typu end-to-end[18]. Jest to standard coraz mniej popularny. Opisany jest w RFC 2705[16]

H.248/Mecago

Podwójna nazwa protokołu nie jest przypadkowa. Rozwój protokołu wpierały dwie organizacje ITU-T oraz IETF. Protokół Mecago jest następcą MGCP. Łączy on cechy MGCP oraz H.323. Głównym założeniem Megaco jest zapewnienie skalowalności większej od gwarantowanej przez H.323. Mimo wsparcia dwóch największych organizacji standaryzujących protokół nie zdobył popularności[18].

Protokoły sygnalizacyjne H.323 oraz SIP w tym SDP zostały opracowane z większą dokładnością w kolejnych rozdziałach.

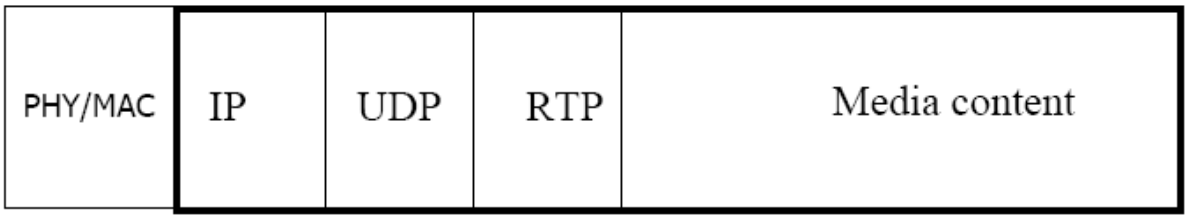
3.4 Protokoły transportowe i Quality of Service

Standardowe protokoły transportowe wykorzystywane w sieci Internet same w sobie nie nadają się za bardzo do przesyłania głosu. W warstwie transportowej stosu IP można

wyszczególnić dwa takie protokoły: TCP oraz UDP. Protokół TCP (Transmission Control Protocol) jest protokołem połączeniowym. Architekturę protokołów połączeniowych można porównać do rozmowy telefonicznej, gdzie najpierw nawiązywane jest połączenie, a następnie przesyłane są dane. Dane formowane są strumieniowo z zachowaniem kolejności oraz z zachowaniem niezawodności[2]. Taką rolę także próbuje pełnić TCP. Dodanie mechanizmów niezawodnego dostarczania danych. Realizowane jest to przez wprowadzenie numeru sekwencyjnego pakietu oraz funkcje kontroli transmisji i potwierdzenia. Zaginiony pakiet jest zawsze retransmitowany[2]. TCP zajmuje się więc kontrolą przepływu, gdzie koniecznością jest przesyłanie potwierdzeń otrzymania pakietu. Protokół TCP zajmuje się także innymi aspektami połączenia takimi jak kontrola przeciążeń.
Protokół TCP nie jest dobrym rozwiązaniem dla aplikacji działających w czasie rzeczywistym, jakimi są klienty telefonii IP, gdyż powoduje duże opóźnienia przez mechanizmy niezawodnego dostarczania danych. Szczególnie, gdy połączenie między komputerami jest nawiązywane poprzez sieć Internet, gdzie występują nieznane warunki przepływu pakietów, protokół połączeniowy generuje duże opóźnienia, co powoduje, że jest wielokrotnie wolniejszy od rozwiązań bezpołączeniowych.

Drugim protokołem transportowym jest UDP, protokół bezpołączeniowy, który nie gwarantuje dostarczenia wszystkich pakietów do odbiorcy. Działa on na zasadzie „best effort", czyli próbuje jak najlepiej dostarczyć pakiet, ale jeśli mu się to nie uda to nie retransmituje. UDP jest używany do przesyłania między innymi głosu, gdyż eliminuje przetwarzanie zagubionych pakietów[33]. Wadą, którą dostrzegamy z punktu widzenia przesyłania głosu może być brak znaczników czasu, co powoduje brak kontroli opóźnień oraz brak kontroli kolejności docierających do odbiorcy pakietów.
Aby wyeliminować wady protokołów warstwy sieci i warstwy transportowej modelu ISO OSI w aplikacjach pracujących w czasie rzeczywistym przez sieć Internet zastosowano protokoły takie jak RTP wraz z RTCP[3] czy też SCTP[5].
RTP (Real-time Transport Protocol) jest protokołem specjalnie zaprojektowanym do obsługi aplikacji przesyłających dane w czasie rzeczywistym. W roku 1996 opisany został po raz pierwszy w RFC 1889[4]. Jest on uzupełnieniem protokołu UDP w parametry zapewniające dobre efekty podczas transmisji wymagających uporządkowania i niezawodności. RTP doskonale spełnia wymagania aplikacji przenoszących głos, obraz czy też dane symulacyjne[3]. Nie jest to czysty protokół transportowy, gdyż nie posiada wielu funkcji takich jak zarządzanie przepływem pakietów, czy kontrolą przeciążeń. Jest to protokół

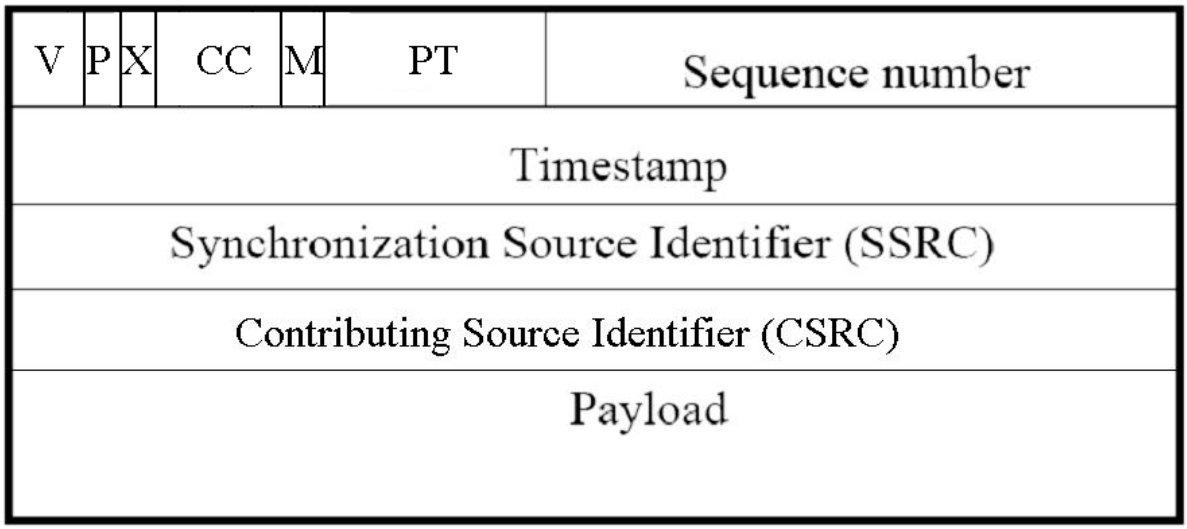
podrzędny dla UDP jak można to zauważyć na rysunku 3.2. UDP/RTP dostarcza niezbędnych możliwości do prawidłowej komunikacji w czasie rzeczywistym.



Rys. 3.2 Schemat pakietu danych stosowany dla telefonii IP[1]

Protokół RTP jest protokołem wykorzystującym stos IP/UDP. RTP składa się z dwóch części[3]:

- **Część danych**
  wspiera aplikacje czasu rzeczywistego poprzez przenoszenie danych wraz z dodatkowymi informacjami w nagłówku. Najważniejsze z nich to:
  - **Payload type (PT)** – reprezentuje typ przenoszonej informacji. Identyfikuje format ładunku RTP i określa sposób jego interpretacji przez aplikację. Może także zawierać informacje o sposobie kodowania danych[4].
  - **Sequence number** – numer sekwencyjny pakietu. Każdy kolejno wysyłany pakiet danych otrzymuje numer. W ten sposób odbiorca może weryfikować czy nadeszły wszystkie dane oraz może układać dane względem ich prawdziwej kolejności.
  - **Timestamp** - znacznik czasu. Wprowadza próbki zegara względem pierwszego oktetu pakietu RTP. Zegar musi być liniowy, by pozwalał synchronizować i obliczać wahania przybywających pakietów. Musi być także wystarczająco dokładny by było można mierzyć wahania przybycia paczek (np. jeden takt na ramkę wideo zazwyczaj nie wystarcza)[4].



Rys.3.3 Nagłówek protokołu RTP

Pozostałem pola nie są tak istotne dla możliwości protokołu i oznaczają:
V – wersję protokołu RTP,
P – znacznik dołączenia jednego oktetu dodatkowego do danych, wykorzystywany przy stosowaniu niektórych algorytmów szyfrowania,
X – znacznik rozszerzenia nagłówka,
CC – znacznik mówiący o liczbie pól CSRC, może być ich od 0 do 15, jest to pole 4 bitowe
M – znacznik wyróżnienia ramki,
SSRC – losowo generowana liczba identyfikująca źródło pakietu (unikalna dla źródła),
CSRC – identyfikator używany opcjonalnie, do przekazywania kluczy czy też identyfikatorów dla danych zawartych w pakiecie w polu Payload. Przykładem może być sytuacja, gdzie CSRS przechowują identyfikatory dla wszystkich źródeł dźwiękowych, które są zmiksowane i przesyłane w pakiecie. Może być do 15 takich pól w jednym pakiecie

- **Część kontroli**
  traktowana czasem jako oddzielny protokół RTCP. Wspiera komunikacje uczestników w grupach konferencyjnych dowolnej wielkości. Synchronizuje strumienie danych poprzez dostarczanie informacji monitorujących. RTCP bazuje na okresowej transmisji pakietów kontrolnych wszystkich uczestników sesji, przy

użyciu tych samych mechanizmów dystrybucji, jakie istnieją dla pakietów przenoszących dane[3].

Protokół RTP nie zajmuje się kwestią rezerwacji pasma, a co za tym idzie kontrolą jakości Quality of Service (QoS). Znaczy to tyle, że RTP będzie robił co może w każdych warunkach panujących w sieci, ale nie gwarantuje, że warunki te zapewnią prawidłową komunikację w czasie rzeczywistym[3]. Stos IP/UDP/RTP przyczynia się do polepszenia komunikacji w czasie rzeczywistym, lecz nie zawiera mechanizmów rezerwujących zasoby sieci. Dlatego też bardzo często wraz z RTP wykorzystywany jest protokół RSVP (ReSerVation Protocol) [9]. RSVP służy do ustalania odpowiednich parametrów transmisji danych takich jak opóźnienie i przepustowość dla konkretnych strumieni danych. Strumienie rozróżniane są przez adres IP typ protokołu oraz numer portu. Protokół wymusza efekt na działanie ruterów na całej ścieżce przepływu strumieni danych, stawiając wymagania co do stanu raz ustanowionego połączenia w dynamicznym środowisku sieci pakietowej. Rezerwacja zasobów jest jednokierunkowa[8]. Rezerwacje zasobów można sobie wyobrazić jako wyznaczenie „rury", w której przesyłane są dane. Schematyczną sytuację współpracy protokołów RTP i RSVP jaka zachodzi w telefonii IP przedstawia Rys.3.4

Rys.3.4 Schemat obrazujący relacje między protokołami RTP i RSVP

Kolejnym protokołem, który może służyć jako wsparcie dla programów czasu rzeczywistego jest SCTP. Protokół ten zapewnia niezawodne przesyłanie danych w warstwie transportowej. Może on operować na pakietach bezpołączeniowego protokołu IP. Oferuje on potwierdzony, wolny od błędów i duplikatów transport datagramów. Wykrywa on przekształcenie danych, ich stratę oraz duplikaty poprzez używanie sum kontrolnych oraz numerów sekwencyjnych. Zaimplementowany jest także mechanizm selektywnej retransmisji zniszczonych lub zmienionych danych[5].

Alternatywą dla stosu UDP/RTP/RSVP może być protokół DCCP (Datagram Congestion Control Protocol) znany wcześniej jako DCP[10]. Jest to protokół transportowy oferujący

obsługę dwukierunkowych połączeń unicast'owych opartych na datagramach z kontrolą przeciążeń. DCCP jest protokołem bez gwarancji dostarczenia danych (*unreliable*), opierającym się na datagramach, który jednak wprowadza pakiety potwierdzające w celu kontroli przeciążenia. Jest czymś pośrednim między TCP a UDP[30]. Protokół jest bardzo elastyczny, pozwala na stosowanie różnorodnych algorytmów do kontroli przeciążenia, przez co kontroluje stosunek punktualności do niezawodności. Protokół jest jeszcze w fazie projektowej, pracują nad nim specjaliści z organizacji IETF. Nie ukazały się jeszcze RFC, wydane zostały na razie szkice (drafts)[30].

3.5 Pozostały stos protokołów

Do spełnienia potrzeb telefonii IP potrzebne są oczywiście jeszcze dodatkowe protokoły, które będą spełniać podstawowe funkcje do ustanowienia i utrzymania połączenia. Jest to standardowy zestaw protokołów używanych powszechnie w Internecie. Protokoły warstwy fizycznej stanowią podstawę dla protokołów warstwy stanu łącza. W warstwie tej używane są protokoły takie jak: PPP, ATM, Ethernet czy też V.34 przy połączeniach modemowych[1]. Usługę ustanowienia połączenia internetowego obsługuje protokół IP w wersji 4 lub 6. Protokoły warstwy transportowej TCP oraz UDP używane są do przenoszenia informacji między maszynami. W warstwie aplikacji pracuje także ważny dla telefonii protokół DNS (Domain Name System), który troszczy się o lokalizację użytkowników na podstawie adresów czy też numerów identyfikacyjnych[37].
Mogą oczywiście być używane inne protokoły, które ułatwiać będą klientom lub urządzeniom Telefonii IP identyfikację w sieci czy też autoryzację użytkowników[37].

Rozdział 4. H.323

4.1 Wstęp

H.323 jest zestawem protokołów do prowadzenia komunikacji głosowych, video oraz konferencyjnych poprzez sieć pakietową. Najnowsza aktualnie jest wersja piąta protokołu przyjęta przez ITU-T w lipcu 2003 roku. Protokół ciągle jest nowelizowany, ostatnie poprawki opublikowane były w styczniu bieżącego roku[20].
H.323 jest standardem zawierającym w sobie wiele rozwiązań potrzebnych do realizacji połączenia w telefonii IP. Są wśród nich H.225.0, H.245, serie dokumentów H.450, H.460 oraz inne powiązane z usługami do kontroli multimediów. Używany jest także standard T.120, który obsługuje połączenie w czasie rzeczywistym pomiędzy dwoma lub większą ilością uczestników w konferencji[21].

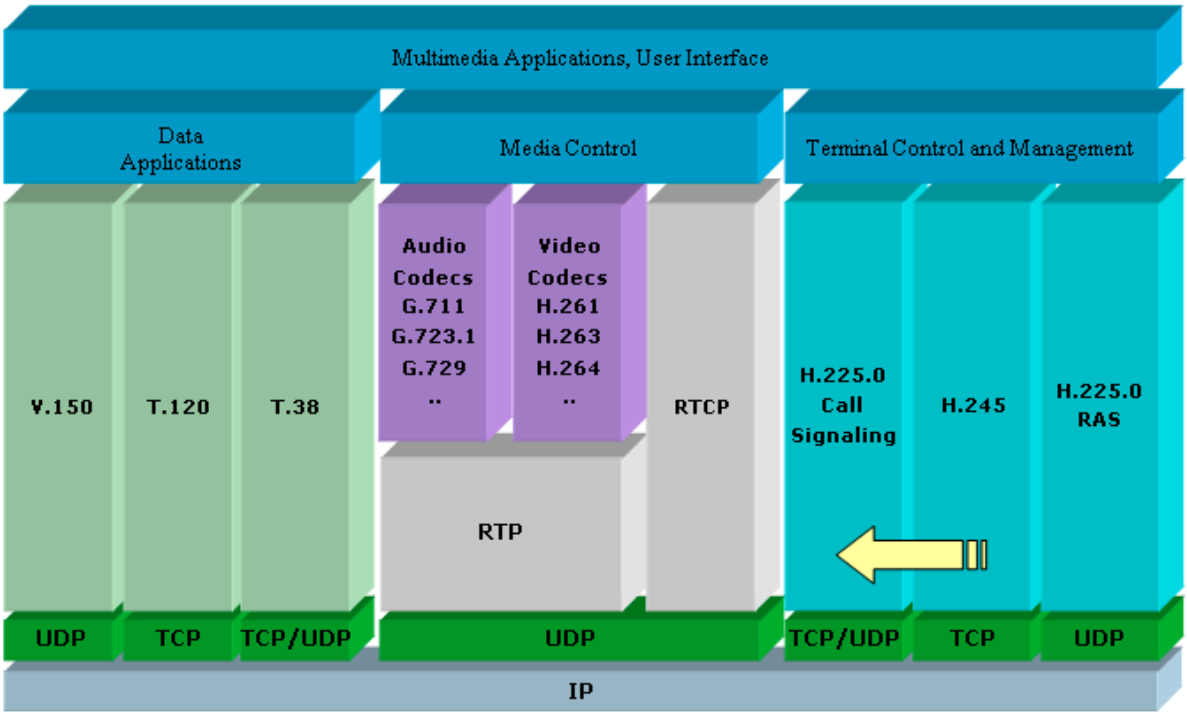
4.2 Funkcjonalność protokołu

H.323 zaprojektowany był na potrzeby komunikacji w czasie rzeczywistym przy użyciu protokołu IP. Do niedawna był liderem rynku do przenoszenia tego typu danych.
W porównaniu z wersją pierwszą, w piątej wersji wprowadzono szereg zmian i rozszerzeń. Zaimplementowano m.in.: sposób kontroli urządzeń za pomocą protokołu HTTP, dokładniejszy system billingów, system rejestracji dużej liczby użytkowników, usługę identyfikacji dzwoniącego, usługę *fast connect*, czyli szybkie łączenie polepszającą jakość usługi oraz wiele innych[21]. Ewolucja protokołu kształtuje się w stronę globalizacji dostępnych usług. W pierwszej wersji H.323 był opracowany do komunikacji poprzez sieć LAN. Kolejne wersje zapewniać zaczęły zarówno komunikację poprzez Internet oraz poprawiały szybkość nawiązywania połączeń[21]. Wersja czwarta wprowadziła GEF (Generic Extensibility Framework). Oznacza to zintegrowanie kodu bazowego dla stosu protokołów architektury H.323 i utworzenie przyjaznego interfejsu do budowania nowych jego funkcji[21]. Wersja piąta nie wprowadziła większych zmian w architekturze.
H.323 wykorzystuje oba protokoły warstwy transportowej TCP i UDP. TCP obsługuje głównie czynności kontrolne przesyłania danych oraz funkcje sygnalizacji. Protokół UDP używany jest do przesyłania danych zawierających głos, video itp.[21]. Kolejnym ważnym zadaniem H.323 jest kompresja stosowana przed formowaniem pakietów do transmisji.

Standard zawiera po kilka kodeków dla sygnału audio oraz video. Kodeki stosowane są w zależności od jakości połączenia oraz możliwości obu klientów komunikujących się ze sobą[21]. Standard H.323 dzieli sieć na strefy *zone*, są to zbiory urządzeń końcowych należących do architektury sieci protokołu zarządzane wspólnie przez urządzenie zwane strażnikiem *gatekeeper* (dokładniej omówione poniżej)[21].

4.3 Struktura stosu protokołów

Standard H.323 działa na bazie protokołów transportowych oraz sieciowych jednak jest od nich niezależny. Specyfikacja nie omawia tych protokołów. Najczęściej stosuje się podstawę TCP/IP lub UDP/IP. W dokumentacji opisane są poszczególne standardy stworzone przez ITU oraz możliwości stwarzane przez nie. Całość daję spójny układ z powodzeniem powszechnie stosowany[21]. Rysunek 4.1 przedstawia układ w jakim współpracują.



Rys. 4.1 Standardowy stos protokołów w standardzie H.323[21]

Wyróżniamy następujące standardy podlegające pod H.323:

- Kodeki Audio – pracują one w przedziale przepustowości 5,3-64 kb/s. Należą do niech:
    - G.711 - używa modulacji częstotliwości PCM (Pulse Code Modulation). Jest on popularnym kodekiem stworzonym dla telefonii analogowej.

Potrzebuje przepustowości 56 kb/s lub 64 kb/s .Z tego właśnie powodu jest mniej odpowiedni dla sieci Internet gdzie przepustowości są zazwyczaj mniejsze[23].

- G.723.1 - daje bardzo dobry stosunek jakość/pasmo, potrzebuje przepustowości od 5,3 kb/s do 6,4 kb/s. Jest standardowo zainstalowany w systemach Windows. Używane kodowanie to ACELP (Adaptive Code Excited Linear Prediction) lub MPMLQ (Multi Pulse Maximum Likelihood Quantization)[23],
- G.728 - używany jest do osiągania wysokiej jakości przy kodowaniu sygnału audio. Potrzebuje przepustowości 16 kb/s. Używa kodowania LD-CELP (Low Delay CELP)[23],
- G.729 - osiąga wysoką jakość kodowania dźwięku przy przepustowości 8 kb/s . Używa CS-ACELP (Conjugate Structure ACELP)[23],

- Kodeki Video – wyspecyfikowanych jest klika kodeków:
  - H.261 - stosowany przy przepustowości od 64 kb/s do 1920 kb/s. Wspiera dwa formaty CIF (Common Intermediate Format) o większej rozdzielczości (352x 288) oraz QCIF (Quarter CIF) osiągający 2 razy mniejszą rozdzielczość (174x144)[24],
  - H.263 - jest kodekiem nowszym od powyższego, obsługuje pięć formatów będących odmianami CIF. Zaprojektowany został, aby móc transmitować dane video przy pomocy mniejszej przepustowości[24],
  - H.264 - jest nowością powstał w 2004 roku, aneksy ukazały się w marcu bieżącego roku, nie należy więc do specyfikacji H.323 v5 lecz najprawdopodobniej zostanie on wkrótce zaaprobowany. Jest to zaawansowany kodek przeznaczony dla usług audiowizualnych[25].
- H.225.0 Call Signaling (Q.931) - protokół zajmuje się sygnalizowaniem połączeń pomiędzy urządzeniami wchodzącymi w skład architektury H.323 (punktami końcowymi *endpoints* i strażnikami *gatekeepers*)[29],
- H.225.0 RAS – Jest protokołem sygnalizacyjnym definiującym wymianę komunikatów pomiędzy punktami końcowymi *endpoints* a strażnikami *gatekeepers*, jest on potrzebny tylko wtedy, gdy w sieci istnieje strażnik i służy on do kontroli punktów końcowych przez strażnika. Skrót RAS pochodzi od Registration (rejestracja punktów końcowych przez strażnika), Admission

(kontrola dostępu do punktu końcowego) i Status (sprawdzanie stanu i lokalizacji *endpoint'u*)[29],

- H.245 używany do kontroli sygnalizacji, używa komend kontrolnych, które są wymieniane podczas połączenia, aby informować i instruować uczestników komunikacji. Działanie H.245 jest obowiązkowe na wszystkich punktach końcowych w architekturze H.323[29].

  Zadania H.245 to:

  - wymiana informacji o możliwościach transmisji np. typ medium, stosowany kodek, przepustowość itp.
  - otwieranie i zamykanie wirtualnych kanałów transmisji dla audio, video oraz danych
  - kontrola przepływu, najczęściej wymiana komunikatów podczas problemów z komunikacją

  oraz wymiana innych komunikatów kontrolnych takich jak sprawdzanie czy klient po drugiej stronie istnieje itp.
- RTP (Real-time Transfer Protocol) oraz RTCP (Real-time Control Protocol) używane do transmisji danych w czasie rzeczywistym w sieci IP (omawiane w rozdziale 1)
- T.120 stos protokołów zajmujący się wymianą danych dla komunikacji konferencyjnej (*multipoint*) w czasie rzeczywistym[29],
- T.38 protokół służący do transportu danych typu fax[29],
- V.150 protokół służący do transportu danych pochodzących z modemu stosowany podczas rozmów hybrydowych np. z sieci analogowej do użytkownika połączonego z siecią H.323[29],

Standard H.323 zawiera także inne protokoły, których nie zawarto na Rysunku 2.1

- H.450.x opisuje dodatkowe usługi dla H.323[29]
- H.235 dostarcza procesów bezpieczeństwa dla usług opartych o H.245 [29]
- H.510 rozpatruje mobilność punktów końcowych [29]

i wiele innych.

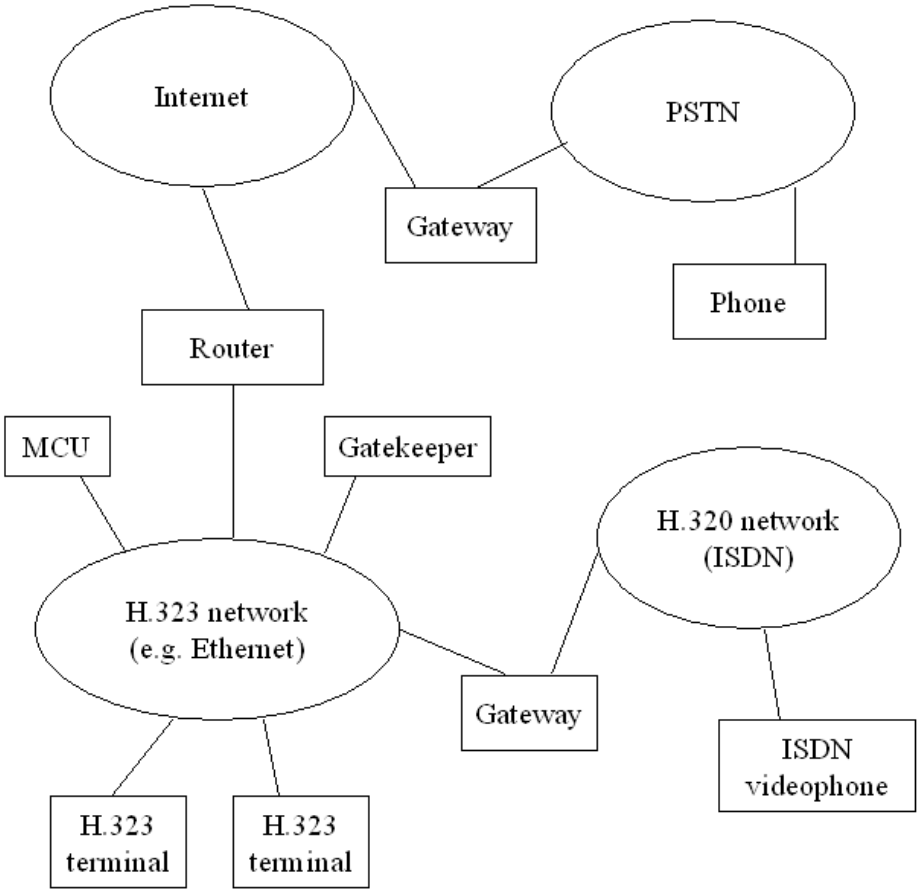
4.4 Urządzenia w standardzie H.323

Architektura sieci H.323 składa się z pięciu podstawowych elementów[28]

- Terminali *Terminals*

- Multipoint Control Units (MCUs)
- Bramek *Gateways*
- Strażników *Gatekeepers*
- Elementów brzegowych *Border Elements*

Pierwsze trzy uważane za punkty końcowe *endpoints* .



Rys. 4.2 Urządzenia w sieci systemu H.323[15]

Podstawowa konfiguracja może zawierać przynajmniej dwa terminale połączone do sieci lokalnej. Do osiągnięcia wydajnego systemu komunikacji wraz z możliwością globalnej komunikacji (sieć zewnętrzna) potrzebne są jednak urządzenia kontrolujące przebieg połączeń.

Często obserwowana jest integracja elementów logicznych tj. Strażnik, Bramka oraz MCU na jednym serwerze. Spotykana jest również sytuacja połączenia terminalu z MCU, co eliminuje potrzebę stosowania dodatkowych urządzeń przy połączeniach konferencyjnych.

4.4.1 Terminale

Zwykle to klienty takie jak:

- Telefony IP,
- Wideofony IP,
- Systemy poczty głosowej,
- Telefony programowe *Soft phones*.

Mają one możliwość inicjacji oraz odbierania zgłoszeń dotyczących komunikacji. Zdolne są także oczywiście do obsługi dwukierunkowego strumienia danych. Podstawową funkcją jest prowadzenie rozmów telefonicznych, natomiast funkcja video czy wymiany danych jest opcjonalna[29].

4.4.2 Multipoint Control Units (MCUs)

Za obsługę konferencji, czyli komunikację trzech lub więcej terminali odpowiedzialne są urządzenia MCU. Każdy terminal uczestniczący w konferencji komunikuje się najpierw z MCU. MCU zarządza zasobami konferencji np. determinuje użycie określonych kodeków do komunikacji w konferencji. MCU składa się z dwóch części:

- Multipoint Controller(MC)
- Multipoint Processors (MP)

MC odpowiada za podejmowanie decyzji o podłączeniu klienta do konferencji oraz sposobie przedstawienia go pozostałym uczestnikom. Kolejnym ważnym zadaniem jest synchronizacja klientów, aby zapewnić optymalne warunki komunikacji.
MP zajmuje się obsługą strumienia danych. W większości przypadków możliwości terminali różnią się od siebie. MP miksuje różne strumienie danych i rozprowadza je do uczestników konferencji[29].

4.4.3 Bramki

Bramka odpowiedzialna jest za połączenie sieci pakietowej z innymi rodzajami sieci. Przykładam może być połączenie sieci H.323 z siecią PSTN lub inna siecią telefonii IP np. opartą o protokół SIP. Bramka przeprowadza, więc dosyć złożone operacje. Nie tylko musi zmienić część sygnalizacyjną komunikatów, ale czasem także konwertować strumień mediów. Zapewnione oczywiście musi być działanie w czasie rzeczywistym z zachowaniem jak najmniejszych opóźnień[29].

4.4.4 Strażnicy

Są to urządzenia opcjonalne w systemie H.323, aczkolwiek bardzo użyteczne. Urządzenia te tworzą tak zwaną strefę (*zone*) i zarządzają nią. Każda strefa posiada jednego strażnika. Zgłoszenia wewnątrzstrefowe zarządzane są przez strażnika strefy. Zgłoszenia międzystrefowe obsługiwane są przez dwóch a nawet więcej strażników. Strażnik nadzoruje przebieg usług takich jak kontrola dostępnego pasma czy translacja adresu terminalu. Przyjmuje i odrzuca zgłoszenia do strefy, może także autoryzować użytkowników. Jeżeli strażnik jest obecny w sieci, terminale są zobowiązane do korzystania z niego[22].

4.4.5 Elementy brzegowe

Elementy brzegowe są często połączone ze strażnikami. Zajmują się wymianą informacji adresowych. Alokują informacje adresowe w celu zmniejszenia ruchu tych informacji między strefami (lepsze wykorzystanie pasma). Uczestniczą również w autoryzacji rozmów[22].

4.5 Sposób adresowania w sieci H.323

W tradycyjnej sieci telefonicznej przypisany jest abonentowi statyczny numer, który przechowywany jest w wielu bazach danych i odczytywany w razie potrzeby. Sieci telefonii IP w tym standard H.323 realizują adresowanie dynamiczne[26]. Jest to spowodowane możliwością zmiany np. adresu IP komputera podczas każdego restartu systemu czy też zmianami w użyciu portów do danej stacji znajdującej się za NAT[27].
W H.323 istnieje kilka sposobów adresowania, przedstawione są one w tabelce 4.1

Lp.	Typ Adresu	Format	Przykład
1	Adres IP	<0-255>.<0-255>.<0-255>.<0-255> [:<port>]	192.150.109.1 192.150.109.1:1720
2	H.323 ID	Znaki Unicode (nie więcej niż 256)	IDENTYFIKATOR
3	E.164 ID	Cyfry oraz znaki * # (nie więcej niż 128)	123456
4	e-mail	Zgodny z RFC 822 (nie więcej niż 512 znaków)	moj@mail.pl
5	URL	Nie więcej niż 512 znaków	ras://imie.domena.pl

Tab. 4.1 Typy adresów w sieci H.323

Typy adresu nr 4 oraz 5 obsługiwane są przez wersję H.323 v2 i wyższe. Terminal może być identyfikowany przez 1 adres IP oraz wiele adresów pozostałych typów. Adresy są rejestrowane przez urządzenie strażnika *gatekeeper*[26].

4.6 Przegląd operacji i schemat sesji

Standard H.323 jest używa 3 protokołów sygnalizacyjnych, które uzupełniają się nawzajem[29]. Istnieją zatem trzy grupy wymienianych komend. Są to:

- Komendy protokołu RAS
    - *Registration request*
    - *Admission request*
    - *Bandwidth request*
    - *Disengage request*
    - *Info request*

  Komendy te są kierowane do strażnika i są one potwierdzane lub odrzucane poprzez zwrot komunikatów typu:

  - *XCF* odpowiedź pozytywna
  - *XRJ* odpowiedź negatywna

  gdzie X oznacza pierwszą literę wysłanej komendy np. dla *Registration request* potwierdzeniem jest *RCF*

- Komendy protokołu H.245 [29]
    - *Master-slave determination*
    - *Terminal capability set*
    - *Open logical channel*
    - *Close logical channel*
    - *Request mode*
    - *Send terminal capability set*
    - *End session*

  Komendy wymieniane pomiędzy terminalami w celu ustalenia parametrów, rozpoczęcia i zakończenia sesji kanałów multimedialnych. Dozwolonymi odpowiedziami na komendy mogą być:

  - *Acknowledge,*

- *Reject,*
- *Confirm,*
- *Release.*

- Komendy protokołu H.225.0 Call Signaling (Q.931) [29]
  - *Alerting*
  - *Call proceeding*
  - *Connect*
  - *Setup*
  - *Realase*

  Komendy są nieodzowne do zapoczątkowania procesu połączenia. Najczęściej występują na samym początku oraz na końcu komunikacji.

4.7 Przykładowe połączenie dwóch terminali

W tabeli 4.2 przedstawione są kolejne komendy realizowane podczas prostego połączenia dwóch terminali bez pośrednictwa strażnika (*gatekeeper'a*) [29]

Nr kolejny komendy	Terminal A	Terminal B	Pochodzenie komendy
1	*Setup*		Q.931
2		*Alerting*	Q.931
3		*Connect*	Q.931
4	*termCapSet*		H.245
5		*termCapAck*	H.245
4		*termCapSet*	H.245
5	*termCapAck*		H.245
6	*masterSlvDet*		H.245
7		*masterSlvDetAck*	H.245
8	*masterSlvDetConfirm*		H.245
9	*openReq*		H.245
10		*openAck*	H.245
9		*openReq*	H.245
10	*openAck*		H.245
11	*endSession*		H.245
11		*endSession*	H.245
12	*Release*		Q.931

Tab. 4.2 Wymiana komunikatów pomiędzy terminalami podczas komunikacji

Inicjacja połączenia odbywa się za pośrednictwem komendy *Setup*. Terminal B odpowiada komendą *Alerting,* co oznacza, że powiadamia on osobę docelową dzwonieniem. Gdy

połączenie zostanie zaakceptowane Terminal B wysyła komendę *Connect*. Stan wymiany danych multimedialnych obu terminali poprzedzony jest ustaleniem parametrów połączenia. Oba terminale wysyłają komendy *termCapSet*, aby przekazać je współuczestnikowi transmisji. Potwierdzenie otrzymania parametrów realizowane jest komunikatem *termCapAck*. Następnie wybierany jest terminal nadrzędny w transmisji, służą do tego komunikaty *masterSlvDet* i *masterSlvDetAck*. Kolejnym krokiem jest otwarcie kanału logicznego przez terminal Master oraz otwarcie kanału logicznego przez terminal Slave w drugim kierunku. Używane są do tego komunikaty *openReq* oraz *openAck*. Komendy *endSession* determinują zakończenie połączenia. Ostatnim komunikatem jest *Release*[29].

Przedstawiony przykładowy przebieg sesji jest bardzo prostym przypadkiem. Omija on komunikację ze strażnikiem. Jeżeli strażnik istnieje w sieci to komunikacja musi odbywać się za jego pośrednictwem. Terminale komunikują się z nim protokołem H.225.0 RAS. Od wersji czwartej H.323 może także ustanawiać sesję w trybie *fast connect*, który pozwala ominąć komunikaty w tabelce wpisane na szarym tle. Pozwala to oczywiście na zwiększenie szybkości nawiązywania połączeń [29].

Rozdział 5. SIP

5.1 Wstęp

SIP jest protokołem warstwy aplikacji zajmującym się kontrolą sesji multimedialnych. Skutecznie konkuruje ze standardem H.323, co jest widoczne w zestawieniach dostępnych urządzeń i produktów obsługujących te standardy[32]. Stworzony został początkowo do multimedialnych transmisji rozsiewczych takiej jak telewizja internetowa czy też radio internetowe[37]. Szybko jednak zauważono jego możliwości zastosowania dla telefonii IP. Jego rozwój kierował się właśnie pod tym kątem. Jak dotąd wykorzystywany jest z powodzeniem w zakresie telefonii pakietowej a prognozy zapowiadają, że stanie się liderem[37]. Opracowany został przez grupę roboczą IETF MMUSIC (Mutliparty Multimedia Session Control). SIP został zrealizowany na podstawie wcześniej znormalizowanych standardów IETF[35]. Dzięki temu zyskuje elastyczność oraz skalowalność.

5.2 Funkcjonalność protokołu SIP

SIP jest protokołem warstwy aplikacji sterującym kluczowymi czynnościami dla uzyskania połączenia. Zestawia sesję między dwoma lub wieloma użytkownikami. Może modyfikować sesję oraz ją zakończyć. Sesja jest rozumiana jako połączenie telefoniczne użytkowników, jako konferencja multimedialna lub telefoniczna, a także jako transmisja rozsiewcza sygnału multimedialnego. Oznacza to, że sesja może być nawiązywana w trybie unicast lub multicast. W trakcie sesji istnieje możliwość zmiany parametrów połączenia tzn. możne być dołączony nowy uczestnik do rozmowy lub też nastąpić zmiana wykorzystywanych kodeków[35]. Protokół SIP w sposób klarowny obsługuje mobilność użytkowników, co w sieciach PSTN było co najmniej bardzo trudne do zrealizowania. Pojęcie mobilności rozumiane jest jako możliwość inicjowania, odbierania połączeń oraz dostępu do własnego profilu w każdym miejscu po zalogowaniu. Logowanie może się odbywać z każdego terminala podłączonego do sieci obsługującego protokół SIP. Mobilność w SIP, musi zatem zapewniać usługę identyfikacji w sieci użytkowników zmieniających swoje położenie. Standard zajmuje się tym, aby nowo zalogowanego użytkownika odnaleźć w sieci i ewentualnie nawiązać z nim połączenie[35]. Protokołem podkładowym w warstwie transportu dla SIP jest zazwyczaj

UDP. TCP także może wspierać SIP, lecz stosowane jest to wtedy, gdy komunikaty SIP nie mieszczą się w jednym pakiecie UDP[33] (największy pakiet UDP to 65 535 bajtów, ze względu na 16 bitowe pole długości pakietu[34]). SIP może stosować również inne rozwiązania np. protokół SCTP (Stream Control Transmission Protocol) (zob. 5.12.2). Jest to możliwe, gdyż SIP nie jest związany z żadnym protokołem niższej warstwy[33].

5.3 Struktura stosu protokołów dla SIP

SIP korzysta ze standardów opracowanych przez IETF. Do architektury należą następujące protokoły używane w sygnalizacji:

- SDP (Session Description Protocol)[48],
- SAP (Session Announcement Protocol)[49],

oraz protokoły używane przy transporcie danych multimedialnych:

- RTP (Real-time Protocol)[4],
- RTSP (Real-time Streaming Protocol)[12],
- RSVP (ReSerVation Protocol)[9],

Ponadto w stosie SIP ważną rolą odgrywa popularny w Internecie DNS[19] (Domain Name System).
Protokoły RTP, RTSP i RSVP omówione zostały w Rozdziale 3.

SDP
służący do opisu sesji multimedialnych[48], szczegóły na temat protokołu znajdują się w punkcie 5.11

SAP
służący do ogłaszania istnienia multicastowej sesji multimedialnej[37],

DNS
Protokół ten tak powszechnie używany w Internecie jest także ściśle związany z architekturą SIP. Wspomaga on system adresacji użytkowników sieci SIP. Użytkownicy posiadają identyfikatory w postaci wskaźników URI (Uniform Resource Indicator), które mają format podobny do adresów e-mail lub nawet postać numeru telefonu[35]. Determinuje to konieczność użycia systemu DNS do przekształcania nazw hostów i domen na adresy IP. Użycie URI jako klucza do identyfikacji użytkowników bardzo pozytywnie wpływa na

możliwości rozwoju SIP. Umożliwia łatwą integracje z systemami korzystającymi z URI, które jest powszechnie stosowane w sieci Web[35].
System DNS nie tylko współpracuje ze standardem URI. Wspiera on także systemy telefoniczne. Rozszerzenie DNS o nazwie ENUM (Electronic Numbering System) wspiera powszechny standard numeracji E.164, który jest standardem organizacji EITF i został zatwierdzony przez organizację ITU (International Telecommunications Union)[38].

5.4 Elementy architektury SIP

SIP przewiduje dwa typy urządzeń działających w sieci[35]. Są nimi :

- Terminale – urządzenia końcowe, działające po stronie użytkownika, nazywane również User Agents (UA), są to zazwyczaj aplikacje (softphones) lub dedykowane urządzenia podobne do zwykłych telefonów,
- Serwery – prowadzące różne usługi w celu łatwej komunikacji w sieci Internet

5.4.1 Terminale (UA)

Terminale są to elementy pośredniczące w komunikacji między siecią a użytkownikiem. Posiadają one dużą funkcjonalność i oferują możliwości konfiguracyjne. UA w standardzie SIP składają się z dwóch części:

- UAC (User Agent Client) – jest to część kliencka UA generująca komunikaty np. inicjująca połączenie[35],
- UAS (User Agent Server) – jest to część serwerowa UA oczekująca na komunikaty od innych urządzeń w sieci np. na połączenie od innego klienta oraz odpowiadająca na komunikaty tych urządzeń bazując na: reakcji użytkownika, rezultacie wykonania programu czy innych mechanizmach [35].

Struktura budowy UA dzielona na klienta i serwer powoduje, że może on sobie radzić bez dodatkowych urządzeń w procesie prostej komunikacji. Terminal generuje komunikaty i potrafi odpowiadać na prośby innego terminala. Gdy UAC wyśle żądanie może ono przejść przez kilka serwerów należących do architektury SIP, które przekażą dalej to żądanie do odpowiedniego UAS. Gdy UAS wyśle odpowiedź, pójdzie ona w stronę UAC[35].

5.4.2 Serwery

Serwery w sieci SIP są urządzeniami wspomagającymi komunikację między terminalami. Minimalna wymagana konfiguracja może wcale nie zawierać serwerów. Wyróżniane są trzy rodzaje serwerów:

- **Serwer Proxy (delegat)** – którego zadaniem jest znajdowanie drogi połączenia między urządzeniami końcowymi. Przekazuje on zgłoszenia SIP od UAC do następnych serwerów (włączając w to UAS) monitując zgłoszenie. Zgłoszenia mogą przechodzić przez kilka serwerów, a ich liczba zależy od konfiguracji i struktury sieci. Serwery Proxy monitorują stan zgłoszenia. Przekazywanie zgłoszeń jest głównym zadaniem tego typu serwera. Serwer Proxy realizuje funkcje takie jak:
  - uwierzytelnianie i autoryzacja użytkowników
  - kontrola dostępu do sieci (może realizować określoną politykę)[35],
- **Serwer Redirect (przekierowujący)** – którego zadaniem jest znajdowanie drogi połączenia urządzeń końcowych (podobnie jak serwera Proxy), lecz działa na zasadzie odpowiedzi do urządzenia, które wygenerowało zgłoszenie. Serwer przekierowujący po odebraniu żądania od UAC nie przesyła go dalej, ale odpowiada na nie, zwracając adres kolejnego serwera, z którym należy się skontaktować w celu poszukiwania właściwego serwera, czyli UAS. UA wysyła ponownie zaproszenie do sesji wykorzystując do tego uzyskany adres. Znaleziony w ten sposób serwer może być znów serwerem Redirect, serwerem Proxy lub właściwym serwerem użytkownika końcowego UAS. Serwer Redirect zapomina o zgłoszeniu zaraz po jego przetworzeniu[35].
- **Registrar (rejestracyjny)** – którego zadaniem jest rejestracja bieżącej lokalizacji UAC. Gdy użytkownik loguje się na terminalu w innej niż dotychczas lokalizacji, Registrar zapisuje tą lokalizację i właściwości użytkownika[35].

W praktyce często spotyka się serwer SIP łączący wszystkie omówione wyżej funkcje, aczkolwiek nie jest to koniecznością. Serwery mogą być oddzielnymi procesami na różnych maszynach[32].

Spotyka się również definicje dla czwartego rodzaju serwera architektury SIP. Mowa tu o **serwerze lokalizacji (Localization Server)**. Jest to serwer, który zajmuje się tylko i wyłącznie przechowywaniem informacji. Są to wszelkie informacje rejestracyjne. Stosowany wówczas, gdy istnieje potrzeba przechowywania dużej ilości danych. Ma on charakter serwera bazo-danowego. Konkretny serwer lokalizacji zazwyczaj współpracuje tylko z

serwerami tej samej domeny (węzła), gdyż komunikacja z nim nie zawiera się w standardzie SIP i odbywać się może różnymi sposobami np.za pomocą dedykowanych protokołów[36].

5.5 Adresacja

SIP posiada funkcjonalność rejestracji globalnie rozróżnialnego adresu. Adresy SIP mogą wyglądać podobnie do adresów poczty elektronicznej lub nawet jak numery telefonu[35]. Adres składa się z nazwy użytkownika, którą może zastąpić numer, oraz z domeny sieci, w której znajduje się serwer SIP. Adresem może być także numer telefoniczny z sieci PSTN połączony z adresem bramki przekształcającej format SIP na format PTSN[36].
Ogólna składnia „sip URI" przedstawia się następująco:

sip:user:password@host:port;uri-parameters?header

oczywiście najczęściej wszystkie składowe nie są wykorzystywane[36].
Przykładowe adresy w sieci SIP mogą wyglądają następująco[36]:

- sip:michal.sloma@tel_domain.org – adres wykorzystujący nazwę użytkownika,
- sip:+8185551234@bramka.pl – adres w schemacie E.164,
- sip:+8185551234 – adres E.164, nie zawierający domeny,
- sip:michal@tel_domain.org:5090 – specyfikuje dodatkowo numer portu,
- sip:michal:haslo@tel_domain.org – specyfikuje dodatkowo hasło,
- sip:michal@tel_domain.org;transport=tcp – specyfikuje metodę transportu za pomocą protokołu TCP,
- sip:tel_domain.org;method=REGISTER?to=michal%40tel_domain.org – rejestruje adres dla użytkownika michal używając żądania (Method) REGISTER w domenie tel_domain.org (żądania protokołu SIP omówione zostały poniżej).

W grudniu 2004 roku przyjęto RFC 3966 zatytułowane „The tel URI for Telephone Numbers"[39]. Omawia ono stosowanie wskaźników „tel URI" w sieci SIP postaci np.

- `tel:+1-212-555-0101`
- `tel:1234;phone-context=munich.example.com`

„tel:" - oznacza sposób interpretacji dalszej części URI

“phone-context=munich.example.com” – dodatkowy znacznik stosowany w sieci SIP oznaczający serwer Proxy SIP.
Składnia URI postaci „tel:” zawiera w sumie dwadzieścia siedem znaczników co pozwala dokładnie określić adresata i jego parametry[39].

5.6 Rodzaje wiadomości w SIP

Forma wiadomości w SIP jest bardzo zbliżona do wiadomości wysyłanych przez SMTP[46] czy HTTP[47]. Wyróżniamy dwa typy wiadomości[42]:

- request (żądanie)
- response (odpowiedź)

Wiadomości używają standardowego formatu dla wiadomości (message) opisanego w RFC 822[43].

Wiadomość w SIP ma następującą postać[42]:

- start-line – request-line dla REQUEST oraz status-line dla RESPONSE
- Header – nagłówek, możliwe są różne rodzaje (general-header, request-header, entity-header)
- Linia pusta
- Body – opcjonalne wnętrze żądania

5.6.1 Request

Żądanie ma następującą postać[42].
reqest-line – linia ta zawiera nazwę żądania, adresata żądania oraz protokół i jego wersję
Będzie to np.:
INVITE sip:chris@sipcenter.com SIP/2.0
header – nagłówek żądania składa się z kilku linii i mogą to być[42]:

- CSeq – określa numer sekwencyjny w wymianie dla **określonego** komunikatu, np. jeśli aplikacja chce wysłać żądanie INVITE i z jakichś przyczyn nie zostanie ono zaakceptowane (np. odpowiedź z kodem 4xx) lub nie ma na nie odpowiedzi, następne żądania INVITE tej aplikacji będą zawierały kolejne numery sekwencyjne,
- To – określa adresata pakietu. Adres powinien mieć prawidłowy format adresacji w SIP (czyli sip URI), co zostało omówione powyżej,

- From – określa nadawcę pakietu. Adres powinien mieć format sip URI,
- Call-ID – jest to unikalny identyfikator zaproszenia do rozmowy oraz rejestracji dla określonego klienta,
- Via – pole wskazuje ścieżkę przebytą jak dotąd przez żądanie, zapobiega to zapętlaniu się żądań oraz zapewnia możliwość przebycia tej samej drogi przez odpowiedź na żądanie, co rozwiązuje problemy z firewall'ami czy też innymi niecodziennymi sytuacjami z rutowaniem,
- Contact – pole może się pojawić w żądaniach takich jak INVITE, ACK i REGISTER oraz w odpowiedziach 1xx, 2xx, 3xx i „485 Ambiguous" (oznaczający niejasne żądanie[42]). Determinuje ono URL gdzie użytkownik może być dostępny dla celów dalszej komunikacji[42].
- Content-Type – determinuje zawartość **body** żądania poprzez wskazanie typu mediów tam zawartych. Przykłady to:
    - Content-Type: application/sdp
    - Content-Type: text/html; charset=ISO-8859-4
- Content-Length – determinuje długość **body** przedstawioną w postaci dziesiętnej liczby oktetów
- Max-Forwards – wartością pola jest liczba dziesiętna oznaczająca ile razy żądanie może zostać przekazane do następnego serwera, jest to zabezpieczenie przed pętlami,
- Expires – wartość pola determinuje datę i czas, po której żądanie wygasa i staje się nieważne. Wartość może pojawiać się w dwóch formatach: SIP-date lub dziesiętna liczba sekund np.:
    - Expires: Thu, 01 Dec 1994 16:00:00 GMT
    - Expires: 360

  Pole używane jest w żądaniach REGISTER oraz INVITE
- Authorization – pole służy do autoryzacji UA w serwerze, jest ono opcjonalne. Zazwyczaj jest ono dodawane, gdy UA dostanie odpowiednią prośbę w odpowiedzi (response) na żądanie niezawierające tego pola. Pole składa się z informacji takich jak użytkownik, domena, oraz informacje potrzebne do autoryzacji.

Dokładny wykaz pól, które mogę się znaleźć w nagłówku żądania zawiera tabela 3.1

linia pusta – jest to linia służąca do oddzielenia żądania od informacji zupełnie innej postaci. Postać ciała żądania może być bardzo różna, mogą to być także dane binarne.

body – treść żądania, często jest pusta. Może przenosić dane zapisane w innym protokole lub dane bezpośrednio przeznaczone do odczytania przez użytkownika. Treść może być wykorzystana np. do przenoszenia informacji protokołu SDP. Zawartość Body, czyli co będzie się znajdować w treści komunikatu determinuje specjalne pole nagłówka (Header) o nazwie Content-Type.

5.6.2 Response

Ma następującą postać[42]:

status-line – linia ta zawiera nazwę i wersję protokołu oraz numer i odpowiedź na żądanie. Przykładowa linia status-line to:

SIP/2.0 200 OK

header – nagłówek w odpowiedzi (response) jest podobny do nagłówka żądania. Może on zawierać pewne pola, które nie mogą znajdować się w nagłówku żądania. Dokładnie pokazane jest to w tabeli 5.1,

linia pusta – podobnie jak w żądaniu oddziela nagłówek od ciała odpowiedzi (response),

body – treść odpowiedzi, zawierać może takie same informacje jak w żądaniu

Nazwa pola nagłówka	Występowanie
Accept	Request, 415
Accept-Encoding	Request, 415
Accept-Language	Request, 415
Allow	200, 405
Authorization	Request
Call-ID	Request, Response
Contact	Request, 1xx, 2xx, 3xx, 485
Content-Encoding	Request, Response
Content-Length	Request, Response
Content-Type	Request, Response
CSeq	Request, Response
Date	Request, Response
Encryption	Request, Response
Expires	Request, Response
From	Request, Response
Hide	Request
Max-Forwards	Request
Organization	Request, Response
Proxy-Authenticate	407
Proxy-Authorization	Request
Proxy-Require	Request
Priority	Request
Require	Request
Retry-After	Request, 404, 480, 486, 503, 600, 603

Response-Key	Request
Record-Route	Request, 2xx
Route	Request
Server	Response
Subject	Request
Timestamp	Request, Response
To	Request, Response
Unsupported	420
User-Agent	Request, Response
Via	Request, Response
Warning	Response
WWW-Authenticate	401

Tabela 5.1 Pola dozwolone do użycia w nagłówkach pakietów protokołu SIP[42]

5.6.3 Przykładowe komunikaty protokołu SIP

Układ komunikatów SIP na przykładzie żądania INVITE oraz odpowiedzi Trying.

INVITE jest rodzajem żądania, czyli ma formę **Request**

Informacje w treści (body) zawierają dane dla w formacie protokołu SDP (zob. 5.11) i służą do ustanawiania sesji multimedialnej.

```
INVITE sip:chris@sipcenter.com SIP/2.0          <- request-line
CSeq: 1 INVITE                                  <- header
To: sip:chris@sipcenter.com
From: sip:fred@10.20.30.40
Call-ID: 652302859179069176O@10.20.30.40
Via: SIP/2.0/UDP 10.20.30.40
Contact: < sip:fred@10.20.30.40 >
Content-Type: application/sdp
Content-Length: 123
                                                <- linia pusta
v=0                                             <- body
o=- 982769551076 982769551076 IN IP4 10.20.30.40
s=-
c=IN IP4 10.20.30.40
t=0 0
m=audio 5004 RTP/AVP 8 3 0
```

Trying jest odpowiedzią, czyli **Response**

```
SIP/2.0 100 Trying                              <- status-line
To: sip:chris@sipcenter.com                     <- header
From: sip:fred@10.20.30.40
CSeq: 1 INVITE
Call-ID: 652302859179069176O@10.20.30.40
Via: SIP/2.0/UDP 10.20.30.40
Content-Length: 0
                                                <- linia pusta
```

5.7 Rodzaje żądań (Methods)

SIP wykonuje różne polecenia użytkownika wykorzystując do tego kilka specyficznych żądań. Zachowują one oczywiście konwencje wiadomości (message) omówioną powyżej. Żądania są generowane przez terminal UAC i przesyłane do serwera, który je interpretuje. Wywołują one reakcje serwera, który podejmuje określone akcje.
Wyróżniamy następujące żądania protokołu SIP[42]:

INVITE
Żądanie wysyłane jest w celu nawiązania połączenia przez UAC z UCS innego użytkownika. Stanowi zaproszenie do sesji.

ACK
Potwierdza otrzymanie odpowiedzi na zaproszenie INVITE

BYE
Żądanie wysyłane w celu zakończenia sesji lub braku chęci dalszego uczestniczenia, jeśli jest to konferencja

CANCEL
Żądanie wysyłane w celu rezygnacji z czekania na ustanowienie sesji

REGISTER
Żądanie wysyłane do serwera REGISTRAR w celu poinformowania o bieżącej lokalizacji

OPTIONS
Służy do uzyskania informacji o możliwościach serwera odbierającego

INFO
Żądanie dodane później, nie istniejące jeszcze w pierwszym RFC o SIP czyli RFC 2543[42]. Transportuje ono informacje o sesji takie jak obrazek czy tony DTMF. Opisane osobno w RFC 2976[44]

5.8 Kody odpowiedzi (Response) w SIP

Podobnie jak w wielu innych protokołach standard SIP posiada opracowane kody przesyłane w odpowiedziach (Response) determinujące stan połączenia i sesji. Kody protokółu SIP zbliżone są do kodów stosowanego bardzo powszechnie w Internecie HTTP/1.1
Znaczenie kodów jest następujące[42]:

1xx *ODPOWIEDZI TYMCZASOWE*

uzyskanie w odpowiedzi takiego kodu oznacza, że żądanie (request) zostało odebrane przez docelowy serwer i jest w stanie przetwarzania

Przykłady:

100 Trying

180 Ringing

2xx *ODPOWIEDZI POMYŚLNE*

odpowiedź z kodem zaczynającym się dwójką oznacza, że żądanie zostało zrozumiane i zaakceptowane

Przykłady:

200 OK

3xx *ODPOWIEDZI PRZEKIEROWAŃ*

odpowiedzi tego typu uzyskujemy, gdy potrzebna jest jeszcze jakaś procedura, aby dopełnić wykonanie żądania

Przykłady:

300 Multiple Choices

301 Moved Permanently

302 Moved Temporarily

4xx *ODPOWIEDZI BŁĘDÓW ŻĄDAŃ*

odpowiedź taka oznacza, że składnia żądania (request) jest niepoprawna lub serwer nie może zrealizować tego żądania

Przykłady:

400 Bad Request

401 Unauthorized

402 Not Found

5xx *ODPOWIEDZI BŁĘDÓW SERWERA*

odpowiedzi takie mówią, że serwer nie może wykonać żądania permanentnie

Przykłady:

500 Server Initial Error

501 Not Implemented

6xx *ODPOWIEDZI BŁĘDÓW GLOBALNYCH*

odpowiedź taka informuje o poważnym błędzie, żądanie nie będzie zrealizowane przez żaden serwer.

Przykłady:

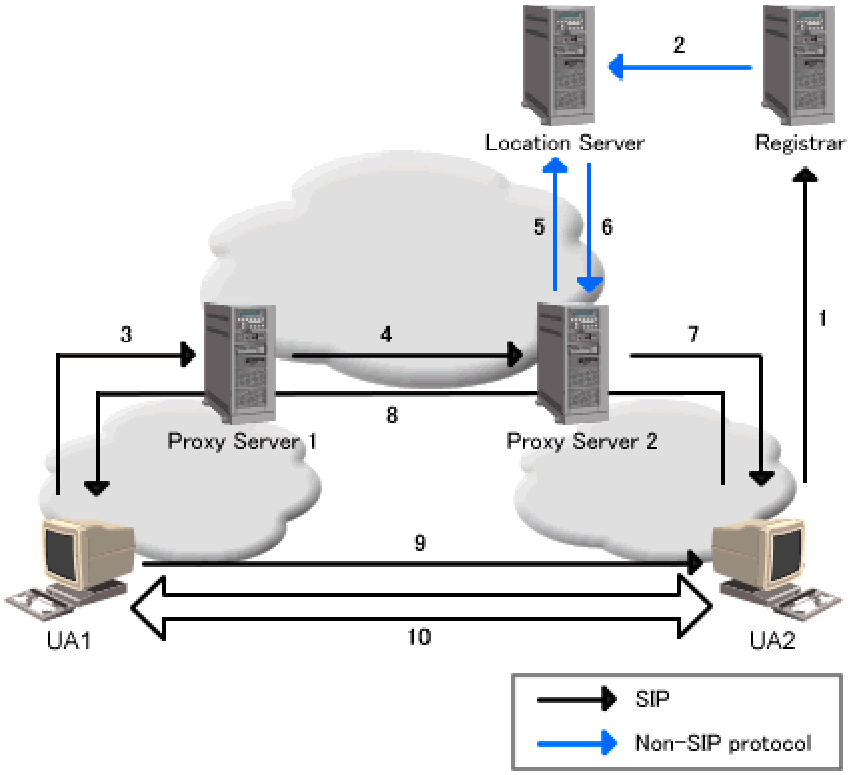
600 Busy Everywhere

603 Decline

604 Doesn't Exist

5.9 Przebieg sesji w standardzie SIP

Poniżej zostaną pokazane przykładowe połączenia dwóch użytkowników korzystających z terminali UA1 (User Agent) i UA2. Rysunek 5.1 przedstawia schemat sesji, w której uczestniczą dwa terminale, serwery Proxy, serwer Registrar oraz serwer Lokalizacji. Zakładamy, że połączenie chce nawiązać UA1 z terminalem UA2.



Rys. 5.1 Schemat sesji SIP. UA1 nawiązuje udane połączenie z UA2[40]

Ogólny opis przebiegu sesji z rysunku 5.1 [36].

1. UA2 loguje się i od razu wysyła komunikat do serwera Registrar, aby zarejestrować swoją lokalizację. Komunikat z żądaniem REGISTER.

```
REGISTER sip:10.20.30.40 SIP/2.0
To: < sip:chris@sipcenter.com >
From: < sip:chris@sipcenter.com >
Call-ID: -6342858789046433O5@140.150.160.70
CSeq: 1 REGISTER
Via: SIP/2.0/UDP 140.150.160.70:5060
Contact: < sip:chris@140.150.160.70 >
Expires: 3600
Content-Length: 0
```

2. Serwer Registrar współpracuje z serwerem Lokalizacji i zapisuje dane, które uzyskał od UA2 w zasobach serwera Lokalizacji. Operacja ta nie jest wspierana przez standard SIP.
3. UA1 chce zainicjować połączenie wysyłając żądanie INVITE do UA2, które przekazuje do najbliższego serwera, w tym przypadku jest to serwer Proxy.

```
INVITE sip:chris@sipcenter.com SIP/2.0
CSeq: 1 INVITE
To: sip:chris@sipcenter.com
From: fred< sip:fred@10.20.30.40 >
Call-ID: 652302859179069176O@10.20.30.40
Via: SIP/2.0/UDP 10.20.30.40
Contact: < sip:fred@10.20.30.40 >
```

```
Subject: no subject
Content-Type: application/sdp
Content-Length: 123
```

Treść komunikatu została pominięta.

4. Serwer Proxy przekazuje komunikat do następnego serwera Proxy, który według niego może wiedzieć o lokalizacji UA2. Serwer dodaje do nagłówka pakietu kolejne pole *Via* zawierającym jego adres.

```
INVITE sip:chris@sipcenter.com SIP/2.0
CSeq: 1 INVITE
To: sip:chris@sipcenter.com
From: fred< sip:fred@10.20.30.40 >
Call-ID: 652302859179069176O@10.20.30.40
Via: SIP/2.0/UDP 20.20.30.40
Via: SIP/2.0/UDP 10.20.30.40
Contact: < sip:fred@10.20.30.40 >
Subject: no subject
Content-Type: application/sdp
Content-Length: 123
```

Treść komunikatu została pominięta.

5. Drugi Serwer Proxy wysyła pytanie (nie są to komunikaty SIP) o lokalizację UA2 do serwera Lokalizacji.
6. Serwer lokalizacji znajduje w swoich zasobach lokalizację UA2 i przesyła ją do Drugiego serwera Proxy.
7. Serwer Proxy dostarcza żądanie INVITE do UA2 dodając kolejne pole *Via*.
8. UA2 odpowiada na żądanie UA1, które wraca tą samą drogą co przyszło tzn. przez dwa serwery Proxy. Droga powrotna determinowana jest przez pola *Via*.

```
SIP/2.0 200 OK
To: sip:chris@sipcenter.com;tag=-994822506
From: fred< sip:fred@10.20.30.40 >
CSeq: 1 INVITE
Call-ID: 652302859179069176O@10.20.30.40
Via: SIP/2.0/UDP 10.20.30.50:5060;branch=C1C334F113C4000000E4D1911E77*4*0*0
Via: SIP/2.0/UDP 10.20.30.40
Contact: < sip:chris@10.20.30.60 >
Content-Type: application/sdp
Content-Length: 101
```

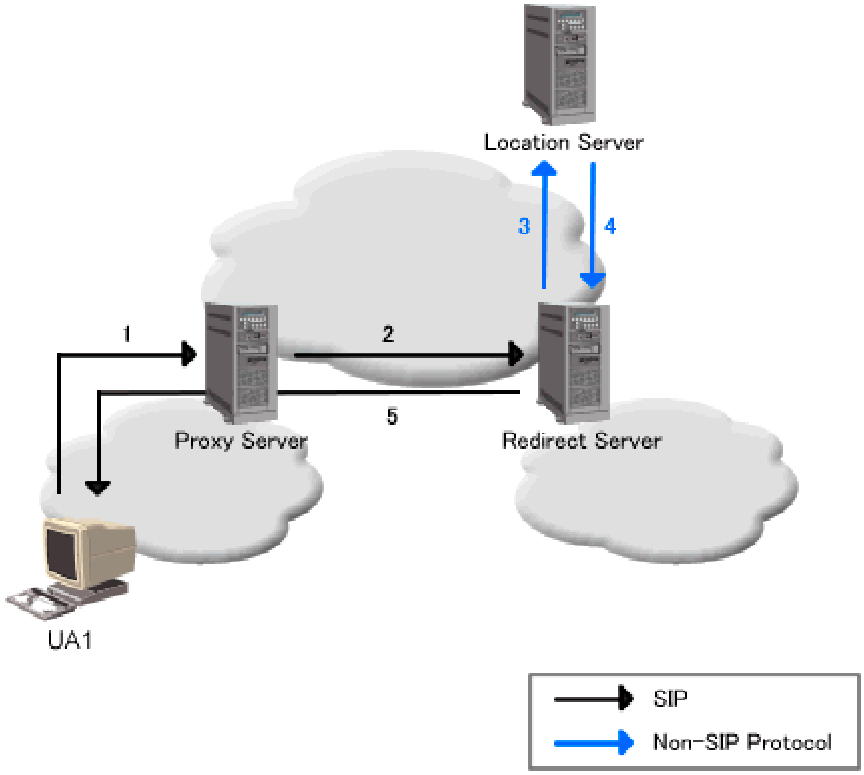
Treść komunikatu została pominięta.

9. UA1 potwierdza otrzymanie odpowiedzi wysyłając już bezpośrednio do UA2 (nie do najbliższego serwera Proxy) żądanie ACK.

```
ACK sip:chris@10.20.30.60 SIP/2.0
CSeq: 1 ACK
To: sip:chris@sipcenter.com;tag=-994822506
From: fred< sip:fred@10.20.30.40 >
Call-ID: 652302859179069176O@10.20.30.40
Via: SIP/2.0/UDP 10.20.30.40
Content-Length: 0
```

10. Komunikacja zostaje nawiązana na poziomie protokołów transportu.

Sytuacja nie musi być od razu taka klarowna jak na rysunku 5.1 oraz w nawiązanie połączenia może być zaangażowany serwer Redirect. Schemat sesji, w którym uczestniczy serwer Redirect przedstawiony jest na rysunku 5.2



Rys 5.2 Schemat sesji SIP. UA1 nie zna lokalizacji UA2[40]

Ogólny opis sesji przedstawionej na rysunku 5.2 [36]

1. UA1 chce zainicjować połączenie wysyłając żądanie INVITE do UA2, które przekazuje do domyślnego serwera Proxy. Żądanie INVITE ma postać dokładnie taką jak w przypadku rozważanym powyżej.
2. Serwer Proxy przekazuje komunikat do następnego serwera. Jest to serwer Redirect.
3. Serwer Redirect sprawdza informacje na temat UA2 w zasobach znanego sobie serwera lokalizacji.
4. Serwer lokalizacji zwraca informacje na temat lokalizacji UA2
5. Serwer Redirect wysyła komunikat z informacją o lokalizacji UA2 do UA1.

5.10 Szczegółowa wymiana komunikatów SIP podczas przebiegu sesji

Poniżej przedstawione są dokładne przebiegi sesji podczas rejestrowania UA do serwera Registrar oraz podczas rozpoczynania rozmowy.

5.10.1 Rejestracja UA w serwerze Registrar

Żądanie REGISTER jest niezbędne we wszystkich rozwiązaniach opartych o serwer Proxy. Serwer Registrar odgrywa rolę jako pierwszy i zajmuje się mechanizmami lokalizacji użytkowników[40].

Rys. 5.3 Schemat wymiany komunikatów podczas rejestracji w serwerze Registrar[40]

UA chcąc przeprowadzić rejestracje powinien wygenerować żądanie REGISTER, które zostanie przesłane do serwera Registrar. Żądanie powinno mieć format jak poniżej [40].

```
REGISTER sip:10.20.30.40 SIP/2.0
To: < sip:chris@sipcenter.com >
From: < sip:chris@sipcenter.com >
Call-ID: -6342858789046433O5@140.150.160.70
CSeq: 1 REGISTER
Via: SIP/2.0/UDP 140.150.160.70:5060
Contact: < sip:chris@140.150.160.70 >
Expires: 3600
Content-Length: 0
```

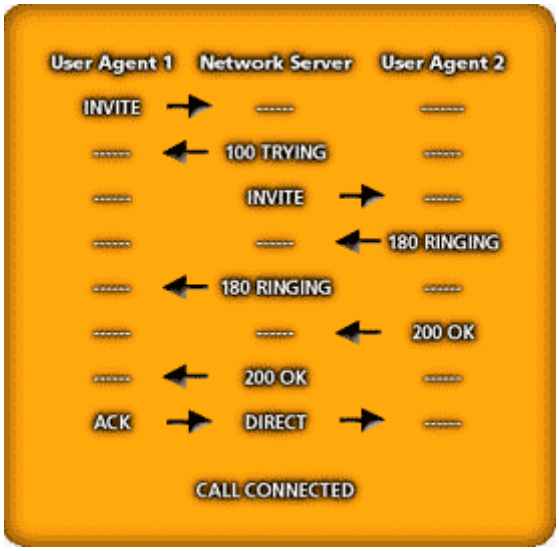
Serwer Registrar po otrzymaniu poprawnego komunikatu REGISTER generuje odpowiedź 200 OK i przesyła ją z powrotem do UA

```
SIP/2.0 200 OK
To: < sip:chris@sipcenter.com >;tag=C1C334F113C4000000E4D0D1F6F1
From: < sip:chris@sipcenter.com >
CSeq: 1 REGISTER
Call-ID: -6342858789046433O5@140.150.160.70
Via: SIP/2.0/UDP 140.150.160.70:5060
Contact: < sip:chris@140.150.160.70 >;action=proxy;expires=3599
Expires: Wed, 21 Feb 2001 13:57:13 GMT
Content-Length: 0
```

Po otrzymaniu takiej odpowiedzi UA może uznać, że jest zalogowany do serwera i jest dostępny dla innych użytkowników.

5.10.2 Podstawowa sesja nawiązująca połączenie głosowe

Rozpoczęcie sesji w celu nawiązania połączenia zawsze zaczyna się żądaniem INVITE wysłanym przez UA1 do domyślnego serwera Proxy, serwera, w którym jest zarejestrowany UA2, lub też bezpośrednio do UA2[40]. Poniżej przedstawiona jest sesja nawiązująca połączenie, w której uczestniczy jeden serwer Proxy.
UA1 jest terminalem z adresem sip:fred@10.20.30.40
UA2 jest terminalem z adresem sip:chris@sipcenter.com



Rys. 5.4 Schemat wymiany komunikatów w sesji nawiązującej połączenie głosowe w protokole SIP[40]

UA1 wysyła wygenerowane żądanie INVITE

```
INVITE sip:chris@sipcenter.com SIP/2.0
CSeq: 1 INVITE
To: sip:chris@sipcenter.com
From: fred< sip:fred@10.20.30.40 >
Call-ID: 652302859179069176O@10.20.30.40
Via: SIP/2.0/UDP 10.20.30.40
Contact: < sip:fred@10.20.30.40 >
Subject: no subject
Content-Type: application/sdp
Content-Length: 123

v=0
o=- 982769551076 982769551076 IN IP4 10.20.30.40
s=-
c=IN IP4 10.20.30.40
t=0 0
m=audio 5004 RTP/AVP 8 3 0
```

Treść komunikatów zawiera dane w formacie protokołu SDP. Dokładne omówienie protokołu znajduje się w punkcie 5.11.
Serwer Proxy przyjmuje żądanie i jeśli jest poprawne wysyła odpowiedź do UA1

```
SIP/2.0 100 Trying
To: sip:chris@sipcenter.com
From: fred< sip:fred@10.20.30.40 >
CSeq: 1 INVITE
Call-ID: 6523028591790691760@10.20.30.40
Via: SIP/2.0/UDP 10.20.30.40
Content-Length: 0
```

Następnie zakładając, że serwer wie jak skontaktować się z UA2, niezwłocznie przesyła żądanie INVITE do UA2 dodając do jego nagłówka kolejne pole *Via*

```
INVITE sip:chris@10.20.30.60 SIP/2.0
CSeq: 1 INVITE
To: sip:chris@sipcenter.com
From: fred< sip:fred@10.20.30.40 >
Call-ID: 6523028591790691760@10.20.30.40
Via: SIP/2.0/UDP 10.20.30.50:5060;branch=C1C334F113C4000000E4D1911E77*4*0*0
Via: SIP/2.0/UDP 10.20.30.40
Contact: < sip:fred@10.20.30.40 >
Subject: no subject
Content-Type: application/sdp
Content-Length: 123

v=0
o=- 982769551076 982769551076 IN IP4 10.20.30.40
s=-
c=IN IP4 10.20.30.40
t=0 0
m=audio 5004 RTP/AVP 8 3 0
```

UA2, gdy otrzyma żądanie INVITE generuje na jego podstawie odpowiedź *180 Ringing,* przesyła go do serwera zgodnie kolejnością pól *Via* i informuje o nadejściu połączenia użytkownika UA1.

```
SIP/2.0 180 Ringing
To: sip:chris@sipcenter.com
From: fred< sip:fred@10.20.30.40 >
CSeq: 1 INVITE
Call-ID: 6523028591790691760@10.20.30.40
Via: SIP/2.0/UDP 10.20.30.50:5060;branch=C1C334F113C4000000E4D1911E77*4*0*0
Via: SIP/2.0/UDP 10.20.30.40
Content-Length: 0
```

Serwer odbiera odpowiedź UA2 i przekazuje je do UA1 w postaci jak przedstawiono poniżej

```
SIP/2.0 180 Ringing
To: sip:chris@sipcenter.com
From: fred< sip:fred@10.20.30.40 >
CSeq: 1 INVITE
Call-ID: 6523028591790691760@10.20.30.40
Via: SIP/2.0/UDP 10.20.30.40
Content-Length: 0
```

Dzięki temu, że UA1 otrzyma odpowiedź *180 Ringing* wie, że żądanie dotarło do UA2 i czeka na jego reakcję. Kiedy UA2 podejmie połączenie wysyła pakiet *200 OK* do serwera. Odpowiedź budowana jest na podstawie żądania INVITE i zawiera wszystkie jego pola Via, co pozwala na wysłanie odpowiedzi tą samą drogą, co przyszło żądanie.

```
SIP/2.0 200 OK
To: sip:chris@sipcenter.com;tag=-994822506
From: fred< sip:fred@10.20.30.40 >
CSeq: 1 INVITE
Call-ID: 652302859179069176O@10.20.30.40
Via: SIP/2.0/UDP 10.20.30.50:5060;branch=C1C334F113C4000000E4D1911E77*4*0*0
Via: SIP/2.0/UDP 10.20.30.40
Contact: < sip:chris@10.20.30.60 >
Content-Type: application/sdp
Content-Length: 101

v=0
o=- 0 0 IN IP4 10.20.30.60
s=-
c=IN IP4 10.20.30.60
t=0 0
m=audio 5004 RTP/AVP 8 3 0
```

Serwer przekazuje pakiet do UA1 odejmując pole nagłówka *Via*.

```
SIP/2.0 200 OK
To: sip:chris@sipcenter.com;tag=-994822506
From: fred< sip:fred@10.20.30.40 >
CSeq: 1 INVITE
Call-ID: 652302859179069176O@10.20.30.40
Via: SIP/2.0/UDP 10.20.30.40
Contact: < sip:chris@10.20.30.60 >
Content-Type: application/sdp
Content-Length: 101

v=0
o=- 0 0 IN IP4 10.20.30.60
s=-
c=IN IP4 10.20.30.60
t=0 0
m=audio 5004 RTP/AVP 8 3 0
```

UA1 wysyła pakiet potwierdzający i inicjuje połączenie głosowe

```
ACK sip:chris@10.20.30.60 SIP/2.0
CSeq: 1 ACK
To: sip:chris@sipcenter.com;tag=-994822506
From: fred< sip:fred@10.20.30.40 >
Call-ID: 652302859179069176O@10.20.30.40
Via: SIP/2.0/UDP 10.20.30.40
Content-Length: 0
```

Po takiej wymianie komunikatów oba UA mają do wykonania ogrom pracy związanych z przejęciem zasobów takich jak urządzenia dźwiękowe, inicjacją kodeków i otworzeniem sesji RTP[40].

5.11 Składnia protokołu SDP

Ważnym zagadnieniem dotyczącym połączeń głosowych w SIP jest protokół SDP. Używany jest on do negocjacji możliwości określonego UA dotyczących mediów. Sesje multimedialne, które SDP wpiera, mogą być typu punkt-punkt jak i również rozsiewcze[48].
W protokole zawarte mogą być informacje o:

- strumieniach danych - sesja multimedialna może składać się ze strumienia głosowego i strumienia wideo lub oddzielnych strumieni np. tylko głosowego,
- adresach docelowych,
- portach - numer portu służącego do wysyłania i otrzymywania strumieni danych,
- rodzaju przenoszonych danych,
- metodach kodowania danych multimedialnych,
- czasach rozpoczęcia i zakończenia sesji[48]

Protokół przekazuje te informacje w postaci tekstowej. Zawarte są one w treści (body) niektórych żądań i odpowiedzi SIP. Żądanie INVITE powinno zawierać dane SDP.
Poniżej przedstawiony jest przykładowy opis w protokole SDP

```
v = 0
o = g.bell 87728 8772 IN IP4 132.151.1.19
s = My session
u = http://www.ietf.org
e = g.bell@bell-telephone.com
c = IN IP4 132.151.1.19
b = CT:64
t = 3086272736 0
k = clear:manhole cover
m = audio 3456 RTP/AVP 96
a = rtpmap:96 VDVI/8000/1
m = video 3458 RTP/AVP 31
m = application 32416 udp wb
a = orient:portrait
```

Protokół charakteryzuje się jednoliterowymi oznaczeniami poszczególnych właściwości przedstawionymi w jednej linii.
Poniżej opisane zostały znaczenia niektórych linii stosowanych w SDP[45].

v (version) - linia określa wersję protokołu SDP,

o (originator) - linia mówi kto jest inicjatorem sesji, zawiera informacje o nazwie hosta, użytkownika czy też numerze sesji, jest ona postaci o = <username> <session id> <version> <network type> <address type> <address>,

s (session) - linia informująca o nazwie sesji,

u (url) - linia informacyjna mówiąca o URL'u użytkownika,

e (e-mail) - linia zawierająca e-mail użytkownika,

c (connection) - mówi o parametrach sesji jest ona postaci c = <network type> <address type> <connection address>,

b (bandwidth) - linia informuje o szerokości pasma, wartość wyrażana jest w kb/s i ma postać CT:64 co oznacza pasmo 64 kb/s

t (times) - linia oznaczająca czas rozpoczęcia i zakończenia sesji wyrażana w sekundach jest ona postaci t = <start time> <stop time> Czas wyrażany jest w postaci dziesiętnej zgodnie z protokołem NTP (Network Time Protocol). Wartość „0" oznacza nielimitowanie sesji.

k (encryption keys) - linia informuje o sposobie kodowania danych oraz o kluczu umożliwiającym rozkodowanie danych, jest ona postaci k = <method> <encryption key>,

m (media) - linia określająca strumienie danych, w przykładzie powyżej występują trzy linie ***m*** określające kolejno strumień : audio, wideo i danych
linia m jest postaci m = <media> <port> <transport> <fmt list>,
Parametry oznaczają:

- <media> - określa rodzaj przesyłanego strumienia. Może to być głos, obraz, dane, strumienie kontrolne itp.,
- <port> - numer portu docelowego
- <transport> - określa rodzaj protokołu transportowego użytego do przeniesienia strumieni, zazwyczaj stosowany jest tu zwrot RTP/AVP (Real Time Protocol/Audio/Video Profile), co oznacza, że protokołem użytym będzie RTP z profilem AVP, czyli za jego pomocą przenoszone będą dane Audio/Video. Inne profile mają być zdefiniowane w przyszłości[48].
- <fmt list> - lista numerów oznaczających typy danych w profilu (mogą to być np. oznaczenia kodeków). Pojawienie się numeru mówi, że określony typ danych profilu może być użyty w sesji Audio/Video. Dla każdego

numeru wymienionego w liście może występować dodatkowo parametr ***a*** określający cechy strumienia,

a (attribute) - linia określająca atrybuty dla strumieni np. parametry kodeka

5.12 Innowacje wprowadzane do standardu protokołu SIP

W tym podrozdziale przedstawione zostaną niektóre z modyfikacji pierwotnego standardu dla stosu protokołu SIP. Niektóre z nich znajdują się w tzw. draft'ach[41], czyli dokumentach wydawanych specjalistów oraz przez IETF zawierających szkice proponowanych zmian czy propozycje nowych technologii. Niektóre z tych dokumentów wnoszą dodatkowe informacje dla standardów internetowych a inne mogą definiować nawet urządzenia i ich cechy, które muszą być spełnione dla poprawnego działania standardu.
Wiele innowacji zostaje zaakceptowanych i zostają one formułowane w RFC (Request For Comment) i publikowane jako oficjalne specyfikacje standardów organizacji IETF.
Wiele innowacji tutaj przedstawionych ma już swoje w RFC.

5.12.1 Uaktualnienia standardu SIP zawarte w RFC

Dużo rozszerzeń standardu pojawiło się w obszarze żądań i odpowiedzi. Żądania zaakceptowane przez RFC 3261[35] (drugie RFC dotyczące całego standardu) są to:

- ACK
- BYE
- CANCEL
- INVITE
- OPTIONS
- REGISTER

Implementacja aplikacji przeznaczonych do komunikacji za pomocą SIP wymogła następujące dodatkowe żądania:

- INFO – używane do przenoszenia informacji kontrolnych lub dodatkowych dla sesji. Najczęściej używane do przenoszenia obrazków związanych z sesją. Funkcjonalność żądania została opisana w RFC 2976 [44].
- PRACK – podobne do ACK, stosowane jednak jako tymczasowa odpowiedź. Używane w celu uzyskania większej niezawodności podczas dostarczania pakietów.

Dzięki PRACK można częściej wysyłać potwierdzenia i otrzymywać potwierdzenia. Chociaż nie są to potwierdzenia ostateczne, to mogą one być wykorzystywane np. do wizualizacji postępu nawiązywania połączenia. Funkcjonalność żądania została opisana w RFC 3262 [51].

- UPDATE – pozwala klientowi zaktualizować parametry sesji bez wpływu na stan rozmowy. Funkcjonalność żądania została opisana w RFC 3311 [54].
- SUBSCRIBE – używane w rozbudowanych strukturach standardu SIP np. serwer Registar i kontrolowani przez niego użytkownicy, pozwala, aby węzły struktury mogły informować innych członków struktury o zmianach i innych ważnych zdarzeniach dotyczących struktury lub jej członków. Dzięki temu może być rozwiązywany w prosty sposób problem dostępności użytkownika. Żądanie definiowane wraz z NOTIFY (opisane poniżej) Funkcjonalność żądania została opisana w RFC 3265 [52].
- NOTIFY – używane do powiadamiania użytkowników wpisanych za pomocą SUBSCRIBE na listę węzła struktury o zmianach stanu użytkownika wysyłającego NOTIFY. Mechanizm wykorzystywany do realizacji stanu obecności użytkownika. W takim przypadku komunikaty NOTIFY zawierają informację ma temat obecności użytkownika, czyli np: dostępny, zajęty, zaraz wracam itp. Funkcjonalność żądania została opisana w RFC 3265 [52].
- MESSAGE – żądanie wprowadzone w celu dodania możliwości wysyłania wiadomości tekstowych. Funkcjonalność żądania została opisana w RFC 3428 [55].
- REFER – żądanie, które pozwala używać zasobów wysyłającego żądanie przez odbierającego żądanie. Przykładem może być użyczenie kontaktu (adresu SIP) z własnej listy kontaktów innej osobie. Funkcjonalność żądania została opisana w RFC 3515 [57].

Wraz z żądaniami pojawiły się oczywiście nowe odpowiedzi. Były one rezultatem wprowadzenia nowych żądań. Tabela 5.2 zawiera listę najpopularniejszych odpowiedzi. Odpowiedzi dodane później niż w RFC 3261 [35] zostały wyróżnione[72].

Klasa	Odpowiedź i kod odpowiedzi
Odpowiedzi pomyślne (2xx)	• ***ACCEPTED - 202 (opisane w RFC3265)***
Błędy żądań (4xx)	• ***BAD EVENT - 489 (opisane w RFC3265)*** • ***REQUEST PENDING - 491 (opisane w RFC3265)*** • ***UNDECIPHERABLE - 493 (opisane w RFC3265)***

Tabela 5.2 Najnowsze odpowiedzi w protokole SIP. [35]

Jednym z ważniejszych RFC wydanym niedługo po uaktualnieniu w RFC 3261[35] jest RFC 3263[50] dotyczące lokalizacji serwerów SIP za pomocą protokołu DNS. SIP używa procedur protokołu DNS, aby klient (UA) mógł rozwiązywać SIP URI, identyfikujący użytkownika w sieci, w adres IP, numer portu oraz protokół transportowy.
W dokumencie został przedstawiony następujący scenariusz. Użytkownik (UA1) w domenie A chce się skomunikować z użytkownikiem (UA2) w domenie B. Żeby tego dokonać, UA1 wysyła odpowiedni komunikat do serwera Proxy1 w swojej domenie (domena A). Proxy1 przekazuje komunikat do serwera proxy w domenie użytkownika drugiego (domena B) czyli do Proxy2. Proxy2 przekazuje komunikat do użytkownika UA2.
Podczas tej krótkiej wymiany komunikatów należy zauważyć jeden problem. Proxy1 musi ustalić adres serwera SIP, czyli Proxy2 w domenie B. Żeby tego dokonać, Proxy1 korzysta z procedur protokołu DNS używając rekordów SRV oraz NAPTR[19]. Dokument RFC 3263 pomaga zauważyć związane z tą sytuacją problemy i opisuje rozwiązania[50].

W grudniu 2002 roku wydane zostało RFC dotyczące zaakceptowania nowego pola w nagłówku SIP o nazwie *Reason*[50]. Stwierdzono, że pole *Reason* może być pomocne w uzyskaniu informacji, dlaczego otrzymaliśmy określoną odpowiedź od serwera lub innego UA. Takie same odpowiedzi SIP mogą się pojawić z różnych powodów. Przykładem może być żądanie SIP CANCEL, może być wynikiem zakończenia rozmowy lub też zerwania rozmowy przed jej odebraniem. Pole *Reason* jest również używane do przechowywania końcowych kodów statusu w komunikatach tymczasowych, co rozwiązuje problemy różnorodne problemy pojawiające się podczas odpowiedzi SIP oznaczane skrótem HERFP (Heterogeneous Error Response Forking Problem).
Dokument RFC 3326 opisuje jak stosować to pole i do jakich celów używane jest z powodzeniem.[53]

Ciekawą propozycję przyjęto w roku 2003 na temat kompresji komunikatów SIP. RFC 3486 [56] opisuje mechanizmy informowania urządzeń SIP, że pożądana jest kompresja komunikatu lub komunikatów. Mechanizmy te realizowane są przy pomocy dodatkowych cech wpisanych w pole *Via* nagłówka oraz dołączane do SIP URI (przedstawiono poniżej).

- `Via: SIP/2.0/UDP server1.foo.com:5060;branch=z9hG4bK87a7;comp=sigcomp`
- `sip:alice@atlanta.com;comp=sigcomp`

Oznaczenie *comp=sigcomp* w polu *Via* determinuje zastosowanie kompresji w odpowiedzi na żądanie, natomiast *comp=sigcomp* w SIP URI oznacza, że żądanie powinno być skompresowane używając kompresji SigComp dokładnie opisanej w innym RFC - "Signaling Compression (SigComp)", RFC 3320 [58].

Wartym wspomnienia jest także RFC 3911 [59] przyjęte w październiku 2004 roku. Definiuje ono użycie nowego pola nagłówka o nazwie *Join*, służącego do przyłączenia użytkownika do istniejącego już połączenia.
Jawne użycie pola nagłówka *Join* jest stosowne w następujących sytuacjach:

- konferencja jeszcze nie istnieje – żądanie INVITE z polem *Join* zostanie wtedy odebrane jako zwykłe zaproszenie do rozmowy,
- dołączający się do rozmowy nie wie, że rozmowa do której chce się dołączyć, jest już konferencją,
- dołączający się do rozmowy nie zna URI konferencji.

Join może występować jedynie w żądaniach INVITE. Pole nagłówka *Join* posiada atrybuty niezbędne do dołączenia się do istniejącej rozmowy.
Są to:

- call-id,
- to-tag,
- from-tag

Przykład pola może wyglądać następująco:

```
Join: 12adf2f34456gs5;to-tag=12345;from-tag=54321
Join: 87134@192.0.2.23;to-tag=24796;from-tag=0
```

Pierwszy argument to call-id, następnie po średniku to-tag oraz from-tag [59]

Przykładowa sesja pomiędzy trzema UA, gdzie stosowany jest nagłówek z polem *Join.* W sesji biorą udział trzy UA: A, B i C. Klienty B i C nawiązują normalne połączenie ustalając specjalne tag'i (oznaczenia) dla pól *From* oraz *To* Tagi te wykorzystywane są następnie przez Klienta A w zbudowaniu poprawnego pola *Join.*

Wiadomość 1 przesłana od C do B

```
INVITE sip:bob@example.org SIP/2.0
To: <bob@example.org>
From: <carol@example.org>;tag=xyz
Call-Id: 7@c.example.org
CSeq 1 INVITE
Contact: <sip:carol@c.example.org>
```

Wiadomość 2 przesłana od B do C

```
SIP/2.0 200 OK
To: <bob@example.org>;tag=pdq
From: <carol@example.org>;tag=xyz
Call-Id: 7@c.example.org
CSeq 1 INVITE
Contact: <sip:bob@b.example.org>
```

Wiadomość 3 przesłana od C do B

```
ACK sip:carol@c.example.org SIP/2.0
To: <bob@example.org>;tag=pdq
From: <carol@example.org>;tag=xyz
Call-Id: 7@c.example.org
CSeq 1 INVITE
```

Wiadomość 4 przesłana od A do B zawierająca pole nagłówka *Join*

```
INVITE sip:bob@b.example.org SIP/2.0
To: <sip:bob@example.org>
From: <sip:alice@example.org>;tag=iii
Call-Id: 777@a.example.org
CSeq: 1 INVITE
Contact: <sip:alice@a.example.org>
Join: 7@c.example.org;to-tag=xyz;from-tag=pdq
```

W wiadomościach nie zostały przedstawione wszystkie pola, aby nie zaciemniać meritum.

Ostatnie (stan na maj 2005) wydane RFC związane ze stosem protokołów SIP dotyczy okresowego sprawdzania ważności sesji [60]. RFC 4028 zatytułowane „Session Timers in the Session Initiation Protocol" definiuje rozszerzenie dla SIP, które pozwala na okresowe odświeżanie sesji poprzez ponowne wysyłanie żądań INVITE lub wysyłanie żądań UPDATE.

Uaktualnienia takie pozwalają urządzeniom UA lub serwerom Proxy stwierdzić czy sesja SIP jest wciąż aktualna i aktywna.
Rozszerzenie SIP opisane w tym dokumencie definiuje dwa dodatkowe pola mogące znaleźć się w nagłówku żądań, które determinować będą cel wysyłanego żądania.
Nagłówki definiowane w rozszerzeniu to:

- Session-Expires
- Min-SE

Pierwszy z nich *Session-Expires* zawiera planowany czas sesji. Drugi, *Min-SE* definiuje minimalną dozwoloną wartość czasu sprawdzania aktywności sesji.[60]
Przykład stosowania pól nagłówków:

```
Session-Expires: 3600
Min-SE: 3600
```

Wartości obu pól wyrażone są w sekundach.

Tabela 5.3 zawiera informacje w jakich żądaniach i odpowiedziach można stosować wyżej omówione pola.

Pole nagłówka	gdzie	ACK	BYE	CAN	INV	OPT	REG	PRA	UPD	SUB	NOT
Session-Expires	Request	-	-	-	opcja	-	-	-	opcja	-	-
Session-Expires	2xx	-	-	-	opcja	-	-	-	opcja	-	-
Min-SE	Request	-	-	-	opcja	-	-	-	opcja	-	-
Min-SE	422	-	-	konieczne	-	-	-	konieczne	-	-	-

Tabela 5.3 Dozwolone rozmieszczenie dla pól nagłówka *Session-Expires* oraz *Min-SE*[60]

5.12.2 Uaktualnienia przedstawione w szkicach (drafts)

W styczniu 2005 zaproponowano wprowadzenie obsługi nowego protokołu transportowego SCTP (Stream Control Transmission Protocol) przez standard SIP [61]. Propozycje umieszczono w dokumencie zatytułowanym „The Stream Control Transmission Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP)”[61].
Dokument specyfikuje mechanizmy użycia protokołu SCTP jako transportu pomiędzy obiektami SIP. SCTP jest nowym (październik 2000 RFC 2960 [62]) protokołem transportowym, który posiada cech potrzebnech do efektywnego przesyłania danych w czasie rzeczywistym. Protokół SIP z założenia jest właściwie niezależny od protokołów warstwy

transportowej, więc włączenie SCTP do szeregu obsługiwanych protokołów jest procesem zbliżonym do włączenia TCP. Polega to głównie na dostosowaniu procedur, gdyż były one definiowane dla protokołów UDP i TCP. Dokument draft-ietf-sip-sctp-06.txt definiuje właśnie przebieg procedur dla SCTP w SIP.

Kolejnym dokumentem wartym uwagi jest draft-ietf-sip-identity-05.txt [63] zawierający propozycje rozszerzeń dotyczących poprawienia uwierzytelniania użytkowników (UA) w SIP. Dokument zatytułowany jest „Enhancements for Authenticated Identity Management in the Session Initiation Protocol" i stwierdza w swojej treści, że stosowane metody autentykacji użytkowników nie są adekwatne do zabezpieczeń kryptograficznych możliwych do stosowania dla komunikatów SIP. Przedstawione są w nim mechanizmy do bezpiecznej identyfikacji twórców komunikatów. Realizowane jest to poprzez wprowadzenie dwóch nowych pól do nagłówka. Są nimi:

- Identity
- Identity-Info

Identity służy do przekazywania podpisu używanego do weryfikacji tożsamości, natomiast *Identity-Info* zawiera odniesienie do certyfikatu osoby podpisującej[63].

Przykładowe żądanie INVITE zawierające w nagłówku wyżej omawiane pola może wyglądać następująco:

```
INVITE sip:bob@biloxi.exmple.org SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.example.org>
From: Alice <sip:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.atlanta.example.com>
Identity:"NJguAbpmYXjnlxFmlOkumMI+MZXjB2iV/NW5xsFQqzD/p4yiovrJBqhd3TZke
gnsmoHryzk9gTBH7Gj/erixEFIf82o3Anmb+CIbrgdl03gGaD6ICvkpVqoMXZZjdvSpycyH
Ohh1cmUx3b9Vr3pZuEh+cB01pbMQ8B1ch++iMjw="
Identity-Info: <https://atlanta.example.com/cert>;alg=rsa-sha1
Content-Type: application/sdp
Content-Length: 147
```

Treść wiadomości zawierające informacje SDP zostało pominięte.

Kolejną propozycją jest “An Extension to the Session Initiation Protocol for Request History Information” draft mówiący o konieczności zachowywania historii żądań i odpowiedzi otrzymywanych w czasie komunikacji [64]. Dokument definiuje użycie nowego opcjonalnego pola nagłówka *History-Info*.

Te i wiele innych propozycji przedstawianych jest każdego miesiąca i rozpatrywanych. Niektóre z nich z powodzeniem są stosowane a następnie dołączane do standardu.

Pełną listę uaktualnień i innowacji dla standardu SIP można znaleźć na stronie organizacji IETF w rozdziale poświęconym protokołowi SIP[65].

5.12.3 Rozwiązania wspierające technologię SIP

Rozwiązania omówione w tym punkcie dotyczą problemów komunikacji przez Firewall i NAT[70] .
STUN jest mechanizmem zapewniającym możliwość komunikacji klientów SIP poprzez Firewall/NAT. W architekturze STUN znajduje się serwer odpowiedzialny za dostarczanie klientom takim jak terminale SIP publicznego adresu akceptowanego przez NAT. Większość obecnych terminali SIP umie obsłużyć mechanizm STUN[70].
STUN ma pewne ograniczenia np. nie obsługuje często używanej odmiany NAT - Symmetric NAT.
Mechanizm, który rozwiązuje już ten problem to TURN[70]. Pozwala on klientom za NAT odbierać pakiety UDP zawierające multimedia. Potrzebuje on jednak potężnej maszynyna której będzie chodził serwer TURN, aby nie wprowadzał opóźnień. Obecnie niewiele terminali SIP obsługuje ten standard.

5.13 Bezpieczeństwo w standardzie SIP

Zagwarantowanie bezpiecznych połączeń dla Telefonii IP jest sprawą złożoną. Nie ogranicza się ono jedynie do zapewnienia bezpiecznego transportu strumieni danych zawierających głos, ale bierze pod uwagę także bezpieczeństwo przesyłania komunikatów sygnalizacyjnych[86].
SIP zapewnia następujące kryteria zabezpieczeń komunikatów sygnalizacyjnych:

Poufność – zabezpieczająca wiadomości sygnalizacyjne, wymieniane pomiędzy komunikującymi się jednostkami, przed ich nieuprawnionym uzyskaniem przez strony do tego nieupoważnione[86],

Poufność w SIP osiąga się poprzez zastosowanie mechanizmu szyfrowania. Szyfrowaniu podlegają:

- ✓ cała treść komunikatu,
- ✓ istotne z punktu widzenia bezpieczeństwa pola nagłówka[86].

Uwierzytelnianie (zawierające integralność) – zapewnia kontrolę tożsamości stron i wiadomości sygnalizacyjnych wymienianych pomiędzy nimi[86].

W pierwszej specyfikacji (RFC 2543[42]) definiowane były dwa mechanizmy uwierzytelniania **Basic** oraz **Digest**.
Basic jest to bardzo prymitywny mechanizm nie gwarantujący wystarczającego poziomu bezpieczeństwa, gdyż dane uwierzytelniające użytkownika są przesyłane jako tekst jawny. Czyni to ten mechanizm całkowicie nieodpornym na atak typu powtórka[86].
Mechanizm Digest jest lepiej zabezpieczony od poprzednika (nie ma defektów mechanizmu Basic). Działa w oparciu o metodę wyzwanie/odpowiedź oraz korzysta z funkcji skrótu MD5 (Message Digest 5).
Specyfikacja SIP w RFC 3261[35] rezygnuje już z mechanizmu Basic. Obliguje do korzystania z mechanizmu Digest i wspiera uwierzytelnianie realizowane poprzez mechanizm S/MIME (Secure / Multipurpose Internet Mail Extension), który wykorzystuje do tego celu tunelowanie oraz podpis cyfrowy[86].

Największym mankamentem w bezpieczeństwie komunikacji sygnalizacyjnej SIP jest zbyt duża opcjonalność w stosowaniu mechanizmów bezpieczeństwa, co powoduje rzadkie ich używanie[86].

Rozdział 6. Praktyczne zastosowanie standardu SIP

6.1 Wstęp

W tym rozdziale postaram się przybliżyć aspekty implementacji części standardu SIP a mianowicie, omówić budowę terminala technologii sygnalizacyjnej SIP zwanego potocznie User Agent SIP (UA). W pracy został wykorzystany projekt typu Open Source, czyli projekt z dostępnym kodem źródłowym i posiadający licencję pozwalającą na jego modyfikacje. Można znaleźć go w zasobach Internetu pod nazwą *A JAIN-SIP Applet Phone For the People*[68]. Projekt był prowadzony pod patronatem National Institute of Standards and Technology (NIST), prace nad nim zostały przerwane na rzecz rozwoju większego *SIP-Communicator*[84] projektu w 2003 roku[77]. Aplikacja jest częścią implementowanej przez NIST architektury SIP. Powstały również projekty dotyczące implementacji bibliotek oraz dotyczące implementacji serwera Proxy SIP. Jak wynika z dokumentacji[77] projekty te razem współpracowały z powodzeniem. Wszystkie z nich były konstruowane zgodnie ze specyfikacją SIP. Moim celem było uaktualnić lub zmodyfikować aplikacje kliencką UA, aby współpracowała z obecnie stosowanym w świecie telefonii IP, darmowym serwerem Proxy SIP Express Router v 0.9 organizacji iptel.org[1]. Rozdział zawiera analizę projektu, opis wprowadzonych przeze mnie modyfikacji i osiągnięte efekty.

6.2 Technologia i język programowania oraz biblioteki VoIP

Chcąc stworzyć dziś aplikację z dziedziny VoIP natkniemy się na wiele rozwiązań dostępnych w Internecie. W obecnej chwili standardy VoIP dynamicznie się rozwijają. Potwierdzeniem tego może być duże zainteresowanie prasy oraz komercjalizowanie się już niemal doskonałych rozwiązań. Chcąc zbudować szybko aplikację VoIP pozbawioną błędów na poziomie implementacji standardów komunikacyjnych należy skorzystać z bibliotek. Opracowywane przez zespoły ludzi i testowane w różnych warunkach, dają pewne podstawy do stworzenia aplikacji. Moim zamierzeniem było znalezienie biblioteki implementującej standard sygnalizacyjny SIP oraz implementującej protokoły transmisyjne oraz kodeki. Szukając dogodnego rozwiązania zwróciłem szczególną uwagę na dwa rozwiązania.

6.2.1 eyeBeam SDK

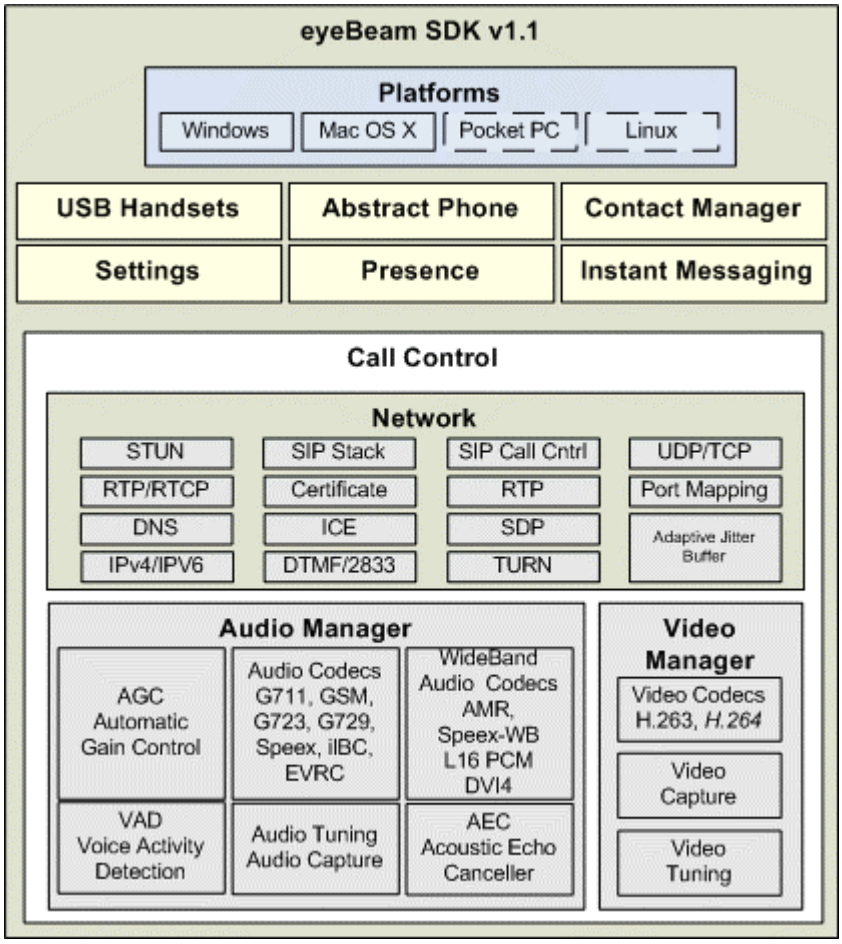
Pierwszym z nich jest „eyeBeam SDK" dostępna odpłatnie poprzez portal www.xten.com[69].

Produkt Xten jest zespoleniem interfejsów dla standardu SIP, zawartych w jednym SDK (Software Development Kit) umożliwiającym szybkie budowanie aplikacji z dziedziny VoIP[69]. Producent zapewnia, że jego produkt jest doskonałym wyborem dla dostawców usług internetowych szukających rozwiązań i chcących wdrożyć rozwiązania VoIP.

SDK zostało zbudowane w języku C i C++, bez użycia biblioteki MFC(Microsoft Foundation Class).

eyeBeam SDK jest to zestaw 28 modułów API (Application Programming Interfaces) pozwalających w łatwy sposób budować aplikację korzystając z dostępnych metod.

Rysunek 4.1 przedstawia najważniejsze aspekty zawarte w wersji 1.1 biblioteki eyeBeam SDK



Rys. 4.1 Struktura budowy biblioteki eyeBeam SDK[69]

Oferta firmy Xten jest bardzo rozbudowana i zapewnia wiele możliwości. Oprócz podstawowych możliwości sieciowych takich jak obsługa gniazd (socket), obługa stosu SIP, protokołów RTP, SDP czy też obsługa protokołu IP w wersjach v4 i v6, eyeBaem zawiera implementację standardów SIP a nawet API do budowania interfejsu użytkownika.
Ważniejsze możliwości biblioteki eyeBeam wymienione są poniżej:

- STUN (Simple Traversal of UDP through NAT) – interfejs pomocny w obsłudze serwera STUN, który zapewnia klientowi SIP działanie za bramą NAT. Na żądanie serwer STUN dostarcza klientowi publiczny adres IP oraz port wykorzystywany do komunikacji [70],
- Proxy API – interfejs pomocny w budowie serwera Proxy dla SIP,
- Codec API – interfejs zawierający implementacje kodeków Audio i Video,
- Acoustic Echo Cancellation API – interfejs zawierający algorytmy redukujące efekt echa podczas rozmowy,
- User Interface API – interfejs pozwalający na łatwe tworzenie powłoki zewnętrznej dla użytkownika aplikacji,
- Phone Book API – interfejs implementujący książkę telefoniczną,
- Crypto API – interfejs zawierający implementacje algorytmów kryptograficznych,
- Instant Messenger API – interfejs obsługujący przesyłanie wiadomości tekstowych,
- Presence API – interfejs zawierający mechanizmy do sprawdzania obecności w sieci użytkownika.

Niestety biblioteka eyeBeam jest produktem komercyjnym, co przeszkadza w dokładnym przyjrzeniu się strukturze i zapoznaniu się z zastosowanymi rozwiązaniami.

6.2.2 JAIN SIP API

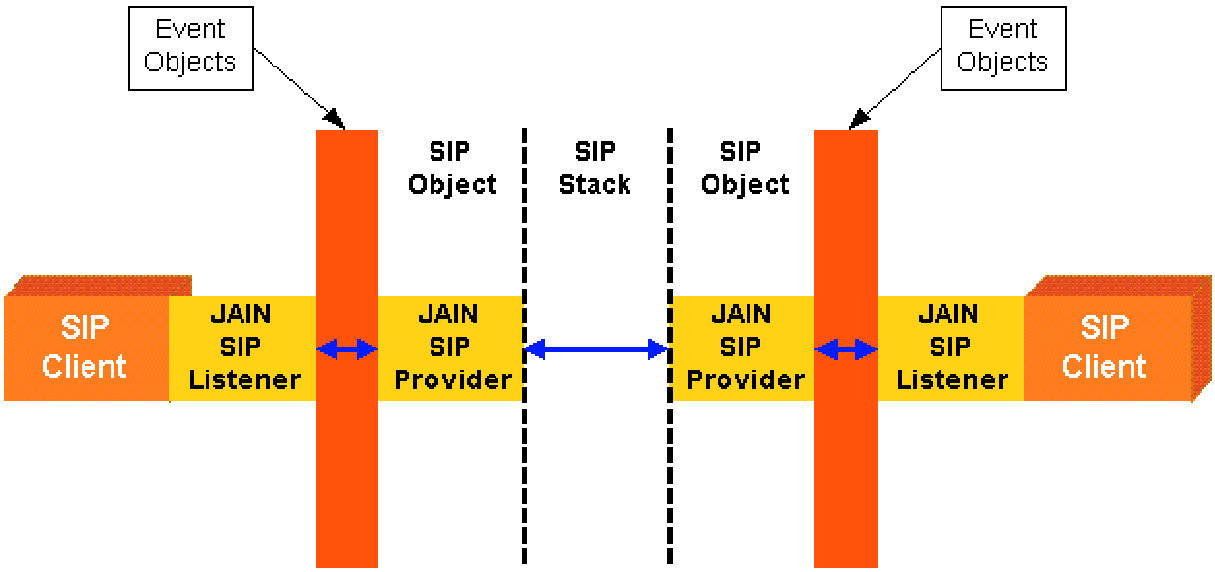
Drugim rozwiązaniem, któremu warto się przyjrzeć jest biblioteka JAIN SIP API (JAIN to skrót od Java API's for the Integrated Networks). Jest to rozwiązanie darmowe z dostępnym kodem źródłowym. Biblioteka została stworzona, aby wspierać rozwój aplikacji w standardzie SIP w języku Java.
JAIN SIP API składa się z następujących części:

- definiuje podstawowy interfejs potrzebny do tworzenia prawidłowo zbudowanych implementacji do wymiany komunikatów SIP,

- zawiera RI (Reference Implementation) czyli przykładową implementację funkcji zawartych w API, która dowodzi poprawności dostępnych w API funkcji. RI także służy demonstracji aplikacji Java napisanych przy pomocy JAIN SIPAPI,
- zawiera TCK (Technology Compatibility Kit) czyli testy poprawności aplikacji budowanych w oparciu o interfejs JAIN SIP.

Aplikacja oparta o JAIN SIP może być utworzona jako tradycyjny program Java lub też jako JavaBean czyli komponent Javy stosujący odpowiednie dla tego standardu zasady składni takie jak na przykład określone nazewnictwo metod[71].

Architektura biblioteki JAIN zawiera dwa istotne obiekty JAIN SIP Provider oraz JAIN SIP Listener. Komunikacja pomiędzy klientami odbywa się jak przedstawiono na rysunku 6.2



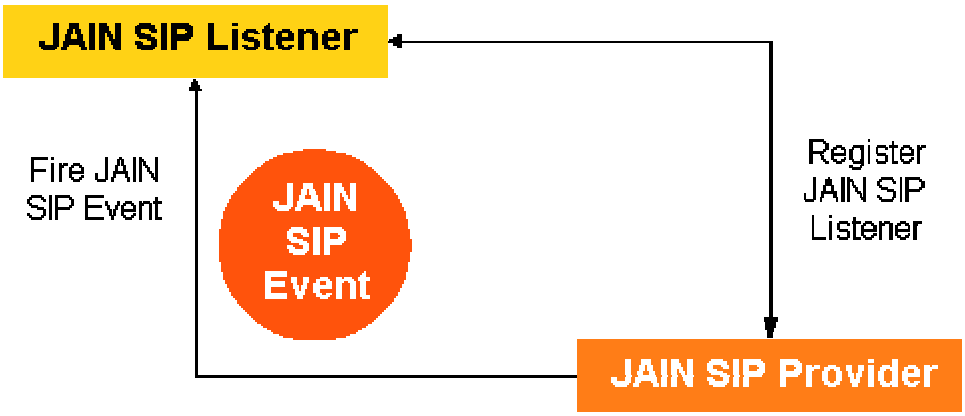
Rys. 6.2 Schemat architektury stosowanej w bibliotece JAIN SIP[71]

W schemacie komunikacji proponowanym przez JAIN SIP wyodrębnione są podstawowe elementy takie jak:

- JAIN SIP Event – jest częścią API, jest to poddany enkapsulacji komunikat SIP (Message objects), który jest przesyłany między JAIN SIP Provider'em a JAIN SIP Listener'em
- JAIN SIP Provider – definiuje obiekt, który zapewnia aplikacji komunikację z zewnętrznymi obiektami architektury SIP[71]

- JAIN SIP Listener – jest częścią API, definiuje obiekt, który używa procedur dostarczanych przez JAIN SIP Provider'a.

Na rysunku 6.3 przedstawiony jest schemat komunikacji obiektów Listener i Provider.



Rys.6.3 Ogólny schemat komunikacji obiektów Listener i Provider w bibliotece JAIN SIP[71]

Obiekt Provider rejestruje obiekt Listener i od tej pory obiekty mogą współpracować wymieniając komunikaty SIP zamknięte w obiektach Event Message. Architektura przewiduje możliwość zarejestrowania jednego obiektu Listener w wielu obiektach Provider. Obiekt Provider związany jest bezpośrednio ze stosem SIP a w konsekwencji z portem protokołu TCP/UDP, którym przesyłane są pakiety[71].
Obiekt Listener obsługuje trzy typy obiektów Event:

- RequestEvent – obiekt zawierający żądanie
- ResponseEvent – obiekt zawierający odpowiedź
- TimeoutEvent – obiekt zawierający zawiadomienie o przekroczeniu dozwolonego czasu na odpowiedź

Obiekty wyżej wymienionych typów są wymieniane z Obiektem Provider [71].
Biblioteka JAIN zbudowana jest w oparciu o pierwszą specyfikacje SIP przedstawioną w RFC 3261.

6.2.2.1 NIST SIP Parser and Stack

NIST SIP Parser and Stack (obecna wersja 1.2) jest to biblioteka oparta na JAIN SIP API, implementująca jej interfejsy oraz zawierająca dodatkowe pakiety[72]. Pakietami rozszerzającymi JAIN SIP API są [72]:

- Parser package – pakiet służący do analizy nagłówków SIP, adresów oraz URL w pakietach SIP,
- JAIN SDP API (JSR[1] 141) – pakiet zawierający implementację protokołu SDP, pakiet jest jeszcze w fazie rozwoju, w lipcu 2004 roku rozpoczęto pracę nad ostateczną wersją, ale nie została ona jeszcze ukończona,
- Stack Package – pakiet zawiera klasy do budowania stosu protokołu SIP,
- JAIN Header package implementation – implementacja interfejsów pakietu Header biblioteki JAIN SIP API,
- JAIN Message package implementation – implementacja interfejsów pakietu Message biblioteki JAIN SIP API,
- JAIN Address package implementation – implementacja interfejsów pakietu Address biblioteki JAIN SIP API.

Biblioteka jest oparta o specyfikacje SIP przedstawioną w RFC 3261.

6.2.2.2 JMF API

JMF (Java Media Framework) API jest biblioteką umożliwiającą dodanie do aplikacji funkcjonalności dźwięku oraz wideo[75]. Jest to rozszerzenie J2SE (Java 2 Platform, Standard Edition) do realizacji odtwarzania, przechwytywania, kodowania strumieni popularnych formatów mediów[75].
W maju 2003 roku wydana została jak dotychczas najnowsza wersja biblioteki 2.1.1e. W listopadzie 2004 roku została dodana obsługa znanego formatu audio MP3[75].
W projekcie do tej pracy wykorzystywana była biblioteka JMF v 2.1.1e

6.3 Projekt UA SIP

W pracy dyplomowej wykorzystany został projekt o nazwie JAIN SIP Applet phone[80]. Projekt jest implementacją User Agent'a SIP zbudowanego na podstawie JAIN SIP 1.1 API

[1] JSR – Java Specification Requests – propozycje specyfikacji technologii dla języka Java

oraz JMF. W rzeczywistości aplikacja korzysta z biblioteki NIST SIP API implementującej większość interfejsów JAIN SIP 1.1 API, lecz powinna ona współpracować także z innymi implementacjami stosu SIP.

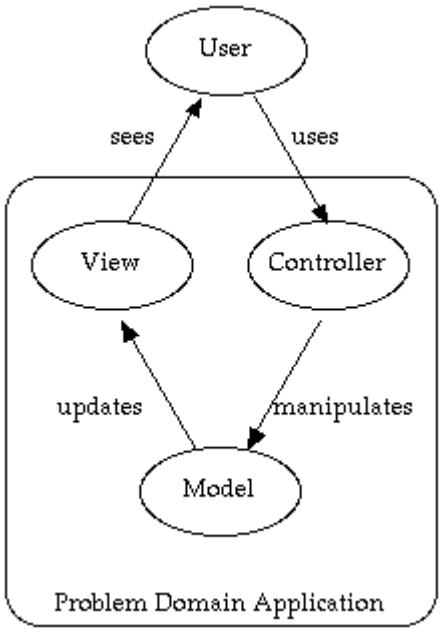
6.3.1 Konstrukcja projektu

Aplikacja jest napisana zgodnie ze znanym wzorcem projektowym MVC (Model-View-Controller)[73], co jest przypominane niemal w każdym pliku projektu JAIN SIP Applet phone. Jest to model trójwarstwowy, zapewniający klarowność kodu oraz łatwość dalszego rozwoju tworzonej aplikacji [73].

Warstwami w modelu są:

- View – warstwa tworząca widok dla użytkownika [73],
- Controller – warstwa odbierająca komunikaty i żądania od użytkownika i przekazująca je dalej do warstwy Model [73],
- Model – warstwa zawierająca logikę aplikacji, obsługująca żądania otrzymywane od warstwy Controller i zlecająca warstwie View wygenerowanie rezultatów [73].

Na rysunku 6.4 przedstawiony jest schemat logiczny modelu MVC.

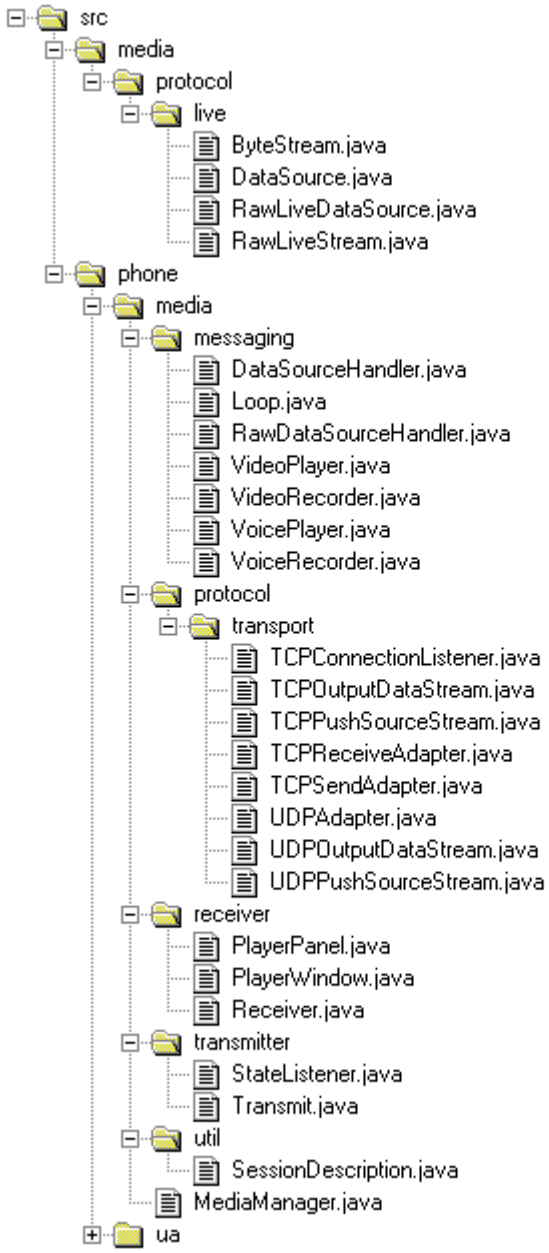


Rys. 6.4 Schemat logiczny aplikacji zbudowanej w oparciu o wzorzec projektowy MVC[73]

W projekcie JAIN SIP Applet phone głównymi klasami odpowiadającymi komponentom w schemacie są:
MessengerController – klasa reprezentująca Controller, odpowiada ona za odbieranie żądań użytkownika i przekazuje je do klasy Modelu, kontroluje stan aplikacji
MessengerManager– klasa implementująca komponent Model, umieszczona w niej jest logika aplikacji
NISTMessengerGUI – część odpowiedzialna za widok, interfejs użytkownika, implementuje komponent View

Wymienione wyżej klasy są klasami reprezentującymi komponenty w modelu MVC. Korzystają one z wielu klas pomocniczych, odpowiedzialnych już za konkretne zadania w projekcie.

Poniżej zamieszczone są rysunki przedstawiające wszystkie pliki projektu. Pierwszy z nich tj. rys. 6.5 zawiera pliki odpowiedzialne za transmisję mediów, natomiast drugi (rys. 6.6) przedstawia pliki budujące aplikację oraz zarządzające sygnalizacją protokołu SIP.

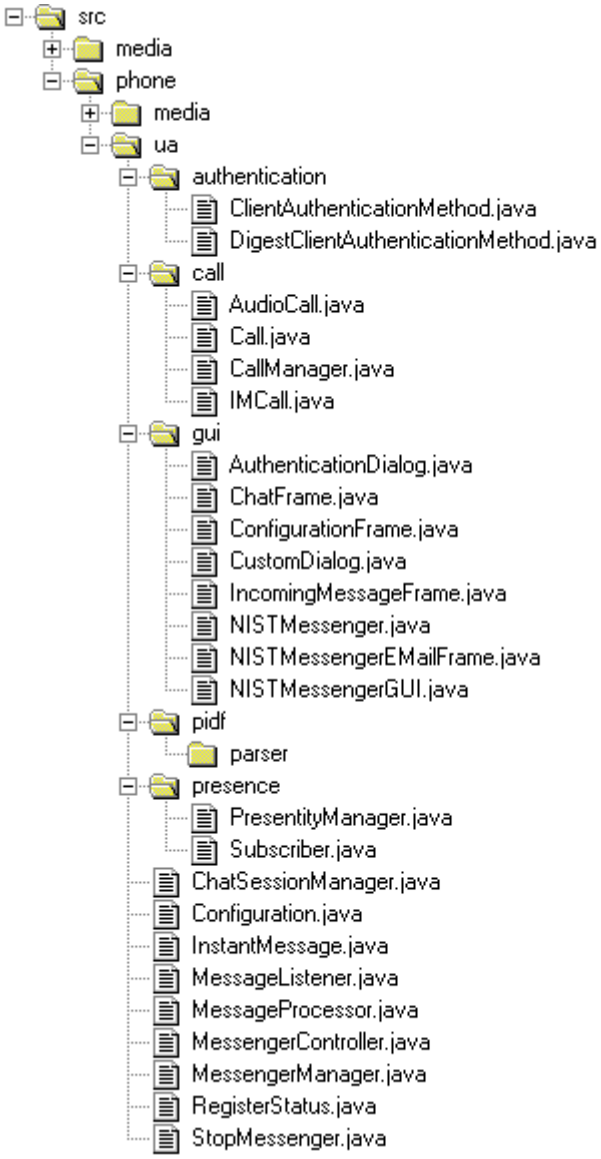


Rys. 6.5 Pliki projektu odpowiedzialne za obsługę protokołów transportowych oraz transmisję danych

Pliki z rys. 6.5 i zawarta w nich implementacja klas zawiera[74]:

- Obsługę strumienia danych MPEG_AUDIO lub GSM,
- Obsługę bufora dla danych MPEG_AUDIO lub GSM,
- Odtwarzanie danych audio,
- Rejestrowanie danych audio,
- Zarządzanie pakietami TCP oraz UDP
- Obsługa pakietów RTP

Pliki z rys. 6.6 zawierają implementację funkcjonalności programu Java oraz implementację sygnalizacji telefonicznej w standardzie SIP na najwyższym poziomie. Znaczy to, że zawarta implementacja dotyczy zachowań aplikacji w określonych sytuacjach ale korzysta z implementacji bibliotecznych dla poszczególnych części takich jak np. konstrukcja komunikatu.



Rys. 6.6 Pliki projektu związane bezpośrednio z aplikacją i zarządzaniem sygnalizacją SIP

Pakiet*ua* (rys. 6.6) zawiera pliki kontrolujące zachowanie całej aplikacji. Jest on także podzielony na kilka podpakietów. Podpakietami zawartymi w *ua* są:

1. Pakiet *authentication* – zawiera on dwa pliki implementujące autentykację czyli potwierdzenia tożsamości użytkownika, w skład pakietu wchodzą:
   - ClientAuthenticationMethod.java – plik zawiera definicję klasy abstrakcyjnej (interface), która zawiera deklaracje metod niezbędnych dla procesu autentykacji,
   - DigestClientAuthenticationMethod.java – plik implementuje interface ClientAuthenticationMethod.java, zawiera metody generujące hash (ciąg znaków alfanumerycznych, wynik funkcji mieszających) służący do autentykacji.

2. Pakiet *call*– zawiera cztery pliki implementujące obiekty zawierające informacje dla różnych rodzajów połączeń:
   - Call.java – klasa abstrakcyjna dla obiektów zawierających informacje o rozmowach,
   - AudioCall.java – implementuje klasę obiektów z informacjami dla rozmów głosowych,
   - IMCall.java – klasa z informacjami dla rozmów tekstowych,
   - CallManager.java – klasa definiująca metody obsługi obiektów rozmów.

3. Pakiet*gui* – zawiera pliki interfejsu użytkownika aplikacji, są to zarówno pliki dla apletu jak i pliki dla oddzielnej aplikacji Java.
   - AuthenticationDialog.java – okno dialogowe do autoryzacji użytkownika,
   - ChatFrame.java – wygląd okna do prowadzenia rozmów,
   - ConfigurationFrame.java – definiuje okno konfiguracji użytkownika,
   - CustomDialog.java – implementacja nieukończona,
   - IncomingMessageFrame.java – okno wyskakujące, gdy ktoś próbuje się połączyć z zalogowanym do tej aplikacji użytkownikiem,
   - NISTMessengerGUI.java – plik definiuje interface dla głównego okna interfejsu
   - NISTMessenger.java – główne okno interfejsu aplikacji,
   - NISTMessengerEMailFrame.java – plik odpowiadający za okno wysyłania wiadomości e-mail. Implementacja w projekcie nie jest ukończona.

4. Pakiet*pidf* – skrót od Presence Information Data Format, jest to pakiet obsługujący format PIDF. Opisany jest on w RFC 3863[76]. W pakiecie *pidf* znajduje się bezpośrednio pakiet *parser*, który przekształca dane w format PIDF i wyłuskuje dane

odbierane w tym formacie. Jest to format używany do danych stanowiących o obecności użytkownika w sieci, o jego obecnym statusie. Format opiera się o język XML[76]. Pliki zawarte w pakiecie *parser* implementują format, aby można było go użyć w aplikacji. Zawartość pakietu parser to:

- AddressTag.java – plik implementuje obsługę tagu AddressTag,
- AtomTag.java – obsługa tagu AtomTag,
- MSNSubStatusTag.java – obsługa tagu MSNSubStatusTag,
- PresenceTag.java – obsługa tagu PresenceTag,
- PresentityTag.java – obsługa tagu PresentityTag,
- StatusTag.java – obsługa tagu StatusTag,
- XMLpidfParser.java – implementuje parser dla języka XML.

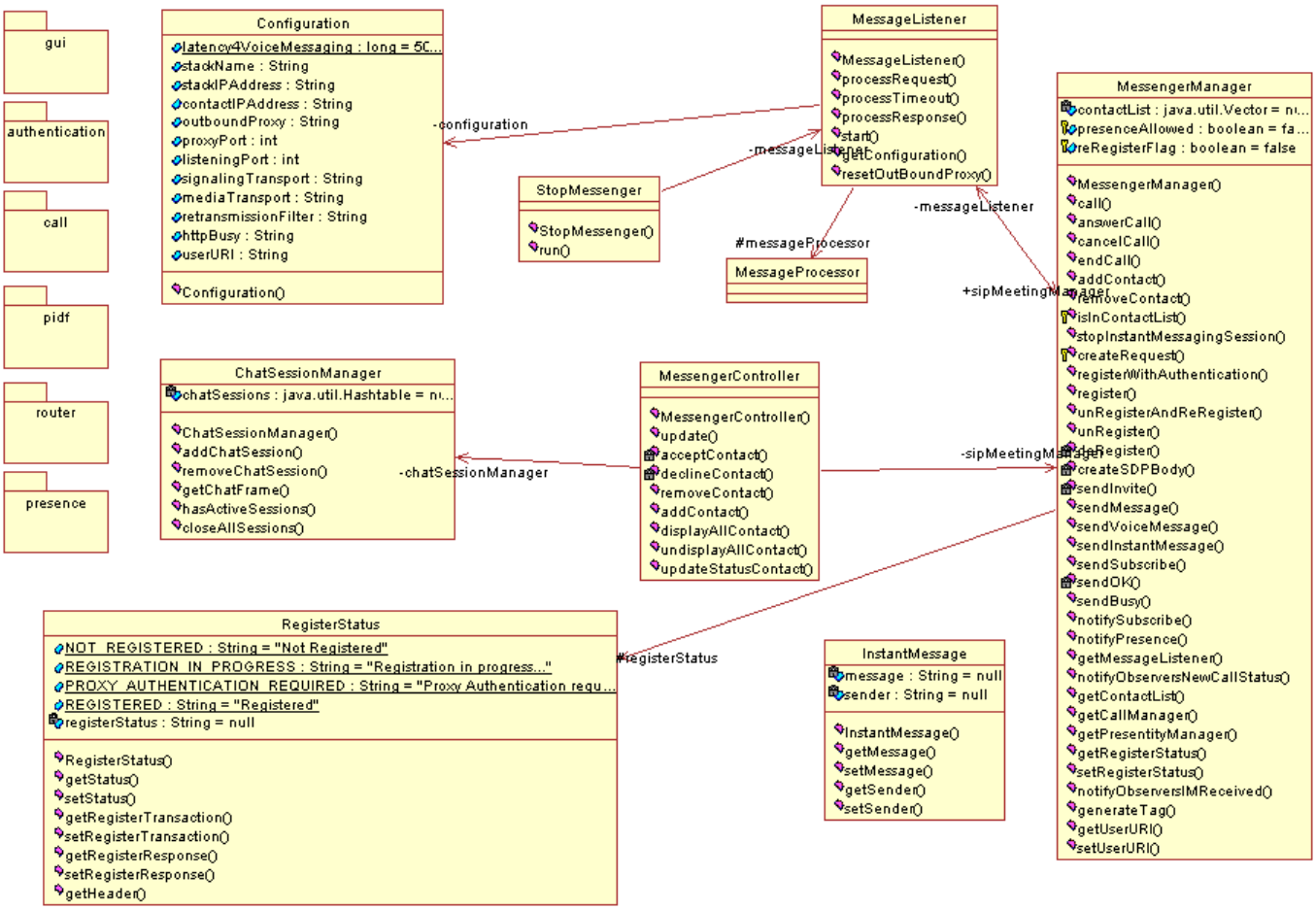
5. Pakiet*presence* – jest odpowiedzialny za obsługę aplikacji w zakresie statusu użytkownika. Używa on między innymi pakietu *pidf*. Zajmuje się dodawaniem użytkowników do listy subskrybentów, na co użytkownik musi się zgodzić oraz powiadamianiem wszystkich subskrybentów o zmianie swojego statusu.
   Pakiet obsługuje odpowiednie do tego celu komunikaty SIP, mianowicie :
   - SUBSCRIBE
   - NOTIFY

   Pliki pakietu to:
   - PresentityManager.java – zamieszczony w nim kod odpowiada za obsługę subskrybentów oraz własnego statusu, jak i za wysyłanie własnego żądania o subskrybcję u innego użytkownika.
   - Subscriber.java – implementuje subskrybenta wraz z jego właściwościami.
6. Pakiet*router* – pakiet jak sama nazwa wskazuje pełni rolę uproszczonego rutera aplikacji, czyli dba o przekazywanie pakietów SIP odpowiednią drogą[74]. Pakiet zawiera pliki:
   - MessengerHop.java – klasa zaimplementowana w pliku reprezentuje hop czyli parametry miejsca (routera) napotkanego na drodze pakietu[74],
   - MessengerRouter.java – w pliku zawarte są metody implementujące między innymi proste rutowanie tzn. Przekazywanie pakietu do serwera Proxy jaki jest zawarty w konfiguracji aplikacji[74].

W samym pakiecie *ua* istnieją pliki mające kluczowe znaczenie dla aplikacji. Są to:

- ChatSessionManager.java – klasa zarządzająca wiadomościami tekstowymi[74],
- Configuration.java – tworzy klasę zawierającą parametry aplikacji[74]
- InstantMessage.java – klasa reprezentująca wiadomość tekstową[74]
- MessageListener.java – implementuje zarządzanie przychodzącymi komunikatami SIP na wyższym poziomie abstrakcji. Logika pliku przedstawiona podczas omawiania architektury biblioteki JAIN SIP[71],
- MessageProcessor.java – implementuje zarządzanie przychodzącymi komunikatami SIP na niższym poziomie abstrakcji. Logika pliku przedstawiona podczas omawiania architektury biblioteki JAIN SIP[71]
- MessengerController.java – klasa implementuje kontrolera (obserwatora zmian) w całej aplikacji. W modelu MVC pełni rolę Contoller.
- MessengerManager.java – klasa implementująca obsługę rozmów oraz wszystkich zmian statusu, W modelu MVC reprezentuje część Model.
- RegisterStatus.java – klasa zajmująca się stanem rejestracji aplikacji,
- StopMessenger.java – klasa obsługująca kończenie działania aplikacji i oddawanie zasobów systemu

Przeprowadzona została analiza Reverse engineering (inżynieria wsteczna), która (po dokonaniu małych poprawek) dała rezultaty w postaci diagramu klas. Inżynieria wsteczna jest to proces badania produktu (urządzenia, programu komputerowego) w celu ustalenia jak dokładnie działa, a także jak i jakim kosztem został wykonany[81]. Prowadzany zazwyczaj w celu zdobycia informacji niezbędnych do skonstruowania odpowiednika[81].
Na rysunku 6.9 widoczny jest diagram klas przedstawiający centralną część projektu: pakiet *ua*.
Inżynieria wsteczna przeprowadzona została przy pomocy narzędzia Rational Rose Enterprise Edition[82].



Rys.6.9 Diagram klas pakietu ua aplikacji

6.3.2 Możliwości projektu JAIN SIP Applet phone

Projekt jest implementacją User Agent'a SIP na licencji open source. Posiada przede wszystkim możliwości prowadzenia rozmów audio oraz możliwości przesyłania wiadomości tekstowych(instant messaging)[77]. Aplikacja przewiduje dwa tryby komunikacji głosowej:

- Przesyłanie głosu w czasie rzeczywistym za pośrednictwem protokołu UDP wraz z RTP,
- Nagrywanie i przesyłanie wiadomości głosowych używając protokołu TCP

Interfejs użytkownika zaimplementowany jest jako aplet lub jako oddzielna aplikacja Java.

Do kompilacji projektu należy użyć narzędzia ANT[78]. Jest to narzędzie podobne do *make* w systemach unix'owych. Projekt zawiera plik build.xml (analogiczny do *makefile* w narzędziu *make*), którego modyfikacja umożliwia zdefiniowanie źródeł do kompilacji[77].

Domyślnie ustawiona jest kompilacja projektu do postaci apletu.

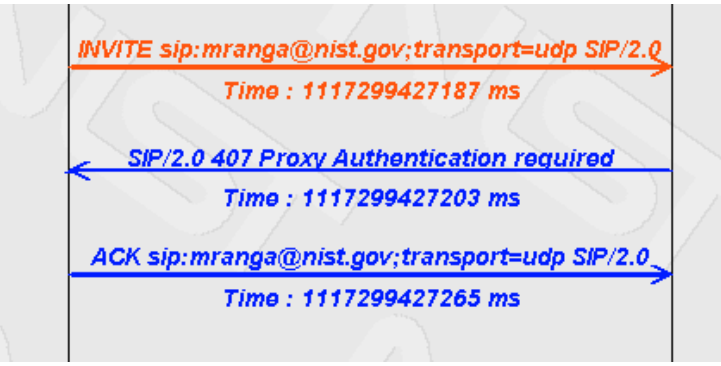
Rozwój aplikacji kierował się w stronę zaimplementowania w niej funkcji obsługi statusu, lecz nie zostało to ukończone.

6.4 Weryfikacja działania projektu

Projekt został uruchomiany i przetestowany we współpracy z dwoma serwerami Proxy architektury SIP. Pierwszym zalecanym rozwiązaniem była współpraca z projektem budowanym równolegle do *JAIN SIP Applet phone* o nazwie *JAIN SIP Presence proxy*[83]. Aplikacja nawiązywała połączenie głosowe bez większych problemów, gdy w konfiguracja serwera nie przewidywała autoryzacji oraz autentykacji. Problemy pojawiały się podczas próby współpracy obu aplikacji w trybie z autoryzacją.
Drugim rozwiązaniem był serwer proxy SIP Express Router (SER) promowany przez organizację www.iptel.org[1]. Dokumentacja projektu mówiła o dobrej współpracy tych aplikacji co okazało się nieprawdą. Dokumentacja nie podaje jednak wersji serwera SER, z którą współpracowała aplikacja. Okazało się, że obecna wersja serwera SER nie współpracuje dobrze z *JAIN SIP Applet phone*. Podobnie jak przy pierwszym serwerze błędy związane były z autentykacją.

6.5 Napotkane błędy projektu

Do wykrycia błędów w komunikacji aplikacji JAIN SIP Applet phone oraz serwera SER należało skompilować aplikację tak, aby widoczna była konsola (cmd). W konsoli obserwować było można informacje o przesyłanych pakietach (tryb debug). Podczas śledzenia współpracy aplikacji JAIN SIP Applet phone oraz serwera SER. Napotkałem błąd podczas próby wysłania pakietu INVITE. Zaistniałą sytuację przedstawia rysunek 6.7



Rys. 6.7 Błąd napotkany podczas próby rozpoczęcia rozmowy głosowej

Dokładne omówienie zawartości pakietów znajduje się w poniższym podrozdziale „Przeprowadzone testy”.
Po dokładnym przeanalizowaniu zaistniałej sytuacji, błąd ten skłonił mnie do bliższego zgłębienia możliwych przyczyn odpowiedzi 407 Proxy Authentication required. Informacje na ten temat znalazłem z RFC 3261 „SIP Session Initiation Protocol”[35]. Problemem okazał się brak uwierzytelniania. Aplikacja nie reagowała na odpowiedź 407 Proxy Authorization required otrzymaną po wysłaniu zaproszenia do rozmowy INVITE. Konieczne było zmodyfikowanie aplikacji w ten sposób, aby umiała się ona prawidłowo zachować w takiej sytuacji.

6.6 Modyfikacje wprowadzone do projektu

Zaimplementowano odpowiednie zachowanie aplikacji, które zapewnia uwierzytelnianie przesyłanych pakietów. Polega to na dodaniu do jego nagłówka (header) pola o nazwie Proxy-Authorization.

```
Proxy-Authorization: Digest
            username="Alice",
            realm="atlanta.com",
            nonce="c60f3082ee1212b402a21831ae",
            response="245f23415f11432b3434341c022"
```

Listing 6.1 Wygląd pola nagłówka Proxy-Authorization

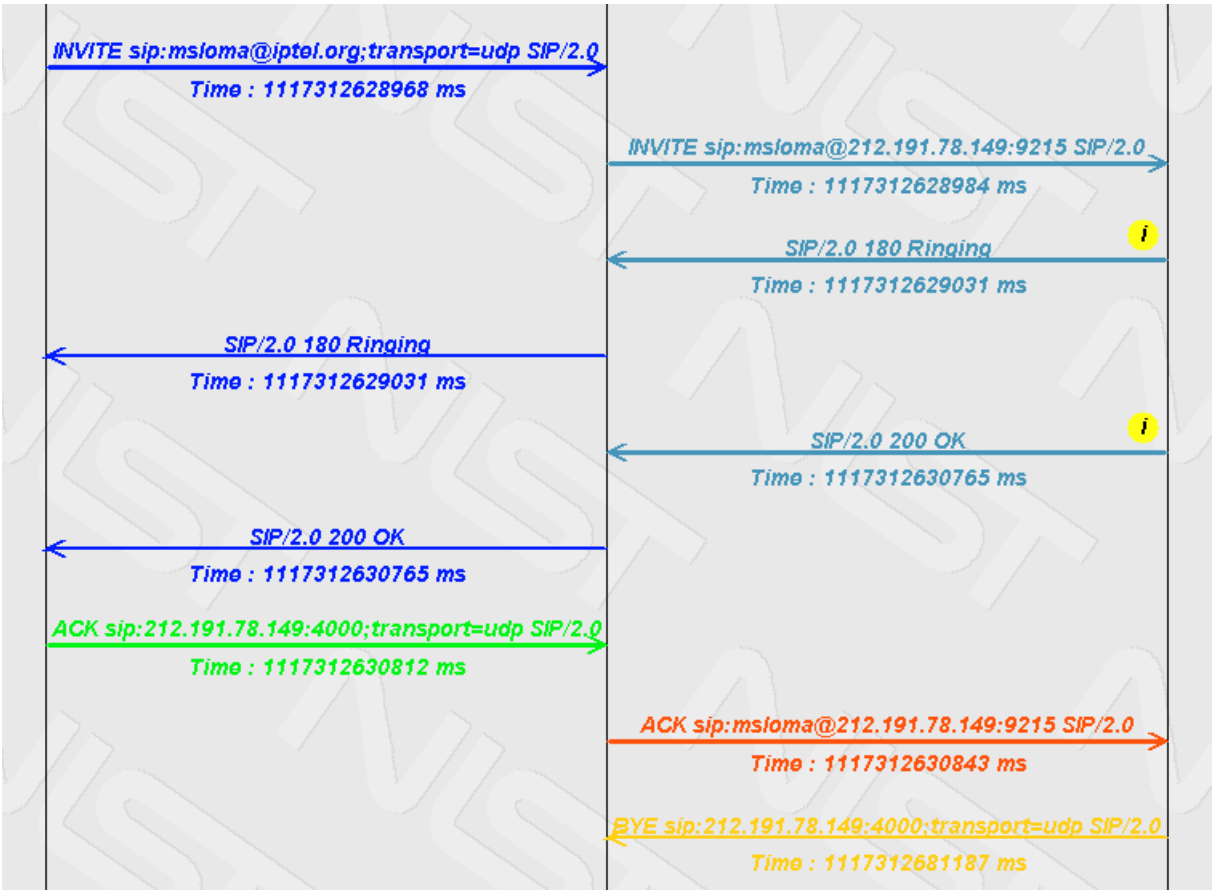
Pole Proxy-Authorization zbudowane jest z kilku części. Bezpośrednio po nazwie pola występuje metoda autoryzacji. Jest to metoda *Basic* lub *Digest*. Następnie wyróżniamy argumenty:

- username – nazwa użytkownika, który chce się autoryzować[35],
- realm – domena serwera[35],
- uri – adres serwera[35],
- algorithm – algorytm hashujący (mieszający)[79],
- opaque – łańcuch znaków generowany przez serwer, który musi być zwrócony w polu autentykacji w niezmienionym stanie przez klienta[79]
- nonce – łańcuch znaków generowany unikalnie przez serwer za każdym razem, gdy zwracana jest odpowiedź 401 Unauthorized[79]

- response – odpowiedź uwierzytelniająca, stworzona według metody autoryzacji na podstawie wartości powyższych argumentów oraz hasła.

Dalszym moim postępowaniem było znalezienie i skorzystanie z metod biblioteki NIST SIP Parser and Stack tak, aby zaimplementować pole nagłóweka Proxy-Authorization. Skorzystałem z klasy zaimplementowanej już dla autoryzacji pakietów REGISTER.
Po wprowadzeniu zmian aplikacji. Jej zachowanie podczas próby rozpoczęcia rozmowy głosowej jest następujące(patrz rysunek 6.8):

- wysłanie nieautoryzowanego pakietu INVITE,
- odebranie odpowiedzi 407 Authorization Required, które zawiera pole *Proxy-Authenticate* ze znacznikiem m.in. nonce potrzebnym do wygenerowania akceptowalnego żądania INVITE,
- wygenerowanie odpowiedzi potrzebnej do zbudowania poprawnego pola *Proxy-Authorization* w nagłówku INVITE,
- zbudowanie i wysłanie żądania INVITE zawierającego pole do autoryzacji,
- Serwer przesyła odpowiedź *100 Trying* co oznacza, że żądanie zostało przez serwer przyjęte i przesłane dalej.



Rys. 6.8 Schemat sesji w poprawionej aplikacji

6.7 Przeprowadzane testy

Wiadomości przedstawione w formie znacznikowej XML uzyskane podczas analizy działającego programu.

Testy zostały przeprowadzone w dla komputerów z publicznymi adresami IP.

Rozpoczęcie sesji (wysłanie pakietu REGISTER do serwera iptel.org)

```
REGISTER sip:iptel.org:5060 SIP/2.0
Call-ID: e36cd7f583bd16f08d43c4bf55db76f3@212.191.78.149
CSeq: 1 REGISTER
From: <sip:msloma@iptel.org>;tag=69
To: <sip:msloma@iptel.org>
Via: SIP/2.0/UDP 212.191.78.149:8193;branch=z9hG4bKda36626c48e78d5c0
ecab33b6cb411c0
Max-Forwards: 2
Contact: <sip:msloma@212.191.78.149:8193;transport=udp>
Content-Length: 0
```

Odpowiedź serwera o potrzebie autoryzacji

```
SIP/2.0 401 Unauthorized
Call-ID: e36cd7f583bd16f08d43c4bf55db76f3@212.191.78.149
CSeq: 1 REGISTER
From: <sip:msloma@iptel.org>;tag=69
To: <sip:msloma@iptel.org>;tag=b51ece8fd8c195776737473dd6552d43.6457
Via: SIP/2.0/UDP 212.191.78.149:8193;branch=z9hG4bKda36626c48e78d5c0
ecab33b6cb411c0;rport=8193
WWW-Authenticate: Digest realm="iptel.org",nonce="4293708edad57d1bfb
e62ec3eec76ecdeda9eb92"
Server: Sip EXpress router (0.9.0 (i386/linux))
Warning: 392 195.37.77.99:5060 "Noisy feedback tells:  pid=15082 req
_src_ip=212.191.78.149 req_src_port=8193 in_uri=sip:iptel.org:5060 out_uri=sip:i
ptel.org:5060 via_cnt==1"
Content-Length: 0
```

Aplikacja ma już wszystkie informacje potrzebne do autentykacji. Generuje zgodnie z metodą *Digest* odpowiedź autentykującą oraz ponawia zaproszenie do rozmowy głosowej.

Ponowna próba zarejestrowanie w serwerze

```
REGISTER sip:iptel.org:5060 SIP/2.0
Call-ID: e36cd7f583bd16f08d43c4bf55db76f3@212.191.78.149
CSeq: 2 REGISTER
From: <sip:msloma@iptel.org>;tag=69
To: <sip:msloma@iptel.org>
Via: SIP/2.0/UDP 212.191.78.149:8193;branch=z9hG4bK1947973e08ac6f5e7
39d89a9ac23fae4
Max-Forwards: 2
Contact: <sip:msloma@212.191.78.149:8193;transport=udp>
Authorization: Digest username="msloma",realm="iptel.org",uri="sip:i
ptel.org:5060",algorithm=MD5,opaque="",nonce="4293708edad57d1bfbe62ec3eec76ecded
a9eb92",response="352b65e071115d54335f81958b9984a1"
Content-Length: 0
```

Pozytywna odpowiedź serwera

```
SIP/2.0 200 OK
Call-ID: e36cd7f583bd16f08d43c4bf55db76f3@212.191.78.149
CSeq: 2 REGISTER
From: <sip:msloma@iptel.org>;tag=69
To: <sip:msloma@iptel.org>;tag=b51ece8fd8c195776737473dd6552d43.d305
Via: SIP/2.0/UDP 212.191.78.149:8193;branch=z9hG4bK1947973e08ac6f5e7
39d89a9ac23fae4;rport=8193
P-Role: master
Contact: <sip:msloma@212.191.78.149:8193;transport=udp>;expires=600
Server: Sip EXpress router (0.9.0 (i386/linux))
Warning: 392 195.37.77.99:5060 "Noisy feedback tells:  pid=15082 req
_src_ip=212.191.78.149 req_src_port=8193 in_uri=sip:iptel.org:5060 out_uri=sip:i
ptel.org:5060 via_cnt==1"
```

```
Content-Length: 0
```

Próba zarejestrowania użytkownika, aby otrzymywać o nim status obecności w sieci

```
SUBSCRIBE sip:slomi@iptel.org;transport=udp SIP/2.0
Call-ID: e31f4386642cd356b5c459398520137b@212.191.78.149
CSeq: 1 SUBSCRIBE
From: <sip:msloma@iptel.org>;tag=62
To: <sip:slomi@iptel.org>
Via: SIP/2.0/UDP 212.191.78.149:8193;branch=z9hG4bK0051b2df5445ab726
d2b1da87242cbc9
Max-Forwards: 2
Contact: <sip:msloma@212.191.78.149:8193;transport=udp>
Content-Length: 0
```

Odpowiedź odmowna serwera (potrzebna autentykacja)

```
SIP/2.0 407 Proxy Authentication Required
Call-ID: e31f4386642cd356b5c459398520137b@212.191.78.149
CSeq: 1 SUBSCRIBE
From: <sip:msloma@iptel.org>;tag=62
To: <sip:slomi@iptel.org>;tag=b51ece8fd8c195776737473dd6552d43.264b
Via: SIP/2.0/UDP 212.191.78.149:8193;branch=z9hG4bK0051b2df5445ab726
d2b1da87242cbc9;rport=8193
Proxy-Authenticate: Digest realm="iptel.org",nonce="429372b2221bccbf
f38f7225db5498116aa541ea"
Server: Sip EXpress router (0.9.0 (i386/linux))
Warning: 392 195.37.77.99:5060 "Noisy feedback tells:  pid=15086 req
_src_ip=212.191.78.149 req_src_port=8193 in_uri=sip:slomi@iptel.org;transport=ud
p out_uri=sip:slomi@iptel.org;transport=udp via_cnt==1"
Content-Length: 0
```

Próba rozpoczęcia rozmowy z użytkownikiem

```
INVITE sip:slomi@iptel.org;transport=udp SIP/2.0
Call-ID: f04d7be5fefee3f57b9cd8838dd98164@212.191.78.149
CSeq: 1 INVITE
From: <sip:msloma@iptel.org>;tag=1662
To: <sip:slomi@iptel.org>
Via: SIP/2.0/UDP 212.191.78.149:8193;branch=z9hG4bKa3a0eb9ab5e0eb5e1
48965c91bbd9dbc
Max-Forwards: 2
Contact: <sip:msloma@212.191.78.149:8193;transport=udp>
Content-Type: application/sdp
Content-Length: 118
```

Negatywna odpowiedź serwera (potrzebna autentykacja)

```
SIP/2.0 407 Proxy Authentication Required
Call-ID: f04d7be5fefee3f57b9cd8838dd98164@212.191.78.149
CSeq: 1 INVITE
From: <sip:msloma@iptel.org>;tag=1662
To: <sip:slomi@iptel.org>;tag=b51ece8fd8c195776737473dd6552d43.7fc7
```

```
Via: SIP/2.0/UDP 212.191.78.149:8193;branch=z9hG4bKa3a0eb9ab5e0eb5e1
48965c91bbd9dbc;rport=8193
Proxy-Authenticate: Digest realm="iptel.org",nonce="429372c3af223dbc
916215371dd845d725d2a3bc"
Server: Sip EXpress router (0.9.0 (i386/linux))
Warning: 392 195.37.77.99:5060 "Noisy feedback tells:  pid=15082 req
_src_ip=212.191.78.149 req_src_port=8193 in_uri=sip:slomi@iptel.org;transport=ud
p out_uri=sip:slomi@iptel.org;transport=udp via_cnt==1"
Content-Length: 0
```

Odpowiedź klienta (potwierdzenie)

```
ACK sip:slomi@iptel.org;transport=udp SIP/2.0
Call-ID: f04d7be5fefee3f57b9cd8838dd98164@212.191.78.149
CSeq: 1 ACK
From: <sip:msloma@iptel.org>;tag=1662
To: <sip:slomi@iptel.org>;tag=b51ece8fd8c195776737473dd6552d43.7fc7
Via: SIP/2.0/UDP 212.191.78.149:8193;branch=z9hG4bKa3a0eb9ab5e0eb5e1
48965c91bbd9dbc
Max-Forwards: 2
Content-Length: 0
```

Poprawiony UA wysyła komunikaty w nieco zmienionej formie.

Komunikat zarejestrowania w serwerze nie zawierał błędu, więc pozostał niezmieniony.

Komunikat zaproszenia drugiego użytkownika do rozmowy (najpierw bez autentykacji)

```
INVITE sip:msloma@iptel.org;transport=udp SIP/2.0
Call-ID: b0e3fe18bd4b72449eb2e716d6619e01@212.191.78.149
CSeq: 1 INVITE
From: <sip:slomi@iptel.org>;tag=6859
To: <sip:msloma@iptel.org>
Via: SIP/2.0/UDP 212.191.78.149:9317;branch=z9hG4bKb60313c052873e32a
1035eed46bc2aa1
MaxContact: <sip:slomi@212.191.78.149:9317;transport=udp>
Content-Type: application/sdp
Content-Length: 117
```

Odpowiedź serwera (potrzebna autentykacja)

```
SIP/2.0 407 Proxy Authentication Required
Call-ID: b0e3fe18bd4b72449eb2e716d6619e01@212.191.78.149
CSeq: 1 INVITE
From: <sip:slomi@iptel.org>;tag=6859
To: <sip:msloma@iptel.org>;tag=b51ece8fd8c195776737473dd6552d43.1a4a

Via: SIP/2.0/UDP 212.191.78.149:9317;branch=z9hG4bKb60313c052873e32a
1035eed46bc2aa1;rport=9317
Proxy-Authenticate: Digest realm="iptel.org",nonce="42938693fcb4504f
11c51c07de85152792b113fa"
```

```
Server: Sip EXpress router (0.9.0 (i386/linux))
Warning: 392 195.37.77.99:5060 "Noisy feedback tells:  pid=15086 req
_src_ip=212.191.78.149 req_src_port=9317 in_uri=sip:msloma@iptel.org;transport=u
dp out_uri=sip:msloma@iptel.org;transport=udp via_cnt==1"
Content-Length: 0
```

Potwierdzenie klienta otrzymania wiadomości

```
ACK sip:msloma@iptel.org;transport=udp SIP/2.0
Call-ID: b0e3fe18bd4b72449eb2e716d6619e01@212.191.78.149
CSeq: 1 ACK
From: <sip:slomi@iptel.org>;tag=6859
To: <sip:msloma@iptel.org>;tag=b51ece8fd8c195776737473dd6552d43.1a4a
Via: SIP/2.0/UDP 212.191.78.149:9317;branch=z9hG4bKb60313c052873e32a
1035eed46bc2aa1
Max-Forwards: 2
Content-Length: 0
```

Zaproszenie tym razem z autentykacją

```
INVITE sip:msloma@iptel.org;transport=udp SIP/2.0
Call-ID: c4a4de87260b3dd346ccf07809407fd0@212.191.78.149
CSeq: 1 INVITE
From: <sip:slomi@iptel.org>;tag=8696
To: <sip:msloma@iptel.org>
Via: SIP/2.0/UDP 212.191.78.149:9317;branch=z9hG4bKa85aa9dedffc1b695
0baa0b1b8bde06e
Max-Forwards: 2
Contact: <sip:slomi@212.191.78.149:9317;transport=udp>
Proxy-Authorization: Digest username="slomi",realm="iptel.org",uri="
sip:iptel.org:5060",algorithm=MD5,opaque="",nonce="42938693fcb4504f11c51c07de851
52792b113fa",response="ef84ee9319883c47a1523d082e48673a"
Content-Type: application/sdp
Content-Length: 117
```

Zawarte informacje w treści pakietu są typu SDP (Content-Type: application/sdp). Ciało pakietu wygląda następująco:

```
v=0
o=slomi 541482 542851 IN IP4 212.191.78.149
s=-
c=IN IP4 212.191.78.149
t=0 0
m=audio 3818 RTP/AVP 5 4 3 0
```

Odpowiedź od serwera na zaproszenie drugiego użytkownika

```
SIP/2.0 100 trying -- your call is important to us
Call-ID: c4a4de87260b3dd346ccf07809407fd0@212.191.78.149
CSeq: 1 INVITE
From: <sip:slomi@iptel.org>;tag=8696
To: <sip:msloma@iptel.org>
Via: SIP/2.0/UDP 212.191.78.149:9317;branch=z9hG4bKa85aa9dedffc1b695
```

```
0baa0b1b8bde06e;rport=9317
Server: Sip EXpress router (0.9.0 (i386/linux))
Warning: 392 195.37.77.99:5060 "Noisy feedback tells:  pid=15095 req
_src_ip=212.191.78.149 req_src_port=9317 in_uri=sip:msloma@iptel.org;transport=u
dp out_uri=sip:msloma@212.191.78.148:4029;transport=udp via_cnt==1"
Content-Length: 0
```

Za chwile użytkownik otrzymuje pakiet stanowiący o tym, że drugi użytkownik jest powiadomiony o zaproszeniu

```
SIP/2.0 180 Ringing
Record-Route: <sip:195.37.77.99;ftag=8696;lr=on>
Call-ID: c4a4de87260b3dd346ccf07809407fd0@212.191.78.149
CSeq: 1 INVITE
From: <sip:slomi@iptel.org>;tag=8696
To: <sip:msloma@iptel.org>
Via: SIP/2.0/UDP 212.191.78.149:9317;rport=9317;branch=z9hG4bKa85aa9
dedffc1b6950baa0b1b8bde06e
Max-Forwards: 1
P-NAT-Check: YES
Content-Length: 0
```

Gdy drugi użytkownik zaakceptuje rozmowę wysyłana jest odpowiedź 200 OK.

```
SIP/2.0 200 OK
Record-Route: <sip:195.37.77.99;ftag=8696;lr=on>
Call-ID: c4a4de87260b3dd346ccf07809407fd0@212.191.78.149
CSeq: 1 INVITE
From: <sip:slomi@iptel.org>;tag=8696
To: <sip:msloma@iptel.org>;tag=5611
Via: SIP/2.0/UDP 212.191.78.149:9317;rport=9317;branch=z9hG4bKa85aa9
dedffc1b6950baa0b1b8bde06e
Max-Forwards: 1
Contact: <sip:msloma@212.191.78.148:4029;transport=udp>
Content-Type: application/sdp
P-NAT-Check: YES
Content-Length: 112
```

Odpowiedź 200 OK zawiera parametry sesji dla mediów w swoim ciele. W tym przypadku było to:

```
v=0
o=msloma 209092 210461 IN IP4 212.191.78.148
s=-
c=IN IP4 212.191.78.148
t=0 0
m=audio 7586 RTP/AVP 5
```

Użytkownik inicjujący potwierdza otrzymanie odpowiedzi

```
ACK sip:msloma@212.191.78.148:4029;transport=udp SIP/2.0
Call-ID: c4a4de87260b3dd346ccf07809407fd0@212.191.78.149
```

```
CSeq: 1 ACK
From: <sip:slomi@iptel.org>;tag=8696
To: <sip:msloma@iptel.org>;tag=5611
Via: SIP/2.0/UDP 212.191.78.149:9317;branch=1455210040
Max-Forwards: 2
Route: <sip:195.37.77.99;ftag=8696;lr=on>
Content-Length: 0
```

W tym momencie ustanawiana jest sesja multimedialna.

Użytkownik chcąc zakończyć sesję multimedialną wysyła komunikat BYE i kończy sesję sygnalizacyjną i determinuje terminale uczestniczące w niej do zakończenia sesji multimedialnej.

```
BYE sip:msloma@212.191.78.148:4029;transport=udp SIP/2.0
Call-ID: c4a4de87260b3dd346ccf07809407fd0@212.191.78.149
CSeq: 2 BYE
From: <sip:slomi@iptel.org>;tag=8696
To: <sip:msloma@iptel.org>;tag=5611
Via: SIP/2.0/UDP 212.191.78.149:9317;branch=z9hG4bKdbffd57f572e27b8f
084f260a2925423
Max-Forwards: 2
Route: <sip:195.37.77.99;ftag=8696;lr=on>
Content-Length: 0
```

Otrzymuje odpowiedź

```
SIP/2.0 200 OK
Record-Route: <sip:195.37.77.99;ftag=8696;lr=on>
Call-ID: c4a4de87260b3dd346ccf07809407fd0@212.191.78.149
CSeq: 2 BYE
From: <sip:slomi@iptel.org>;tag=8696
To: <sip:msloma@iptel.org>;tag=5611
Via: SIP/2.0/UDP 212.191.78.149:9317;rport=9317;branch=z9hG4bKdbffd5
7f572e27b8f084f260a2925423
Max-Forwards: 1
P-NAT-Check: YES
Content-Length: 0
```

Sesja jest zakończona z obu stron

6.8 Konfiguracja sprzętowa i wymagania sprzętowe

Uruchomienie aplikacji wymaga komputera klasy PC z zainstalowaną wirtualną maszyną Javy (JVM Java Virtual Machine). Konieczne jest zainstalowanie JMF (Java Media Framework) czyli biblioteki do obsługi multimediów.
Testy przeprowadzane były na komputerze PC o parametrach:

- CPU: Sempron 2400+
- Pamięć RAM: 512 MB DDR
- OS: Windows 2003 Server
- Java: J2SDK 1.4.2[85]
- JMF: JMF v2.1.1e[75]

Wnioski

W pracy przybliżyłem funkcjonalność, możliwości zastosowania oraz podstawy działania jeszcze młodej dziedziny informatyki, jaką jest Telefonii IP. Zakres pracy obejmuje także omówienie protokołów komunikacyjnych używanych w telefonii IP. Bardziej szczegółowo zająłem się dwoma technologiami sygnalizacyjnymi: H.323 i SIP, które są rozwiązaniami najpopularniejszymi. Część praktyczna pracy obejmuje implementację oprogramowania klienckiego w technologii SIP. W pracy praktycznej został wykorzystany projekt na licencji open source[66] wspierany przez organizację NIST[67]. Aplikacja napisana jest w języku Java z wykorzystaniem dostępnych bibliotek: do obsługi multimediów oraz wspierająca technologię SIP. Aplikacja spełnia zadania uproszczonego terminala. Celem moim było rozbudowanie aplikacji tak, aby współpracowała z popularnym serwerem dedykowanym dla Telefonii IP - SER. Aplikacja rozszerzona została przeze mnie o obsługę uwierzytelniania pakietów, co dopiero pozwoliło na pomyślną współpracę z serwerem a co za tym idzie komunikację.
Działanie oprogramowania nie zostało jednak przetestowane z urządzeniami dedykowanymi jak telefon IP. Spowodowane to było brakiem dostępu do takiego urządzenia.
W związku z tym, że standard sygnalizacyjny SIP jest dość młody, rozwój jego jest nieunikniony. Pracę moją można będzie, więc rozwijać wraz z rozwojem standardu.
Kolejnym ambitnym zadaniem może być dostosowanie aplikacji do komunikacji z rozwiązaniami komercyjnymi takimi jak: serwer np. Cisco CallManager.

Bibliografia

1. D. Sisalem, J. Kuthan *„Understanding SIP”*, Mobile Integrated Services, GMD Fokus http://www.fokus.gmd.de/mobis/siptutorial/
2. T. Socolofsky, C. Kale *“A TCP/IP tutorial”* RFC 1180, styczeń 1991
3. H. Schulzrinne *„RTP News”*, luty 2004, http://www.cs.columbia.edu/~hgs/rtp/
4. H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson *„RTP: A Transport Protocol for Real-Time Applications”* RFC 1889, styczeń 1996
5. A. Jungmaier *„SCTP for Beginners”*, 2003, http://tdrwww.exp-math.uni-essen.de/inhalt/forschung/sctp_fb/
6. R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rycina, M. Kalla, L. Zhang, V. Paxson *„Stream Control Transmission Protocol”* RFC 2960, październik 2000
7. S. Berson „RSVP ReSerVation Protocol”, marzec 1999, http://www.isi.edu/div7/rsvp/rsvp.html
8. *„Gwarancja jakości usług w sieciach IP”*, Biuletyn Informacyjny Grupy SOLIDEX - ISSN 1233-4944 Nr 9-10/1999 (37), wrzesień 1999, http://integrator.solidex.pl/273_727.html
9. R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin *„Resource ReSerVation Protocol (RSVP)”* RFC 2205, wrzesień 1997
10. E. Kohler, M. Handley, S. Floyd, J.Padhye *„Datagram Congestion Control Protocol (DCCP)”*, marzec 2005, http://www.icir.org/kohler/dcp/
11. *„RTSP”* Networld Leksykon, 2005 http://www.networld.pl/leksykon/nw_term_info.asp?termin_nazwa=RTSP
12. H. Schulzrinne, A. Rao, R. Lanphier *„Real Time Streaming Protocol (RTSP)”* RFC 2326, kwiecień 1998
13. H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson *„RTP: A Transport Protocol for Real-Time Applications”* RFC 3550, lipiec 2003
14. „RTCP - Protocol Dictionary”, Javvin Company http://www.javvin.com/protocolRTCP.html
15. A. Karim *„H.323 and Associated Protocols”* Ohio State University http://www.cse.ohio-state.edu/~jain/cis788-99/h323/index.html
16. M. Rango, A. Dugan, I. Elliott, C. Huitema, S. Pickett *„Media Gateway Control Protocol (MGCP) Version 1.0”* RFC 2705, październik 1999
17. F. Cuervo, N. Greene, A. Rayhan, C. Huitema, B. Rosen, J. Segers *„Megaco Protocol Version 1.0”* RFC 3015, listopad 2000
18. A. Janikowski *„Rok z VoIP”*, Networld, maj 2001, http://www.networld.pl/artykuly/9715_2.html
19. A. Salamon *„DNS related RFCs”*, kwiecień 2004, www.dns.net/dnsrd/rfc/
20. „H.323”, Techbulary, http://www.techabulary.com/h/h323.html
21. http://www.packetizer.com/voip/h323/
22. T. Rogowski *„Głos w sieciach danych”*, PC kurier 26/1999, http://www.pckurier.pl/archiwum/art0.asp?ID=3682
23. A. Urbanek *„Głos w pakietach”* Networld, listopad 1999, http://www.networld.pl/artykuly/5722_1.html
24. *„Video Conferencing Standards & Terminology”*, TeamSolusions, luty 2005, http://www.teamsolutions.co.uk/tsstds.html Standardy używane w video konferencjach
25. T. Baur, M. Mandau, K. Sokołowski *„Lepszy niż DivX”* Chip 07/2004, lipiec 2004 http://www.chip.pl/arts/archiwum/n/articlear_106523.html

26. *„H.323 Dialing Tutorial”*, VTEL, luty 1998, http://www.vtel.com/support/galaxy/H323Dialing1.htm
27. K. Egevang, P. Francis *„The IP Network Address Translator (NAT)”* RFC 1631, maj 1994
28. *“H.323 On-line Education”* International Engineering Consortium, http://www.iec.org/online/tutorials/h323/
29. A. Karim *„H.323 and Associated Protocols”* Ohio State University http://www.cse.ohio-state.edu/~jain/cis788-99/h323/index.html
30. E. Kohler, M. Handley, S. Floryd *„Datagram Congestion Control Protocol (DCCP)”* draft-ietf-dccp-spec-11.txt, marzec 2005
31. J. Kuthan *„Selected IETF Internet Drafts and RFCs Relevant to the Internet Telephony”*, maj 2002 http://www.iptel.org/info/players/ietf/
32. *„IP-Telephony Products”* iptel.org http://www.iptel.org/info/products/index.php
33. *„SIP transports”*, wrzesień 2003 http://www.voip-info.org/wiki-SIP+transports
34. J. Postel *„User Datagram Protocol”* RFC 768, sierpień 1980
35. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler *„SIP: Session Initiation Protocol”* RFC 3261, czerwiec 2002
36. *„SIP”* Softfront, 2005 http://www.softfront.co.jp/en/tech/sip.html
37. *„Protokół SIP (Session Initiation Protocol)”* Alcatel, lipiec 2003 www2.alcatel.pl/dwn/wp_SIP-PL.pdf
38. *„Enum/E.164”*, ITU, www.itu.int/osg/spu/infocom/enum/
39. H. Schulzrinne *„The tel URI For Telephone Numbers”* RFC 3966, grudzień 2004
40. http://www.sipcenter.com
41. *„Internet Drafts”* IMC http://www.imc.org/ids.html
42. M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg *„SIP Session Initiation Protocol”* RFC 2543, marzec 1999
43. D. H. Crocker *„Standard for the format of ARPA Internet text messages”* RFC 822, sierpień 1982
44. S. Donovan *„The SIP INFO Method”* RFC 2976, październik 2000
45. H. Schulzrinne, J. Rosenberg, “*The IETF Internet Architecture and Protocols”*, IEEE Network 1999 http://www.computer.org/internet/telephony/w3schrosen.htm
46. J. Klensin *„Simple Mail Transfer Protocol (SMTP)”* RFC 2821, kwiecień 2001
47. R. Fielding, J. Gettys, J. Mosul, H. Erystyk, L. Masinter, P. Leach, T. Berners-Lee *„Hypertext Transfer Protocol - HTTP/1.1”* RFC 2616, czerwiec 1999
48. M. Handley, V. Jacobson *„SDP: Session Description Protocol”* RFC 2327, kwiecień 1998
49. M. Handley, C. Perkins, E. Whelan *„SAP: Session Announcement Protocol”* RFC 2974, październik 2000
50. J. Rosenberg, H. Schulzrinne *„SIP: Locating SIP Servers”* RFC 3263, czerwiec 2002
51. J. Rosenberg, H. Schulzrinne *“Reliability of Provisional Responses in the SIP”* RFC 3262, czerwiec 2002
52. A. B. Roach *“SIP - Specific Event Notification”* RFC 3265, czerwiec 2002
53. H. Schulzrinne, D. Oran, G. Camarillo *“The Reason Header Field for the Session Initiation Protocol (SIP)”* RFC 3326, grudzień 2002
54. J. Rosenberg *“The SIP UPDATE Method”* RFC 3311, sierpień 2002
55. B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, D. Gurle *“SIP - Extension for Instant Messaging”* RFC 3428, grudzień 2002
56. G. Camarillo *“Compressing the Session Initiation Protocol (SIP)”* RFC 3486, luty 2003
57. R. Sparks *“SIP - Refer Method”* RFC 3515, kwiecień 2003

58. R. Price, C. Bormann, J. Christoffersson, H. Hannu, Z. Liu, J. Rosenberg *"Signaling Compression (SigComp)"* RFC 3320, styczeń 2003
59. R. Mahy, D. Petrie *"The Session Initiation Protocol (SIP) "Join" Header"* RFC 3911, październik 2004
60. S. Donovan, J. Rosenberg *"Session Timers in the Session Initiation Protocol (SIP)"* RFC 4028, kwiecień 2005
61. J. Rosenberg, H. Schulzrinne, G. Camarillo *„The Stream Control Transmission Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP)"* draft-ietf-sip-sctp-06.txt, styczeń 2005
62. *„Simple Traversal of UDP Through Network Address Translators (STUN)"* Newport Networks, 2005 http://www.newport-networks.com/whitepapers/fwnatwpes3.html
63. J. Peterson , C. Jennings*„Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)"* draft-ietf-sip-identity-05.txt, marzec 2005
64. M. Barnes *"An Extension to the Session Initiation Protocol for Request History Information"* draft-ietf-sip-history-info-06.txt, styczeń 2005
65. http://www.ietf.org/html.charters/sip-charter.html
66. „Nieoficjalne tłumaczenie licencji open source", 2005 http://www.opensource.ite.pl/page/definicja.html
67. NIST http://www.nist.gov
68. NIST SIP http://snad.ncsl.nist.gov/proj/iptel/
69. Xten, www.xten.com
70. E.Luff „SIP Firewall Traversal and Security" Newport Networks, 2005 www.newport-networks.com/downloads/ SIP-FW-Trav-Security-041203.ppt
71. *„JAIN SIP API"* http://192.18.97.252/ECom/EComTicketServlet/BEGIN74347FE2D49C53CDB6CD359BDB2353A4/-2147483648/844211583/1/379814/379802/844211583/2ts+/westCoastFSEND/7578-jain_sip-1.0-fpr-spec-oth-JSpec/7578-jain_sip-1.0-fpr-spec-oth-JSpec:2/jainsip-fpr-docs.zip
72. „*NIST SIP Parser and Stack (v1.2) API"* http://www-x.antd.nist.gov/proj/iptel/src/nist-sip/jain-sip/docs/api/
73. C. Baray *„The model-view-controller (MVC) design pattern",* marzec 2002, http://www.cs.indiana.edu/~cbaray/projects/mvc.html
74. Komentarze w projekcie JAIN SIP Applet phone
75. http://java.sun.com/products/java-media/jmf/
76. H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, J. Peterson *"Presence Information Data Format: PIDF"* RFC 3863, sierpień 2004
77. Dokumentacja projektu JAIN SIP Applet phone
78. http://ant.apache.org
79. J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart *„HTTP Authentication: Basic and Digest Access Authentication"* RFC 2617, czerwiec 1999
80. https://jain-sip-applet-phone.dev.java.net/
81. http://pl.wikipedia.org/wiki/Reverse_engineering
82. http://www-306.ibm.com/software/awdtools/developer/rose/support/
83. https://jain-sip-presence-proxy.dev.java.net/
84. https://sip-communicator.dev.java.net/
85. http://java.sun.com/

86. W. Mazurczyk *„Bezpieczeństwo SIP jako protokołu sygnalizacyjnego VoIP”* XIX Krajowe Sympozjum Telekomunikacji KST 2003, wrzesień 2003, http://home.elka.pw.edu.pl/~wmazurcz/moja/art/kst2003.pdf

Rysunki

3.1 http://www.cs.columbia.edu/~hgs/internet/
3.2 http://www.iptel.org
3.3 rysunek własny
3.4 rysunek własny
4.1 http://www.packetizer.com/voip/h323/papers/h323_protocol_overview_files/frame.html
4.2 http://www.cse.ohio-state.edu/~jain/cis788-99/h323/index.html
5.1 http://www.softfront.co.jp/en/tech/sip.html
5.2 http://www.softfront.co.jp/en/tech/sip.html
5.3 http://www.sipcenter.com/sip.nsf/html/Testing+User+Agents
5.4 http://www.sipcenter.com/sip.nsf/html/Testing+User+Agents
6.2 http://jcp.org/aboutJava/communityprocess/first/jsr032/index.html
6.3 http://jcp.org/aboutJava/communityprocess/first/jsr032/index.html
6.4 http://www.cs.indiana.edu/~cbaray/projects/mvc.html
6.5 Printscreen z programu JCreator przedstawiający pliki projektu
6.6 Printscreen z programu JCreator przedstawiający pliki projektu
6.7 Printscreen z Proxy serwer viewer integralnej części JAIN SIP Presence proxy
6.8 Printscreen z Proxy serwer viewer integralnej części JAIN SIP Presence proxy
6.9 Diagram z programu Rational Rose Enterprise Edition

Tabele

4.1 *„H.323 Dialing Tutorial”*, VTEL, luty 1998, http://www.vtel.com/support/galaxy/H323Dialing1.htm
4.2 A. Karim *„H.323 and Associated Protocols”* Ohio State University http://www.cse.ohio-state.edu/~jain/cis788-99/h323/index.html
5.1 M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg *„SIP Session Initiation Protocol”* RFC 2543, marzec 1999
5.2 *„NIST SIP Parser and Stack (v1.2) API”* http://www-x.antd.nist.gov/proj/iptel/src/nist-sip/jain-sip/docs/api/
5.3 S. Donovan, J. Rosenberg *“Session Timers in the Session Initiation Protocol (SIP)”* RFC 4028, kwiecień 2005

Listingi

6.1 J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler *„SIP: Session Initiation Protocol”* RFC 3261, czerwiec 2002