

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Telekomunikacja

SPECJALNOŚĆ: Teleinformatyka i multimedia

Praca Dyplomowa Magisterska

Koncepcja symulatora do realizacji i analizy
usług multimedialnych

A concept of Simulator for implementation
and analysis multimedia services

AUTOR:

Kamil Szymański

PROWADZĄCY PRACĘ:

Dr inż, Tomasz Długosz

OCENA PRACY:

WROCŁAW 2015

Spis treści

1. Wstęp	3
2. Technologia VoIP	4
3. Protokoły sygnalizacyjne	5
3.1. Protokół H.323.....	5
3.1.1. Charakterystyka protokołu H.323	6
3.1.2. Architektura sprzętowa	9
3.2. SIP	10
3.2.1. Charakterystyka SIP	10
3.2.2. Rodzaje wiadomości	12
3.2.3. Architektura sieciowa.....	14
4. Porównanie protokołów SIP i H.323	16
5. Aplikacje symulujące protokół SIP	21
5.1. SipInspector	21
5.2. Wireshark.....	22
5.3. SipP.....	24
5.4. MAPS SIP protocol emulator	25
6. Autorska aplikacja	27
6.1. Opis interfejsu graficznego.....	27
6.1.1. Zakładka Input data	28
6.1.2. Zakładka Create/Load Scenario	28
6.1.3. Zakładka symulacji	30
6.2. Zasada działania	30
6.2.1. Zakładka Input data.....	31
6.2.2. Zakładka Create/Load Scenario	31
6.2.3. Zakładka symulacji	33
6.2.4. Pozostałe.....	35
6.3. Porównanie symulatorów	37
7. Podsumowanie	39
8. Bibliografia	40
9. Instrukcja laboratoryjna oraz obsługi aplikacji.....	41
9.1. Instrukcja laboratoryjna	41
9.2. Instrukcja obsługi	41

1. Wstęp

Na przestrzeni wielu lat Internet wyewoluował do takiego stadium, że bez niego większość ludzi, firm, instytucji oraz państw nie mogłyby rozwijać się w tak szybkim tempie jak dotychczas. Stworzony pierwotnie do połączenia super-komputerów pomiędzy ośrodkami pracującymi dla wojsk w USA stał się globalną siecią łączącą miliony komputerów, serwerów czy telefonów. Można śmiało określić go mianem jednym z największych narzędzi XXI wieku, dzięki któremu wymiana informacji na całym świecie jest możliwa praktycznie w czasie rzeczywistym. Dotarciu do takiego stanu rzeczy przyczyniło się powstanie protokołów sieciowych TCP/IP, UDP, WWW, DNS, modelu ISO/OSI czy rozwoju VoIP. Dzięki takim kamieniom milowym Internet rozwinął się do takiego stanu, jaki można aktualnie obserwować.

Przedmiotem niniejszego opracowania jest omówienie jednego z takich odkryć, jakim jest telefonia VoIP oraz analiza protokołów sygnalizacyjnych SIP, H.323. Dzięki wymianie informacji w czasie rzeczywistym Internet zyskał kolejnych zwolenników, którzy to stali się jego częścią i wspomogli rozwój coraz dalej rozwijających się technologii telefonii internetowej. W aktualnych czasach ciężko wyobrazić sobie brak dostępu do takich serwisów i oprogramowań jak facebook, skype, teamspeak czy viber, a które to opierają się właśnie na owej technologii. Można, więc wywnioskować, że VoIP stał się nieodłączną częścią znaczącej ilości ludzi.

Obecnie rozwój sieci zmierza do unifikacji, dzięki której wymagane jest utrzymywanie wyłącznie jednej architektury kablowej i wyeliminowaniu wszelkich specjalistycznych urządzeń sieciowych jak np. centralka telefoniczna. Usługa VoIP polega na cyfrowej reprezentacji sygnału mowy, poddaniu go odpowiedniej kompresji i segregacji na pakiety. Taki strumień pakietów jest następnie przesyłany za pomocą sieci pakietowej wraz z innymi danymi pochodzącymi na przykład od komputerów. W węźle odbiorczym cały proces jest odtwarzany w odwrotnym kierunku, dzięki czemu otrzymujemy naturalny sygnał głosu.[1]

Celem pracy jest analiza porównawcza protokołów sygnalizacyjnych technologii VoIP, przedstawienie utworzonego symulatora protokołu SIP wraz z prezentacją i porównaniem oprogramowań skierowanych do celów dydaktycznych.

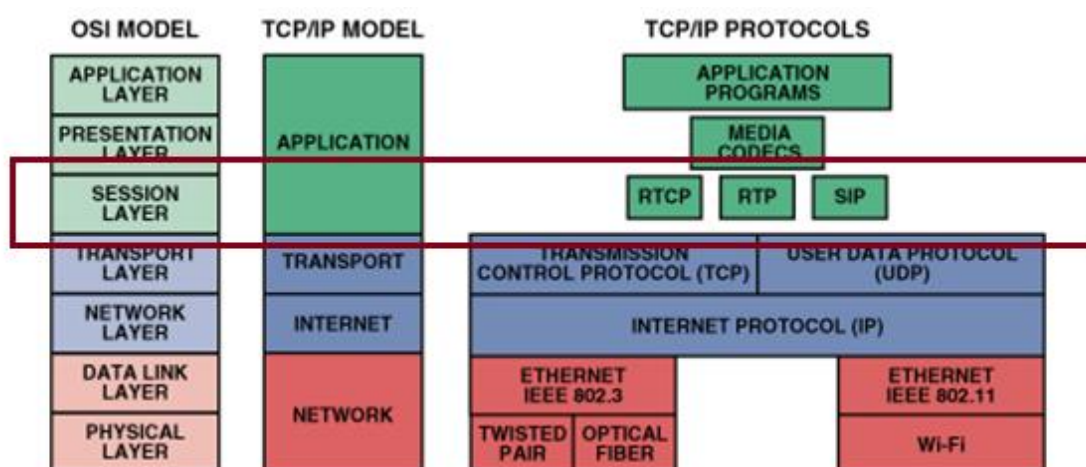
2. Technologia VoIP

Ta część pracy została poświęcona na omówienie podstawowych zagadnień z technologii telefonii VoIP.

VoIP służy do przesyłania dźwięku, obrazów (również ruchomych) poprzez sieci lokalne lub Internet używając protokołu IP. Dane są transportowane w pakietach, które składają się z nagłówka oraz kontenera danych. Nagłówek zawiera informacje potrzebne do odnalezienia urządzenia docelowego, a kontener zawiera skompresowane dane audio-wizualne. Pakiety te przemieszczają się po sieci przy pomocy jednego lub wielu protokołów transportowych. W miejscu przeznaczenia pakiet jest dekodowany i zamieniany z powrotem na sygnał audio-wizualny, który możliwy jest do odczytania przez użytkownika.

Siedmiowarstwowy Model OSI (Open Systems Interconnection) (*Rysunek 1. Prezentacja protokołów sygnalizacyjnych na modelach OSI, oraz TCP/IP*) specyfikuje zalecenia wraz z podziałem sieciowym. Jeśli sesja komunikacyjna połączeniowa pomiędzy dwoma stronami, dane, przez każdą z nich, generowane są na samym szczycie modelu, przechodząc niżej wraz z odpowiednim przetwarzaniem w każdej warstwie. Ostatecznie dane są dostarczane do warstwy fizycznej w celu dostarczeniu go do medium i przetransportowaniu go do miejsca przeznaczenia, w którym to cały proces odbywa się w odwrotnej kolejności[7].

Pierwszym wymogiem stawianym VoIP jest system kontroli sesji w celu określenia dostępności i lokalizacji użytkownika, jak również do zestawiania, modyfikacji i kończenia sesji. Obecnie istnieją dwa protokoły sygnalizacyjne, które przodują na tej przestrzeni. Pierwszym z nich był H.323, a jeszcze w tym samym roku powstał SIP (Session Initiation Protocol), który szybko zyskał na popularności. Na poniższym rysunku wydzielono strefę działania tych protokołów.



Rysunek 1. Prezentacja protokołów sygnalizacyjnych na modelach OSI, oraz TCP/IP

3. Protokoły sygnalizacyjne

Do realizacji telefonii internetowej niezbędne jest wykorzystywanie protokołów umożliwiających sterowanie przebiegiem transmisji w bezpołączeniowym środowisku sieci pakietowej IP. Ten właśnie element, czyli sygnalizacja umożliwiająca ustanawianie i sterowanie przebiegiem sesji wyróżnia w znacznym stopniu telefonię internetową od przekazu innych strumieni w sieci IP. Protokoły sygnalizacyjne stanowią serce telefonii internetowej i są niezbędne do realizacji zaawansowanych usług, oraz do zapewnienia współdziałania z tradycyjnymi sieciami telefonicznymi. Protokół sygnalizacyjny w telefonii internetowej powinien realizować takie funkcje, jak:

- translacja adresów i lokalizacja użytkownika wywoływanego,
- otwieranie i zamykanie sesji,
- negocjacja parametrów sesji (np. sposobu kompresji strumieni informacji) oraz ich zmiana w trakcie połączenia,
- zarządzanie grupą uczestników sesji,
- zarządzanie przebiegiem połączenia.

Obecnie nie ma ogólnie przyjętego standardu sygnalizacyjnego dla telefonii IP, jednak istnieją dwa najbardziej liczące się rozwiązania. Są to Zalecenie ITU-T H.323 oraz zdefiniowany przez IETF (Internet Engineering Task Force) protokół SIP (Session Initiation Protocol). Aplikacje audiowizualne korzystają głównie z tych dwóch najbardziej popularnych protokołów sygnalizacyjnych, więc to one zostaną opisane w kolejnych podrozdziałach.

3.1. Protokół H.323

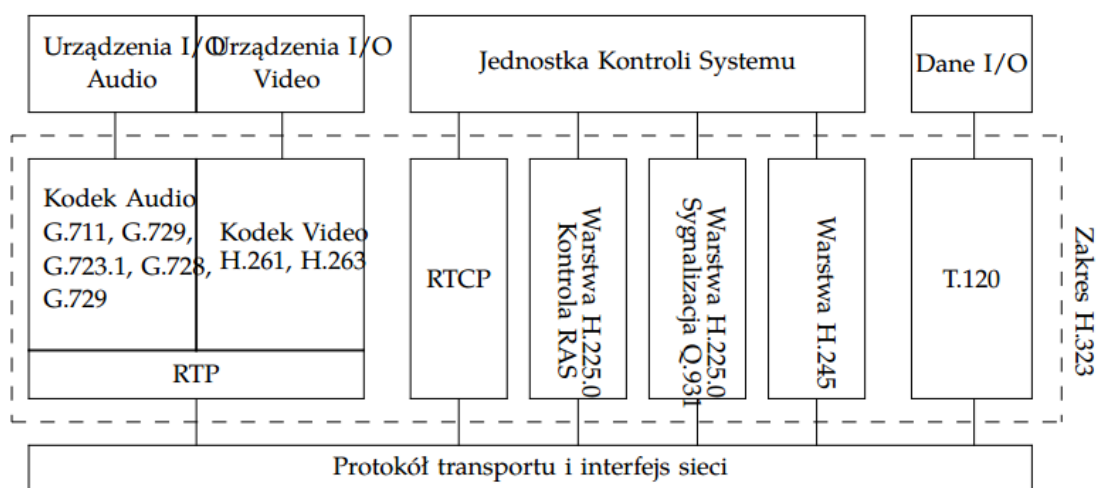
Protokół H.323 powstał w roku 1996 za sprawą instytucji ITU-T oraz IETF, który to w rzeczywistości jest zbiorem zaleceń, umożliwiającym realizację usług multimedialnych czasu rzeczywistego wyłączając QoS (Quality of Service). Obejmuje kontrolę połączenia, multimediiów, sygnalizację i transport. Poszczególne protokoły zawarte w stosie H.323 to:

- H.225.0 - stosowany do sygnalizacji połączenia;
- Q.931 - protokół zapożyczony od ISDN, również stosowany do sygnalizacji połączeń;
- H.245 - zaimplementowany do ustalenie parametrów kanału audiowizualnego;
- H.235 – stosowany w celach bezpieczeństwa i uwierzytelnienia;
- RTP, (Real Time Protocol) - protokół czasu rzeczywistego zdefiniowany przez IETF, używany do transmisji strumieni audio/video;
- H.450.x - używany do dodatkowych usług takich jak przekazywanie/zawieszanie itp.

3.1.1. Charakterystyka protokołu H.323

Niektóre kluczowe funkcje i założenia stosu protokołów H.323 zostały opisane poniżej:

- Zakres zestawu zaleceń H.323 nie obejmuje implementacji przechwytywania strumieni audio/video. Założone zostało, iż przetwarzanie owych strumieni zostaje przeprowadzone po stronie terminala.
- Zestawienie protokołów RTP (real-time transport protocol) oraz RTCP (real-time control protocol) ma za zadanie przesyłanie strumieni audio/video.
- Stos protokołów H.323 znajduje się na szczycie warstwy sieciowej i transportowej. Jeśli siecią, na której opiera się H.323 jest sieć IP to pakiety audio, video i H.225.0 RAS korzystają z protokołu bezpołączeniowego UDP (user datagram protocol) podczas gdy pakiety danych i sterowania (H.245 oraz H.225.0 sygnalizacji połączenia) są przysyłane za pomocą protokołu połączeniowego TCP (transmission control protocol).



Rysunek 2. Stos protokołów H.323[3]

Kontrola połączenia jest istotną częścią w telefonii VoIP, zestawienie i rozłączanie połączeń, zdolność wymiany informacji i negocjacji do celów administracyjnych. H.323 używa trzech protokołów wspólnie się uzupełniających: H.245 kontrola mediów, H.225/Q.931 sygnalizacja połączeń oraz H.225.0 RAS, które zostały opisane poniżej:

H.225.0 Sygnalizacja połączeń

Sygnalizacja połączeń jest podstawowym wymogiem zestawienia i rozłączania połączeń pomiędzy dwoma końcowymi punktami w sieci. H.225.0 używa do tego celu

protokołu Q.931. Q.931, pierwotnie opracowany dla sieci ISDN, został zaadaptowany włączając w to format wiadomości sygnalizacyjnych. Sygnalizacja H.225.0 jest bezpośrednio przesyłana pomiędzy urządzeniami końcowymi. Jednakże jeśli na trasie sygnalizacji istnieje GK, wiadomość może zostać przez niego przetrasowana. Wiadomościami wykorzystywanymi przez ten protokół są:

- Setup and Setup acknowledge,
- Call Proceeding,
- Connect,
- Alerting,
- Information,
- Release Complete,
- Facility,
- Progress,
- Status and Status Inquiry,
- Notify.

H.245 Kontrola mediów

Elastyczność H.323 wymaga negocjacji pomiędzy urządzeniami końcowymi kompatybilnych ustawień parametrów, przed ustaleniem połączenia audio/video. W trakcie trwania połączenia protokół ten nadzoruje i determinuje parametry takie jak, jitter, szyfrowanie, określenie relacji master i slave, otwieranie i zamykanie kanałów służących do transportu mediów. H.245 jest protokołem obowiązkowym we wszystkich skrajnych punktach w sieci. Główne funkcje, które spełnia H.245 są opisane poniżej:

- Uwarunkowania wymiany – każdy z punktów końcowych może odbierać i wysyłać wiadomości z innymi parametrami (bitrate, kodeki, rodzaj wiadomości)
- Otwieranie i zamykanie kanałów logicznych – kontroluje zamykanie i otwieranie odseparowanych kanałów audio/video
- Kontrola przepływu wiadomości – w przypadku wystąpienia zakłócenia w dostarczeniu wiadomości urządzenie końcowe otrzymuje informacje o takim zdarzeniu
- Inne komendy i wiadomości – kilka innych wiadomości i komend może być użyta w trakcie trwającego połączenia np. ustawienie kodeka.

Komendy, które są wymieniane pomiędzy terminalami w celu ustalenia parametrów, rozpoczęcia i zakończenia sesji, to:

- Master-slave determination
- Terminal capability set
- Open logical channel
- Close logical channel
- Request mode
- Send terminal capability set
- End session

Dozwolonych odpowiedzi na wyżej wymienione komendy jest zestaw 4 intuicyjnych wiadomości:

- Acknowledge
- Reject
- Confirm
- Release

H.225.0 RAS

H.225.0 RAS (registration, admission, status) determinuje komunikacje pomiędzy urządzeniami końcowymi a GK, jest używany wyłącznie gdy takowy GK znajduje się w konfiguracji sieciowej. W sieciach IP korzysta z protokołu bezpołączeniowego UDP. Najważniejsze cechy tego protokołu to:

- Rozpoznanie Gatekeepera – W celu dowiedzenia się urządzeń końcowych o adresie GK rozsyłana jest wiadomość rozgłoszeniowa (GRQ), na którą mogą otrzymać odpowiedź od GK (GCF) z zawartym adresem.
- Rejestracja urządzeń końcowych – Gdy GK znajduje się w sieci wszystkie urządzenia końcowe muszą zostać w nim zarejestrowane.
- Lokalizacja urządzeń końcowych – GK używają tego rodzaju wiadomości w celu odnalezienie odpowiednich urządzeń końcowych
- Inne aktywności – GK wykonują wiele innych zadań zarządzających siecią takie jak, kontrola dostępu, określenie stanu czy zarządzanie pasmem.

Możliwe zadania, które są kierowane do GK:

- Registration request
- Admission request
- Bandwidth request
- Disengage request
- Info request

Odpowiedziami mogą być natomiast poniższe wiadomości:

- XCF żądanie rozpatrzone pozytywnie
- XRJ żądanie rozpatrzone negatywnie

Gdzie X jest pierwszą literą żądania np. odpowiedź pozytywna na żądanie Admission request, wygląda następująco: ACF

3.1.2. Architektura sprzętowa

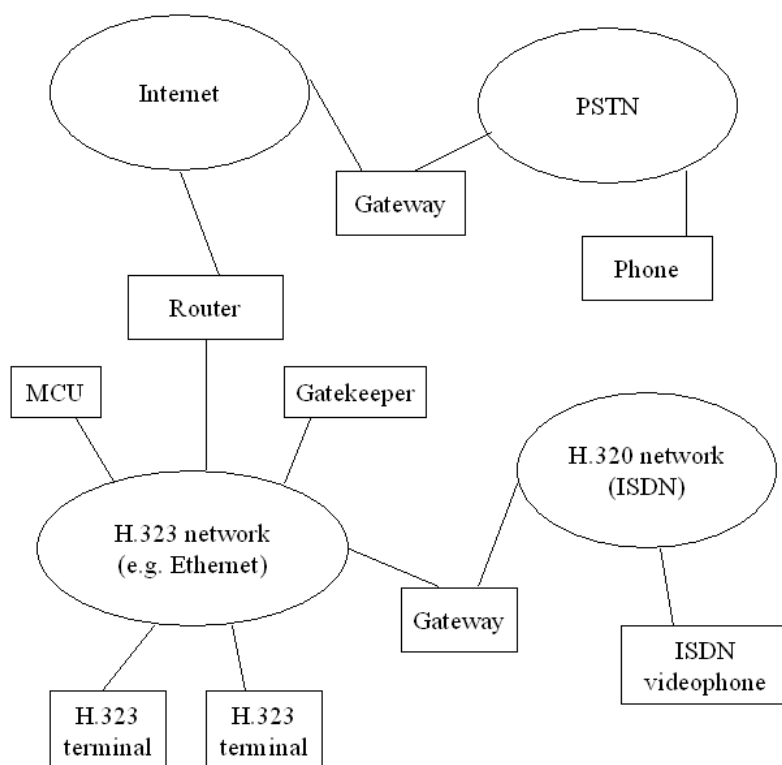
Terminal – Jest zwykłym telefonem VoIP w wersji sprzętowej lub programowej. Mają za zadanie inicjacje oraz odbieranie zgłoszeń dotyczących komunikacji. Dzięki odpowiednim konfiguracjom mogą współpracować z zewnętrznymi sieciami takimi jak PSTN czy ISDN.

Gateway (brama) – Urządzenie pozwalające na dwukierunkową komunikację pomiędzy urządzeniami w odrębnych sieciach. Zazwyczaj jest pomostem pomiędzy sieciami PSTN i IP, ale równie możliwa jest konfiguracja H.323-to-SIP lub nawet H.323-H.323. Brama zazwyczaj składa się z dwóch logicznych części zawartych w jednym urządzeniu:

- Media Gateway Controller (MGC) – obsługuje sygnalizację połączeń
- Media Gateway (MG) – zarządza strumieniami audio i video

Multipoint Conference Unit (MCU) – jednostka MCU używana jest do połączenia wielopunktowego (konferencji). Również podzielona jest na dwie części logiczne Multipoint Controller (MC) i Multipoint Processor (MP), z czego ten ostatni odpowiada za miksowanie kanałów audio/video konferencji, a MC za podejmowanie decyzji o podłączeniu klienta do konferencji.

Wszystkie powyższe urządzenia są skrajnymi punktami infrastruktury sieci (endpoints). Opcjonalną częścią sieci jest Gatekeeper (GK), który to odpowiada za zarządzanie przydzielonym mu obszarem sieci, wydzielonym od pozostałych jej elementów. Najważniejszymi z jego zadań są rejestracja urządzeń końcowych, translacja adresów, kontrola pasma oraz przyjmowanie połączeń. Zestawem urządzeń końcowych zarządzanych przez jednego GK nazywa się strefą.



Rysunek 3. Architektura sprzętowa H.323 [3]

3.2. SIP

Protokół SIP jest alternatywnym rozwiązaniem w stosunku do protokołów zdefiniowanych w ramach Zalecenia H.323. Opracowany został przez grupę roboczą IETF MMUSIC (IETF Multiparty Multimedia Session Control). Zasadnicze założenia protokołu zostały sformułowane również w 1996 roku. SIP został zaprojektowany uwzględniając przy tym uwarunkowania Internetu, czyli według wcześniej znormalizowanych przez IETF koncepcji i protokołów. Dzięki czemu spełnia wymagania szerokiego zakresu skalowalności.

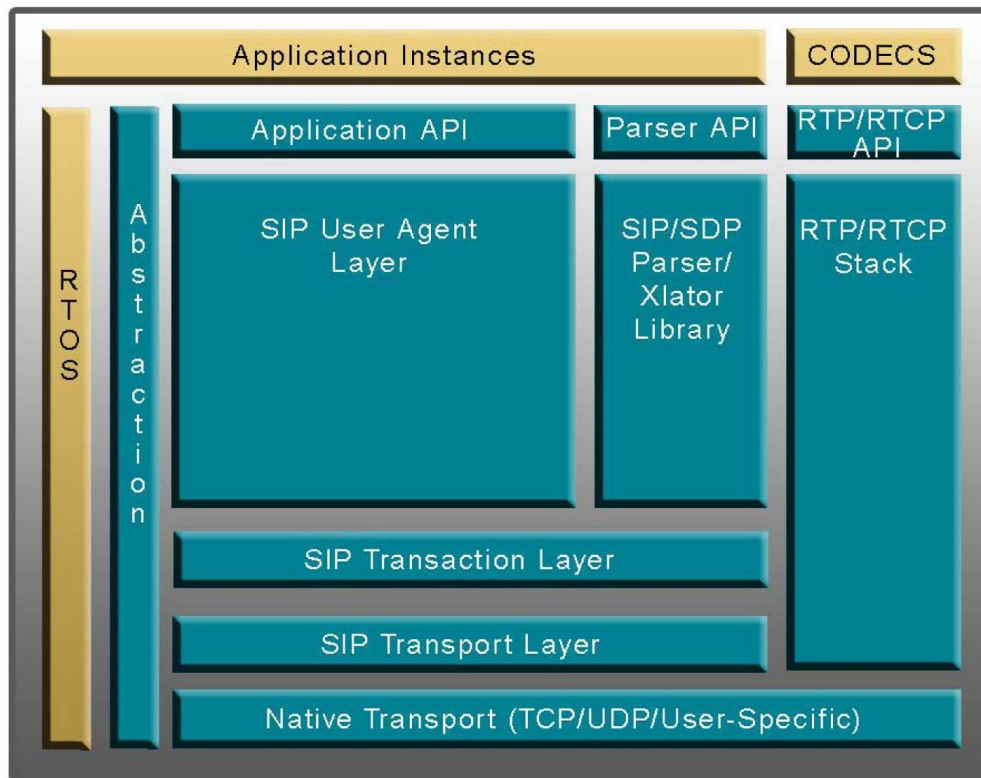
3.2.1. Charakterystyka SIP

SIP jest protokołem sterującym warstwy aplikacji, zajmujący się kontrolą sesji multimedialnych. Dzięki prostocie funkcjonowania protokołu, łatwości implementacji i skalowalności przykuł on uwagę wielu użytkowników i stale wypiera protokół H.323 z sieci VoIP.

Sam protokół nie jest jednak niezależną częścią sieci, a zaprojektowany został, jako część ogólnej architektury systemów multimedialnych, które obejmują:

- RSVP - służący do rezerwacji zasobów sieciowych,

- RTP - służący do transportu danych w trybie czasu rzeczywistego,
- SDP (Session Description Protocol) - służący do opisu sesji multimedialnych,
- SAP (Session Announcement Protocol) - służący do ogłaszania sesji multimedialnych za pomocą trybu multicast,
- RTSP (Real Time Streaming Protocol) - służący do transportu strumienia multimedialnego.



Rysunek 4. Stos protokołów SIP

SIP służy do zestawiania sesji pomiędzy użytkownikami, konferencji multimedialnej bądź telefonicznej lub transmisji rozsiewczej sygnału multimedialnego. Oznacza to, że połączenia mogą odbywać się w trybie unicast oraz multicast. Najważniejszymi cechami tego protokołu są:

- Podobnie jak w H.323 możliwość zmian parametrów podczas połączenia
- Mobilność – możliwość inicjowania, odbierania połączeń oraz dostępu do własnego profilu w każdym miejscu zalogowania
- Skalowalność, elastyczność, rozszerzalność
- Identyfikacja użytkowników – standard stara się aby nowo zalogowani użytkownicy zostali automatycznie odnalezieni w sieci
- Identycznie jak w H.323 możliwość korzystania z obu protokołów transportowych UDP i TCP

- Wiadomości są oparte na bazie czystego tekstu zaczerpniętego z formatowania protokołu HTTP
- SIP, w celu identyfikacji korzysta z SIP URI, który w skrócie jest numerem telefonu użytkownika. Ma format podobny do adresu poczty elektronicznej.

3.2.2. Rodzaje wiadomości

Wiadomości w protokole SIP dzieli się na żądania (requests) i odpowiedzi (responses). Natomiast format takich wiadomości Ma ustalony format:

- Start Line – request Line dla żądań oraz status line dla odpowiedzi
- Header – może występować wielokrotnie orz pod różnymi wariantami, które zostaną opisane w dalszej części pracy
- Body – opcjonalne wewnątrz wiadomości

Przykładowa wiadomość SIP (żądanie INVITE [4]):

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74b43
Max-Forwards: 70
Route: <sip:ssl.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 151
```

```
v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Nagłówki:

Nagłówki mają podobną strukturę jak nagłówki w HTTP. Mogą być rozszerzane na kilka linii pod warunkiem, że na końcu zawierają spację. Warto zaznaczyć, że wiadomość o charakterze żądania musi posiadać co najmniej sześć pól: To, From, Cseq, Call-ID, Max-Forwards oraz Via.

- Via - zawiera adres , na który użytkownik spodziewa się otrzymać odpowiedź na żądanie. Zawarty jest również parametr branch, który pozwala na identyfikację transakcji.
- Max forwards - Służy do ograniczenia liczby skoków żądania, które może przeprowadzać na drodze do miejsca przeznaczenia. Składa się z liczby całkowitej, która jest pomniejszana o jeden przy każdym przeskoku.
- Route - Określa trasę wiadomości.
- From - zawiera także wyświetlaną nazwę użytkownika żądającego oraz SIP URI lub SIPS, które wskazują, zleceniodawcę żądania. To pole nagłówka posiada również parametr tag zawierający losowy tekst, który został dodany do URI przez softphone. Jest on używany do celów identyfikacyjnych.
- To - Zawiera wyświetlaną nazwę użytkownika docelowego oraz SIP URI lub SIPS do którego żądanie zostało skierowane.
- Call-ID - Identyfikuje jednoznacznie sesję lub wiadomości rejestracyjne. Generowany losowo w sposób zapewniający globalną unikalność
- Cseq - Numer CSeq jest zwiększany dla każdego nowego żądania podczas dialogu i jest nim kolejna liczba całkowita.
- Contact: Zawiera SIP URI lub SIPS, który reprezentuje bezpośrednią drogę do skontaktowania się z użytkownikiem żądającym, zwykle składające się z nazwy użytkownika lub adresu IP. Podczas gdy pole nagłówka Via mówi gdzie wysłać odpowiedź, pole kontaktowe nagłówek mówi, gdzie wysłać przyszłe żądania.
- Content-Type - Typ treści ciała wiadomości.
- Content-Length - Długość wiadomości w bajtach
- Expires – wartość pola determinuje datę i czas, po której żądanie wygasa i staje się nieważne.
- Authorization – pole służy do autoryzacji użytkownika UA, jest ono opcjonalne
- Body – zazwyczaj jest pustym polem. Może przysyłać dane w innym protokole lub dane przeznaczone do odczytu przez odbiorcę wiadomości.

Żądania (Requests)

Istnieje sześć podstawowych rodzajów żądań:

- INVITE – zapoczątkowanie wywołania przez zaproszenie użytkownika do sesji.
- ACK – potwierdza odebranie odpowiedzi na żądanie INVITE.
- BYE – zakończenie wywołania.
- CANCEL – anulowanie żądania trwającego.
- REGISTER – rejestrowanie agenta użytkownika i jego lokalizacji (nazwy hosta, IP)

- OPTIONS – komunikowanie wiadomości o możliwościach wywołującego i wywoływanego telefonu SIP.

Odpowiedzi (Responses)

Na żądania SIP odsyłane są odpowiedzi SIP. Wyróżnia się 6 klas kodów odpowiedzi w SIP:

- 1xx – Wiadomości informacyjne.
- 2xx – Odpowiedzi pozytywne.
- 3xx – Odpowiedzi przekierowania.
- 4xx – Odpowiedzi błędnych żądań.
- 5xx – Odpowiedzi błędu serwera.
- 6xx – Odpowiedzi błędów globalnych.

Najpopularniej występującymi odpowiedziami podczas negocjacji połączenia są:

- 100 Trying (Sprawdzenie dostępności),
- 180 Ringing (Dzwonienie),
- 200 OK
- 301 Moved Permanently (Przeniesiony na stałe),
- 302 Moved Temporarily (Przeniesiony tymczasowo),
- 403 Forbidden (Niedozwolone),
- 407 Proxy Authentication Required (Konieczna autentyfikacja pośrednika),
- 480 Temporarily Unavailable (Czasowo niedostępny),
- 486 Busy Here (Zajęte).

3.2.3. Architektura sieciowa

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

- Terminale – urządzenia końcowe
- Serwery – prowadzą różne usługi związane z ułatwieniem komunikacji

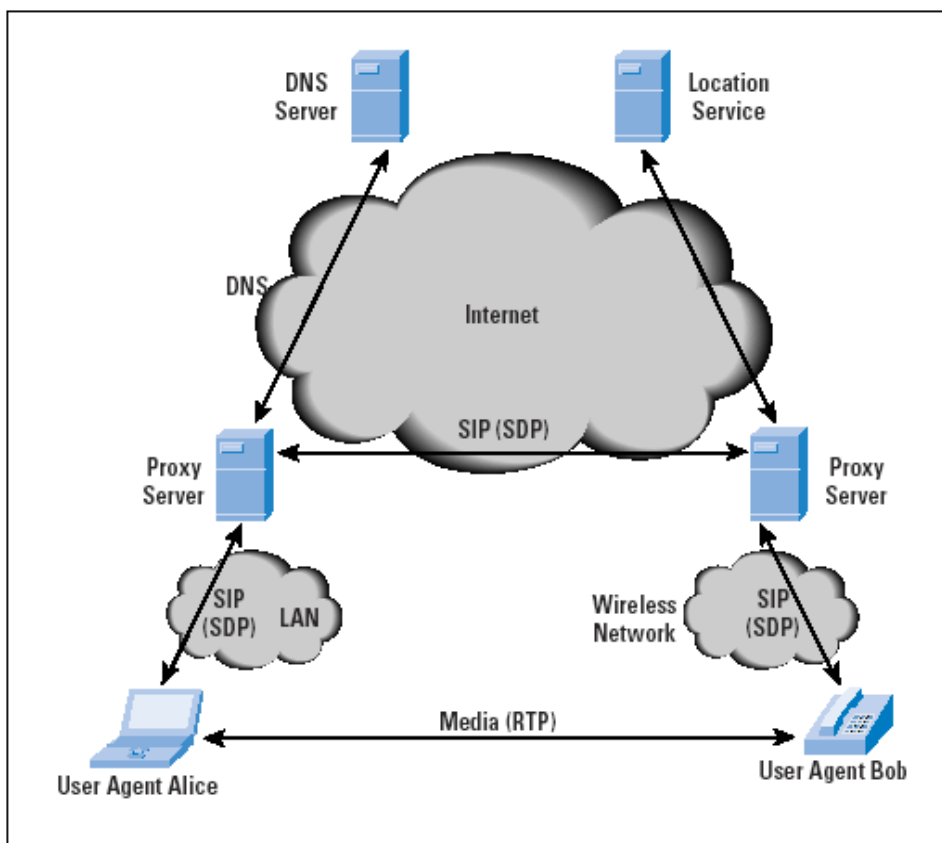
User Agent - Agent jest systemem końcowym, zawyż oprogramowaniem lub fizycznym urządzeniem działającym w imieniu użytkownika i jest elementem pośredniczącym w komunikacji między użytkownikiem a siecią. Agent składa się z dwóch części - klienta UAC (User Agent Client) i serwera UAS (User Agent Server). Dzięki temu, każdy terminal sieci SIP jest węzłem typu "host". Umożliwia to wysyłanie przez użytkownika

zadań protokołu SIP (np. inicjowanie sesji) i odbieranie takich ządań, przesyłanych do niego przez innych agentów oraz wysyłanie odpowiedzi w imieniu użytkownika (np. przyjęcie zaproszenia do sesji).

Serwer proxy - decyduje do którego serwera żądanie powinno być skierowane, po czym kieruje to żądanie. Żądanie może przemierzać poprzez wiele serwerów SIP przed osiągnięciem swego przeznaczenia. Odpowiedź przemierza drogę w odwrotnej kolejności.

Serwer przekierowań - w odróżnieniu od proxy nie przekierowuje ządań do innych serwerów, lecz powiadamia dzwoniącego o aktualnej lokalizacji miejsca przeznaczenia.

Serwer rejestrujący - prowadzi rejestrację User Agent klientów i ich bieżącą lokalizację. Serwery rejestrujące są często lokowane z serwerem proxy i przekierowań.



Rysunek 5. Przykładowa konfiguracja sieci SIP

4. Porównanie protokołów SIP i H.323

	H.323	SIP
Filozofia	<p>H.323 zostało zaprojektowane biorąc pod uwagę wszelkie wymagania usługą multimedialnym korzystającym z sieci IP, w tym przesyłanie audio/video, plików czy tworzenie konferencji. Determinuje cały integralny system do wykonywania tych czynności, wykorzystując mocne strony protokołów IETF i ITU-T.</p> <p>Użytkownicy H.323 mogą spodziewać się podobnej skalowalności i jakości co w sieciach PSTN</p>	<p>SIP został zaprojektowany w sposób elastyczny i modułowy do zestawiania sesji pomiędzy dwoma punktami w sieci. Posiada mniej restrykcyjną koncepcję komunikacji pomiędzy urządzeniami, co przenosi się na łatwość w implementacji.</p>
Złożoność	<p>H.323 ogranicza się do konferencji, więc jego złożoność jest ograniczona. Jednakże jest to skomplikowany protokół, który posiada ponad 736 specyfikacji. Dodatkowo niektóre funkcjonalności są duplikowane jak np. korzystanie z RTP i RTCP do otrzymywania informacji zwrotnej oraz sterowania konferencjami zamiast zaimplementowanego już H.245.</p>	<p>SIP początkowo skupiał się wyłącznie na przesyłaniu głosu. Wraz z jego rozwojem zaimplementowane zostały rozmowy video, udostępnianie aplikacji, komunikatory, czat itp. To doprowadziło do większej złożoności jednak SIP nie posiada tak sztywno przypisanych rozwiązań co H.323 co nadaje mu większą elastyczność.</p>
Niezawodność	<p>H.323 oferuje szereg funkcji do obsługi niepowodzeń przy pomocy infrastruktury sieciowej tj. „dodatkowe GK”, czy</p>	<p>SIP nie ma zdefiniowanych procedur zapobiegających awarią. Może to prowadzić do dłuższych opóźnień podczas wysyłania ponownych</p>

	„dodatkowe urządzenia końcowe”	wiadomości.
Specyfikacja wiadomości	ASN.1 (Abstract Syntax Notation One) restrykcyjnie standaryzuje strukturalną notację wiadomości	SIP korzysta z ABNF, (Augmented Backus-Naur Form), który jest zawarty w dokumentacji RFC 2234
Kodowanie wiadomości	H.323 koduje wiadomości do formatu binarnego, przez co urządzenia wydajniej pracują z takim typem danych.	Wiadomości SIP są kodowane poprzez tekstowy format ASCII. Kosztem czytelności dla ludzi spada wydajność podczas przesyłania takich wiadomości.
Tranposrt multimediiów	RTP/RTCP, SRTP	
Skalowalność - roszierzalność	H.323 jest rozwijany w sposób nie kolidujący z obecnymi właściwościami zachowując przy tam kompatybilność wsteczną.	SIP jest rozwijany w sposób nie kolidujący z obecnymi właściwościami nie zawsze zachowując przy tam kompatybilność wsteczną (RFC 3261 nie jest całkowicie kompatybilny z RFC 2543).
Skalowalność – określenie stanu	H.323 w wersji 1 oraz 2 korzystało wyłącznie z protokołu TCP. Obecnie możliwe jest włączenie również protokołu UDP.	W protokole SIP, transakcje między serwerami mogą być połączeniowe (TCP) lub bezpołączeniowe (UDP).
Skalowalność – identyfikacja użytkownika	GK może zwrócić żądany adres do miejsca inicjalizującego (model bezpośredniego połączenia) lub przetrasować wiadomość SETUP do urządzenia końcowego (model skokowy z GK)	SIP może korzystać z wielu protokołów do ustalenia adresu użytkownika (TRIP, ENUM, DNS). Urządzenia końcowe nie jest angażowane w tym procesie, UA wysyła żądanie INVITE do serwera proxy. W porównaniu do H.323, SIP wymaga co najmniej 3 wiadomości do ustalenia adresu co przenosi się na większą ilość przetwarzania danych.
Adresacja	H.323 wspiera wiele formatów i mechanizmów	SIP posiada uproszczony schemat adresowania URI, podobny do

	<p>do adresacji:</p> <ul style="list-style-type: none"> - E.164 nr telefonu - H.323 ID - URL - Adres transportowy - Adres mailowy - Numer konferencji - numer UIM - numer ISUP 	<p>adresu poczty elektronicznej, np.</p> <p>sip:22444032@phonesystem.3cx.com</p> <p>sip:joe.bloggs@212.123.1.213</p>
Łatwość adaptacji	Aby dostosować usługi, H.323 wymaga więcej interakcji pomiędzy protokołami lub nawet ich całkowitą zmianę.	SIP dzięki polom nagłówkowym i tekstowej naturze wiadomości jest dostosowanie parametrów jest relatywnie prostsza.
Uwierzytelnienie	Tak, przy pomocy H.325	Tak, przy pomocy HTTP, SSL, PGP, S/MIME, oraz wiele innych.
Topologie sieciowe	Unicast, multicast, gwiazda oraz scentralizowane	
Usługi	Oba protokoły posiadają bardzo zbliżone usługi i funkcjonalności. Jednakże szczegółowe porównanie SIP oraz H.323 jest trudne z powodu ilości usług, ich złożoności oraz subiektywnie ocenie jakości.	
Modularność	H.323 nie posiada tak przejrzystej separacji jak SIP. Głównie z przyczyny sprzężeń pomiędzy licznymi protokołami, które są ze sobą ściśle powiązane, aby zachować integralność.	Głównymi zadaniami SIP-a jest identyfikacja, rejestracja, lokalizacja użytkownika oraz sygnalizacja sesji. Dalsze usługi i funkcjonalności mogą zostać przekazane innym protokołom, bez ingerencji w sam protokół sygnalizacyjny.
Łatwość implementacji	Wiadomości sygnalizujące są kodowane binarnie używając ASN.1 PER (PACket Encodig Rules). Przez co informacje w pewien sposób muszą zostać dostosowane dla użytkowników. Dodatkowo do zmapowania abstrakcyjnej składni H.323 wymagany jest parser przez co implementacja i	Wiadomości SIP bazują na czystym tekście używając kodowania UTF-8. Dzięki takiemu podejściu implementacja w językach takich jak JAVA, Perl, Python jest znacznie prostsza.

	odpluskwanie są bardziej skomplikowane.	
Detekcja pętli	Serwer wykrywa pętle jeśli otrzyma żądanie z zamkniętej trasy (pole Via zawiera jego adres).	SIP posiada zaimplementowany algorytm detekcji pętli podobny do BGP (Border Gateway Protocol), który jest bardziej wydajny od uproszczonego zastosowanego w H.323
Rezerwacja zasobów	Nie wspierana przez żaden z protokołów. Oba zalecają z korzystania zewnętrznych rozwiązań takich jak DiffServ czy IntServ	
Zestawienie połączenia	<p>Poprzez UDP zajmuje średnio 1,5 pełnych okrążeń wiadomości.</p> <p>Setup -> <- Connect Ack -></p>	<p>Poprzez UDP zajmuje średnio 1,5 pełnych okrążeń wiadomości.</p> <p>INVITE -> <- 200 OK Ack -></p>
Współlistnienie z PSTN	W H.323 bramy są opcjonalnym komponentem architektury, kiedy jednak chcemy się połączyć z innym rodzajem sieci wtedy pomiędzy interfejsami wymagana jest brama. W bramie zachodzi translacja adresów, kodeków audio/video oraz żądanie połączenia czy jego zakończenie z obu stron interfejsu.	Bramy PSTN w architekturze SIP są dość popularnymi urządzeniami. Jednakże poprzez zaimplementowanie URI standardowe adresy telefoniczne użytkownik może doświadczyć problemy z taką siecią.

Tabela 1. Porównanie SIP i H.323

W świecie VoIP istnieją dwa przodujące rozwiązania protokołów sygnalizacyjnych H.323, który jest bardziej wydajny przepustowościowo oraz SIP, który jest prostszy i łatwiej skalowalny. Poniżej postaram się podsumować zawartość przedstawionego porównania.

W powyższej tabeli zaprezentowane zostało zestawienie protokołów SIP oraz H.323 z zakresu złożoności, rozszerzalności, skalowalności oraz usług. W kwestii funkcjonalności i

usług, które są dostępne H.323 oraz SIP cechują się podobnymi wskaźnikami. Jednakże dodatkowe usługi w H.323 są bardziej restrykcyjnie zdefiniowane. Dlatego też, przewidziane jest mniej problemów z kolejnymi wydaniem protokołu jak i również ze współistnieniem w takich sieciach jak PSTN. Oba protokoły są porównywalne we wsparciu QoS (podobne opóźnienia zestawiania połączenia, brak wsparcia dla rezerwacji zasobów). Głównymi zaletami SIP są: elastyczność w dodawaniu nowych funkcjonalności i ich łatwej implementacji oraz debugowania. Można dostrzec, że oba te protokoły stają się coraz bardziej zorganizowane, udoskonalając się od siebie nawzajem oraz, że różnice, kolejnych wersji zanikają między nimi.

Z praktycznego punktu widzenia, duża część rozwiązań na rynku jest dostarczana do większych firm, również korporacji, gdzie wiele rozwiązań jest specyficzna dla danego przedsiębiorstwa. Z tego powodu dystrybutorzy będą dostarczać kolejne wersje tychże protokołów, dopóki jeden ze standardów nie zostanie wyparty przez drugi, lub oba nie scalą się ze sobą. Jeśli celem jest stworzenie sieci ze wsparciem różnych technologii (np. IP z PSTN) o dużej wydajności wybór protokołu SIP może okazać się nietrafionym zwłaszcza jeśli porównamy go do H.323 w wyższych wersjach. Z drugiej strony SIP jest znacznie prostszym protokołem dzięki wielokrotnemu korzystaniu z pól nagłówka, kodowaniu, kodom błędów, oraz autoryzacji poprzez HTTP, prostota, która jest typowym podejściem dla sieci pakietowych. Kolejny fakt można podkreślić tym, że protokoły sygnalizacyjne zajmują się zestawianiem oraz rozłączaniem połączeń, gdzie odpowiedzialnym za samą transmisję głosu odpowiedzialny jest protokół RTP. W związku z tym wybór pomiędzy H.323 a SIP nie ma wpływu na jakość dźwięku.

Podsumowując z powyższego porównania wynika, że SIP zawiera zbliżone usługi co H.323, utrzymując przy tym dużo niższą złożoność, bogatszą rozszerzalność, głównie dzięki modularności oraz lepszą skalowalność. Dzięki tym faktom, w dzisiejszych czasach SIP jest częściej implementowany w sieciach VoIP. Dodatkowo, można to również zauważyć po ilości zapytań Google, który z protokołów jest bardziej popularny.



Rysunek 6. Trend wyszukiwań hasła SIP oraz H.323 w Google

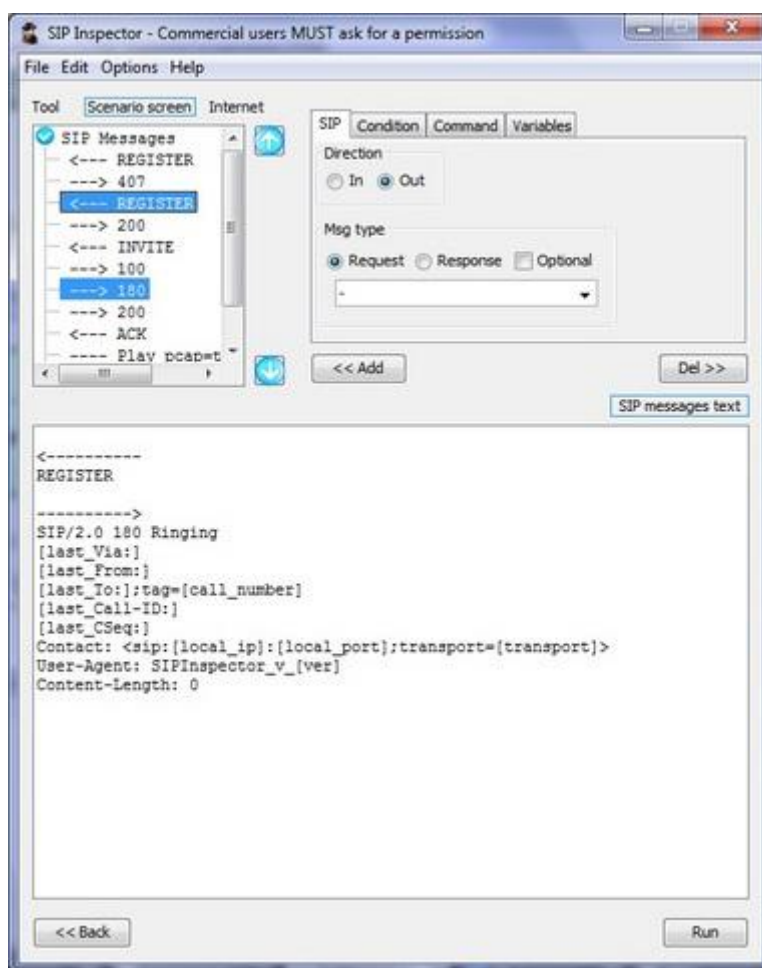
5. Aplikacje symulujące protokół SIP

5.1. SipInspector

Od niedawna stał się płatnym i dość rozbudowanym narzędziem. Jest jednym z popularniejszych tego typu oprogramowaniem, a przynajmniej był do czasu kiedy jego licencja była bezpłatna.

Zestaw głównych funkcji jaki zawiera SipInspector:

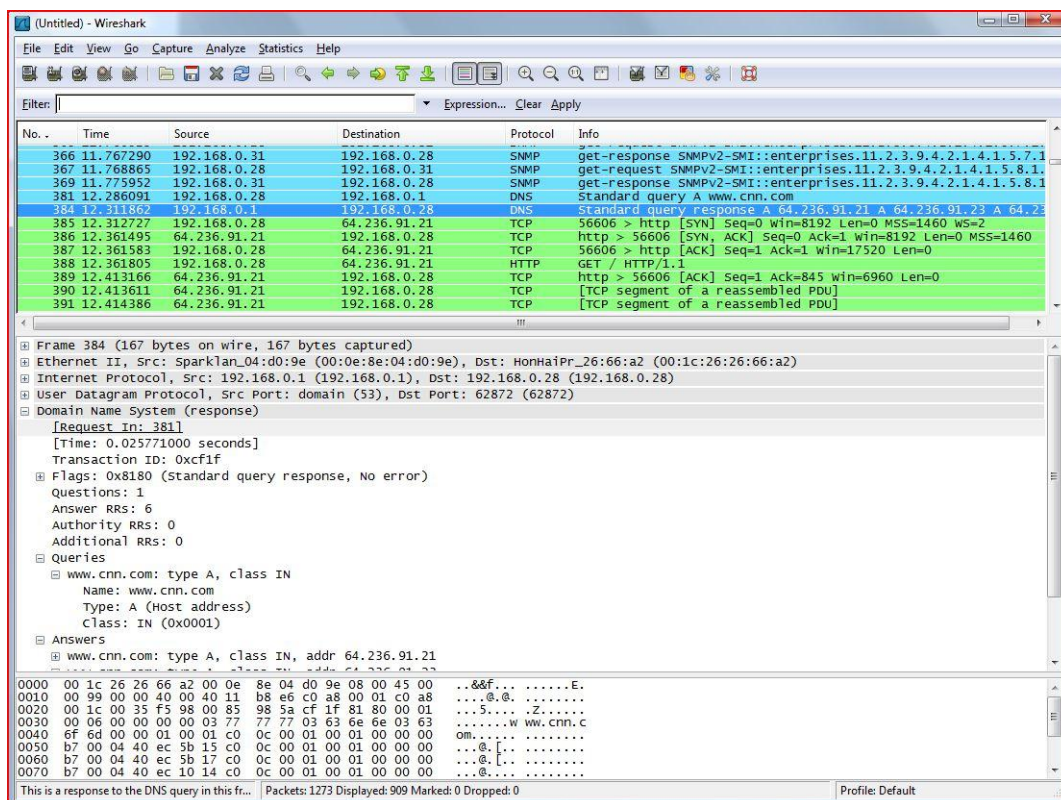
- tworzenie własnych wiadomości i scenariuszy,
- scenariusze warunkowe,
- przetwarzanie wielu scenariuszy jednocześnie,
- możliwość prowadzenie wielu dialogów na przestrzeni jednego scenariusza,
- testowanie pojemności serwerów rejestracyjnych oraz aplikacji,
- możliwość strumieniowania oraz odpowiedzi poprzez protokół RTP,
- kalkulator SDP (pomaga obliczyć pole body w bajtach),
- wyszukiwanie adresów i portów interfejsów z urządzeniami SIP,
- symulowanie serwera aplikacji



Rysunek 7. Interfejs graficzny programu SipInspector

5.2. Wireshark

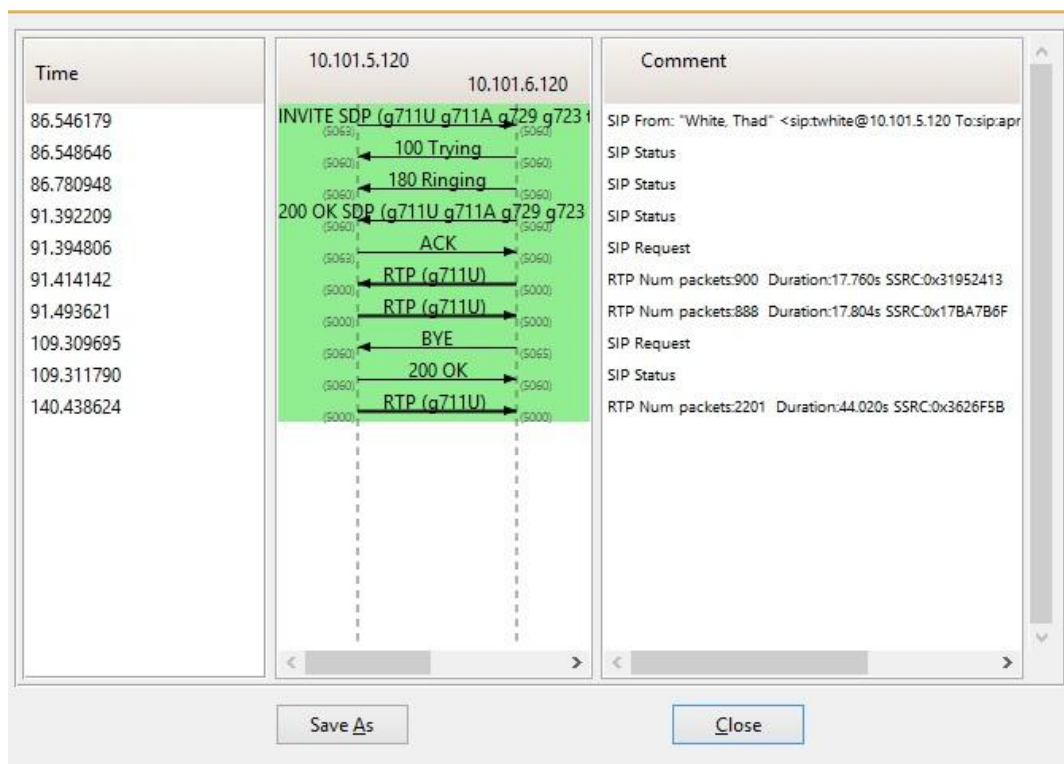
Wireshark jest najbardziej popularnym analizatorem sieciowym. To silne narzędzie dostarcza sieciowe i wyższych protokołów informacyjnych dotyczące zrzucanych danych w sieci. Jak inne programy sieciowe, Wireshark używa biblioteki pcap służącej do przechwytywania pakietów. [5] Jest najbardziej uniwersalnym oprogramowaniem z obecnych w tym rozdziale ponieważ potrafi przechwytywać niemal cały ruch sieciowy. Prezentacja danych jest w prosty i przejrzysty sposób przedstawiana w dwóch oknach głównego interfejsu aplikacji. Dokładnie jest to zobrazowane na Rysunek 8. Główne okno przechwytywania ruchu sieciowego programu Wireshark.



Rysunek 8. Główne okno przechwytywania ruchu sieciowego programu Wireshark

W górnym pasku narzędziowym można znaleźć takie funkcje jak start przechwytywania, zapis sesji, wybór interfejsu oraz dobranie filtra, dzięki któremu będziemy w stanie dostrzec tylko te fragmenty ruchu sieciowego, który nas interesuje. Poniżej przedstawione są przechwycone pakiety, które można sortować pod względem numeru pakiety, czasu przechwycenia, nadawcy, odbiorcy czy protokołu. W ostatniej sekcji wyświetlane są szczegółowe informacje o samym pakiecie, a na samym dole prezentowane są dane w formie heksadecymalnej jak i bardziej przystępnej dla człowieka.

Dla analizy protokołu SIP bardzo przydatną funkcjonalnością programu jest możliwość tworzenia grafów połączenia na podstawie przechwyconych pakietów, który można zobaczyć na Rysunek 9. Przykładowa sesja SIP przedstawiona w Wireshark.



Rysunek 9. Przykładowa sesja SIP przedstawiona w Wireshark

Głównymi zaletami Wireshark są:

- łatwość instalacji,
- przejrzysty interfejs,
- łatwości użytkowania za pomocą przyjaznego interfejsu dużej funkcjonalności i dostępności,
- darmowy,
- wieloplatformowy

5.3. SipP

SipP jest darmowym narzędziem oraz generatorem ruchu protokołu SIP, opartym na licencji Open Source. Obejmuje kilka podstawowych scenariuszy oraz potrafi inicjalizować i rozłączać połączenia przy pomocy metod INVIE, BYE. Potrafi wczytywać scenariusze użytkownika z plików XML. Oferuje dynamiczny interfejs statystyk uruchamianych testów (częstotliwość połączeń, opóźnienia trasy oraz statystyki wiadomości), zrzuty wyników do pliku CSV, korzystanie z protokołu TCP, UDP oraz multipleksacje z zarządzaniem retransmisji.

Inne zaawansowane funkcje obejmują obsługę:

- IPv6 ,
- TLS (Transport Layer Security) ,
- SCTP (Stream Controm Transmission Control),
- uwierzytelniania SIP ,
- scenariusze warunkowe ,
- retransmisji UDP,
- odporność na błędy (przekroczenie limitu czasu połączenia),
- POSIX,
- wyrażenia regularne do wydobywania i ponownego „wstrzyknięcia” pól protokołu,
- dodatkowe opcje dla otrzymanych wiadomości (logowanie, wykonanie komendy systemowej)
- wstrzykiwanie pól z zewnętrznego pliku CSV do symulacji użytkownika.

```

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

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

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

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

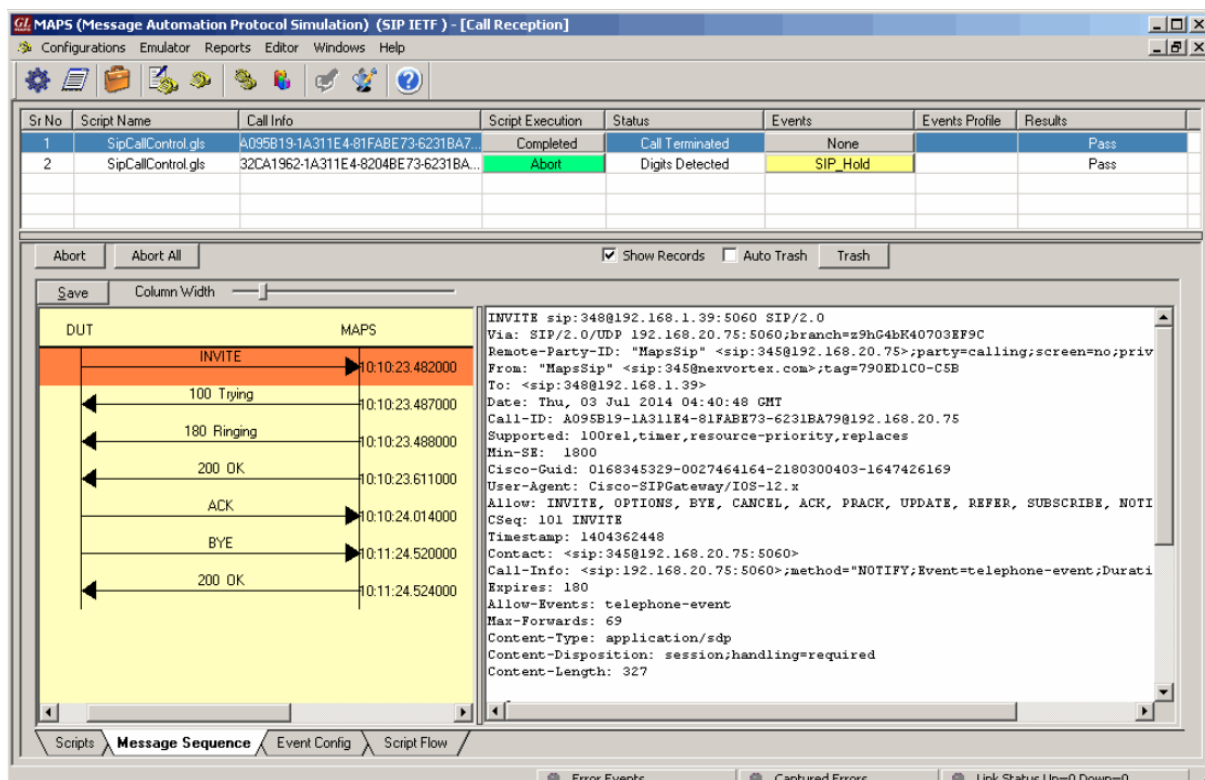
```

Rysunek 10. Interfejs symulatora SipP

5.4. MAPS SIP protocol emulator

MAPS (Message Automation & Protocol Simulation), najbardziej rozwinięty z dotychczas opisywanych symulatorów. Zaprojektowany do testowania infrastruktury SIP, potrafi zasymulować: User Agents (User Agent Client- UAC, User Agent Server-UAS), serwery proxy, przekserowań i rejestracji. Według autorów oprogramowanie jest zdolne do zasymulowania każdego interfejsu w sieci SIP oraz do przeprowadzenia testów zgodności protokołów. Posiada ponad 300 testowych scenariusze, które zawierają kompleksowy przepływ wiadomości, logowanie oraz generację raportów z wynikami testów. Daje użytkownikowi możliwość edycji wiadomości SIP oraz zarządzania scenariuszami.

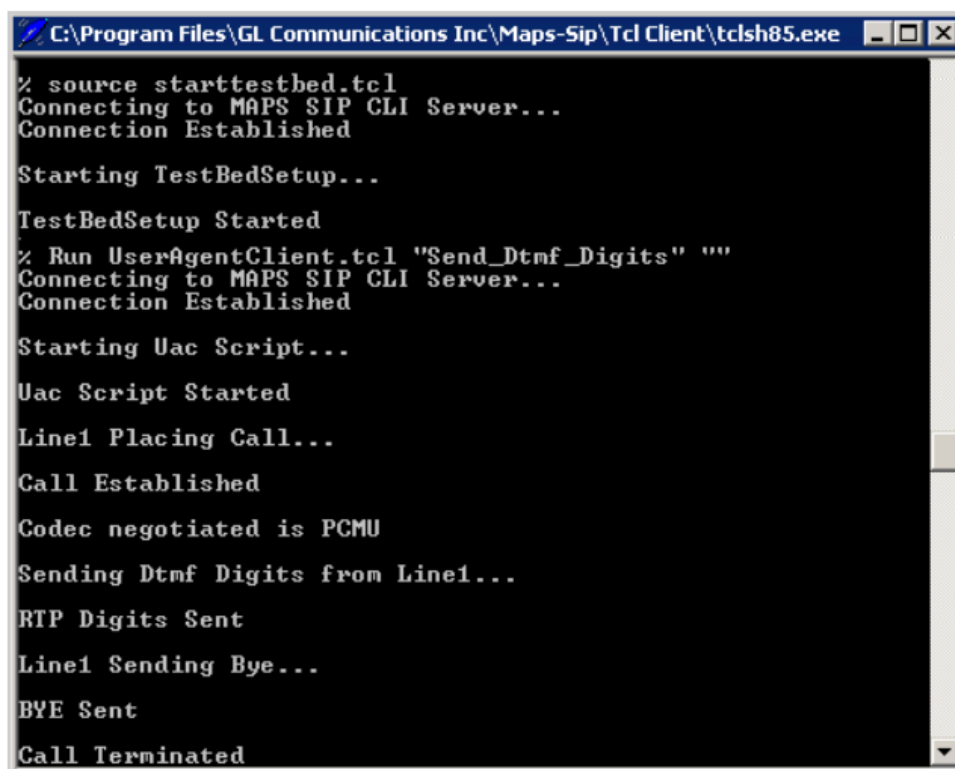
Sekwencje wiadomości (scenariusze) są generowane poprzez skrypty natomiast wiadomości są tworzone na podstawie wbudowanych schematów.



Rysunek 11. Przykładowy interfejs podczas tworzenia połączenia

Główne funkcje aplikacji MAPS:

- Generacje oraz przetwarzanie wiadomości SIP
- Pełne wsparcie personalizacji nagłówek, przesyłu wiadomości
- Wsparcie TCP i UDP (IPv4, IPv6)
- Retransmisja wiadomości z odpowiednim interwałem czasowym
- Skrypty do generacji i odbierania połączeń
- Wsparcie IP spoofing
- Wsparcie transmisji oraz detekcji ruchu RTP takie jak cyfry, pliki dźwiękowe, pojedyncze oraz podwójne tony w sieci IP
- Wsparcie prawie wszystkich kodeków G.711 (mu-Law and A-Law), G.722, G.729, G.726, GSM, AMR, EVRC, SMV, iLBC, SPEEX oraz więcej.
- Automatyczne, zdalne i zaplanowane testy 24/7
- Wsparcie dla Windows 7,8
- CLI (Command Line Interface)
- W pełni zintegrowane, kompletne środowisko testowe dla protokołu SIP



```
C:\Program Files\GL Communications Inc\Maps-Sip\Tcl Client\tclsh85.exe
% source starttestbed.tcl
Connecting to MAPS SIP CLI Server...
Connection Established

Starting TestBedSetup...
TestBedSetup Started
% Run UserAgentClient.tcl "Send_Dtmf_Digits" ""
Connecting to MAPS SIP CLI Server...
Connection Established

Starting Uac Script...
Uac Script Started
Line1 Placing Call...
Call Established
Codec negotiated is PCMU
Sending Dtmf Digits from Line1...
RTP Digits Sent
Line1 Sending Bye...
BYE Sent
Call Terminated
```

Rysunek 12. CLI (Comand Line interface) program MAPS

6. Autorska aplikacja

Przedstawione w poprzednim rozdziale aplikacje, są bardzo zróżnicowane, pod względem funkcjonalności, interfejsu, złożoności, czy koszcie użycia. Jednak do celów edukacyjnych ogrom ich możliwości ogranicza się do korzystania z podstawowych usług, a czasem jest przytłaczająca dla użytkownika. Najlepszym z kandydatów wydawałby się program SipInspector, ale od kiedy stał się płatnym oprogramowaniem jego popularność spadła, a sposobność jego używania nie zawsze jest możliwa z tego właśnie powodu. Wychodząc naprzeciw wymaganiom, powstał dedykowany program symulacyjny dla protokołu SIP. Ma on za zadanie w prosty i przejrzysty sposób przedstawić użytkownikowi zasady działania protokołu SIP oraz możliwość tworzenia swoich scenariuszy. Dodatkowym atutem jest również możliwość podłączenie urządzeń SIP i korzystanie z niego w fizycznych sieciach w celu wykonania prostych testów architektury sieciowej.

6.1. Opis interfejsu graficznego

Główne okno programu składa się z trzech zakładek:

- Input data
- Create/Load scenario
- Nazwa wczytanego scenariusza

6.1.1. Zakładka Input data

Input data | Create/Load Scenario | Session_Establishment_Through_Two_Proxies

Enter source address:
e.g. 192.168.100.1

Enter source port:
e.g. 5060

Proxy 1 address:
e.g. 192.168.100.3

Proxy 1 port:
e.g. 5060

Proxy 2 address:
e.g. 192.168.100.4

Proxy 2 port:
e.g. 5060

Enter destination address:
e.g. 192.168.100.2

Enter destination port:
e.g. 5060

If you don't need any of the hardware leave blank fields (IP and port!)

Saved parameters

Parameters seems to be valid!

Source Address 127.177.219.244:65522
Proxy one Address: 127.38.78.131:19657
Proxy two address: 127.109.70.204:10624
Destination address: 127.0.0.1:31291
Sender: kamszy
Receiver: revszy

☒ Simulation
Enable this to fill all fields automatically

Sender:
Enter sender name

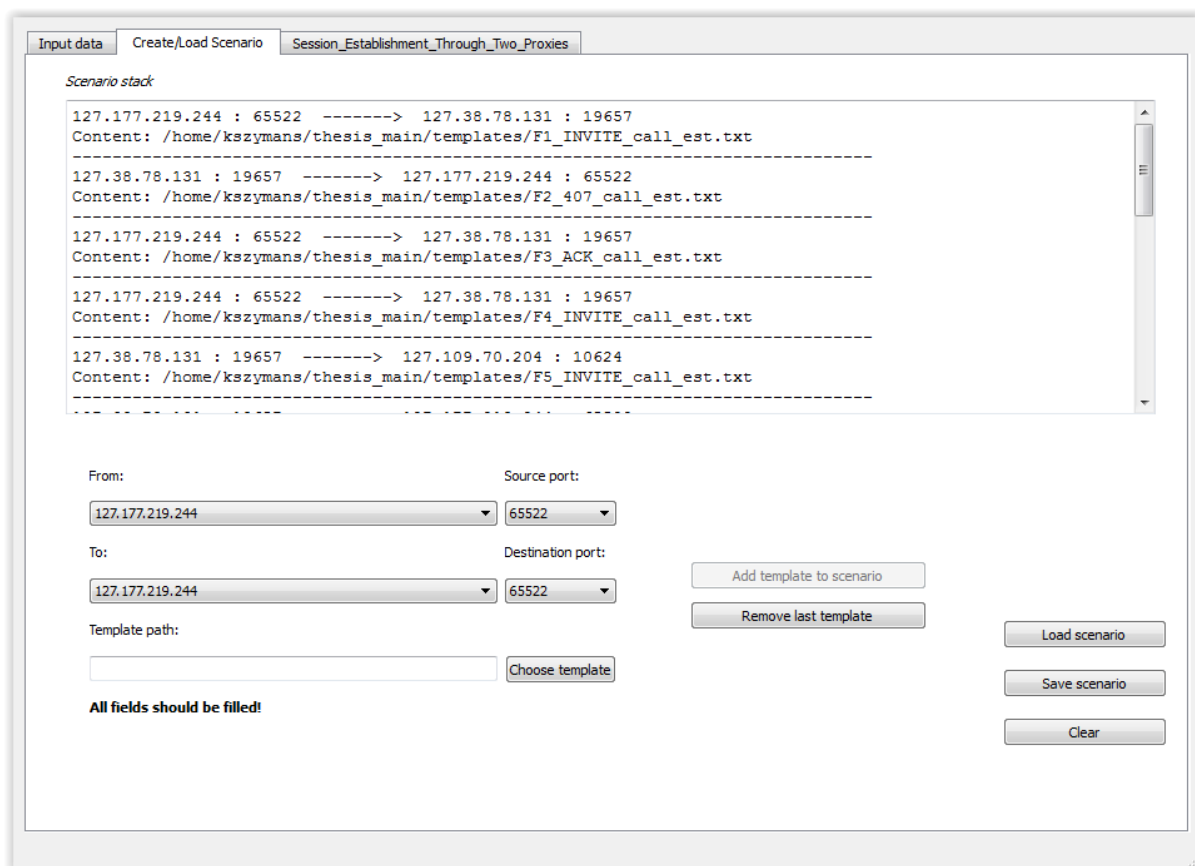
Receiver:
Enter receiver name

Clear Accept

Rysunek 13. Prezentacja zakładki Input

Zakładka Input data służy do wprowadzenia adresów, portów oraz nazwy nadawcy i odbiorcy. Te dane następnie zostaną przypisane odpowiednim urządzeniom, które będą brały udział w symulacji. Jest możliwe wyłączenie niektórych urządzeń z symulacji, w takim przypadku można zwyczajnie je pominąć podczas tworzenia scenariusza lub pozostawić puste pola. Należy jednak zwrócić uwagę, że adres IP, jak i port danego urządzenia powinny pozostać puste. Dodatkową opcją jest funkcja symulacji, dzięki tej właściwości wszystkie adresy zostaną przypisane do interfejsów loopback generując losowy adres IP. Porty natomiast zostaną również wybrane losowo z zakresu 1024 – 65535. W taki sposób nie trzeba się martwić o poprawność wprowadzonych danych, tracąc przy tym możliwość wysyłania wiadomości poza dany komputer.

6.1.2. Zakładka Create/Load Scenario



Rysunek 14. Prezentacja zakładki Create/Load scenario

W tej zakładce, jak sama nazwa wskazuje, można tworzyć lub wczytywać gotowe scenariusze. Przy pomocy wcześniej podanych danych w zakładce Input mamy możliwość utworzyć jedną z wiadomości SIP podając adres (symulowane urządzenie) nadawcy wraz z portem, docelowy adres wraz z portem oraz wiadomość. Treść wiadomości powinna być przechowywana w dowolnym, nieformatowanym pliku tekstowym, zachowując przy tym standardowe formatowanie wiadomości SIP. Przykładową wiadomość przedstawiono na Rysunek 15. Przykładowa wiadomość SIP dodawana do scenariusza.

Wiadomości są tworzone dynamicznie, oznacza to, że wszelkie pola o formacie %(zmienna)s będą zastępowane wartościami podanymi z zakładki Input data. zmienna może przybrać wartości:

- source_ip – adres IP nadawcy, odpowiednik pola Source address
- source_port – port nadawcy, odpowiednik pola Source port
- proxy_one_ip – adres IP pierwszego serwera proxy, odpowiednik pola Proxy 1 address
- proxy_one_port – port pierwszego serwera proxy, odpowiednik pola Proxy 1 port

- `proxy_two_ip` – adres IP drugiego serwera proxy, odpowiednik pola Proxy 2 address
- `proxy_two_port` – port drugiego serwera proxy, odpowiednik pola Proxy 2 port
- `dest_ip` – adres IP urządzenia docelowego, odpowiednik pola Destination address
- `dest_port` – port urządzenia końcowego, odpowiednik pola Destination port
- `sender` – nazwa nadawcy, odpowiednik pola Sender
- `receiver` – nazwa odbiorcy, odpowiednik pola Receiver

```
SIP/2.0 407 Proxy Authorization Required
Via: SIP/2.0/TCP %(source_ip)s:%(source_port)s;branch=z9hG4bK74b43;received=%(source_ip)s
From: %(sender)s <sip:%(sender)s@%(source_ip)s:%(source_port)s>;tag=9fxcde7651
To: %(receiver)s <sip:%(receiver)s@%(dest_ip)s:%(dest_port)s>;tag=3fla112sf
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Proxy-Authenticate: Digest realm="atlanta.example.com"
Content-Length: 0

<Description>
Proxy 1 challenges %(sender)s for authentication
</Description>
```

Rysunek 15. Przykładowa wiadomość SIP dodawana do scenariusza

Dodatkowo aby opisać daną wiadomość należy podać objaśnienie pomiędzy znacznikami `<Description>Opis</Description>`. Aby wczytać scenariusz należy kliknąć Load Scenario i wybrać plik, po czym w polu Scenario Stack, zostaną wyświetlone wiadomości zawarte w scenariuszu. Do programu zostały przygotowane scenariusze:

6.1.3. Zakładka symulacji

Rysunek

Po wczytaniu scenariusza można przystąpić do symulacji. Jak widać na załączonym architekturze sieciowa składa się z 4 urządzeń, urządzenia nadawczego, odbiorczego oraz dwóch serwerów. Z takiego zestawu można utworzyć $4^2 - 4 = 12$ konfiguracji. Po rozpoczęciu symulacji należy klikać w przycisk *Next* w celu wysłania kolejnych wiadomości. W polu *Connection flow* pojawiać się będą kolejno odebrane wiadomości. Ikony wraz z adresami obrazują architekturę sieciową, dzięki czemu użytkownik w przejrzysty sposób, może dostrzec zależność między protokołem sygnalizacyjnym a rolą urządzeń w sieci.

6.2. Zasada działania

Program został napisany w wysokopoziomowym języku Python, dzięki czemu kod jest przejrzysty, a implementacja kolejnych wersji będzie łatwa. Aplikacja jest wieloplatformowa, więc przy zapewnieniu odpowiednich bibliotek będzie działać pod systemem Windows jak i Linux. Do stworzenia symulatora wykorzystałem zmodyfikowaną wersję skryptu *sniff.py*, Jacka Szuberta, który nie zostanie omówiona w tej pracy.

Aplikacja została napisana w języku Python w wersji 2.7 z użytą biblioteką *PyQT* do obsługi interfejsu graficznego. Zastosowane zostały zalecenie PEP8, które to nadają kodu przejrzystości, dzięki poprawnym zasadą składni, tworzenia zmiennych czy stosowaniu komentarzy. Zostały utworzone własne wyjątki, zwiększając w ten sposób czytelność kodu podczas wystąpienia nieprzewidzianych błędów. Aplikacja została zabezpieczona przed wprowadzaniem niepoprawnych danych, lub zachowań użytkownika. Podczas próby niepoprawnego korzystania z oprogramowania wyświetlony zostanie odpowiedni komunikat, który powinien nakierować użytkownika do pożądanego celu. Dodatkowo wszelkie zdarzenia można zaobserwować w dolnej części okna głównego, na pasku statusu. Poniżej opiszę oraz przedstawię kod najważniejszych funkcji programu.

6.2.1. Zakładka Input data

Za wprowadzanie i obsługę danych użytkownika odpowiedzialny jest zestaw funkcji:

```
def getParameters(self):
    """This function captures input parameters and saves it to
    dictionary"""
def randomLoopback(self):
    """Return ip address 127.{}.{}.{} """
def loadParameters(self):
    """Separates ip addresses and ports onto two lists (source it
    from template_vars)."""
def checkIps(self):
    """Check any irregularity of IP addresses"""
def checkPorts(self):
    """Check any irregularity of ports"""
def checkSenderReceiver(self):
    """Check any irregularity of sender and receiver field"""
def prepareFieldsToSimulation(self):
    """Clears and adds ips and ports in proper fields"""
def printParameters(self):
    """Method to generates startup message with current
    parameters """
```

Funkcje *checkIps*, *checkPorts*, *checkSenderReceiver* odpowiadają za poprawność wprowadzonych danych przez użytkownika. Sprawdzają czy któryś z adresów IP nie jest zdublowany, czy para IP - port danego urządzenia są uzupełnione lub puste, czy format IP jest poprawny lub czy port jest z dopuszczalnego zakresu.

6.2.2. Zakładka Create/Load Scenario

Tworzenie, wczytywanie scenariuszy oraz dodawanie, usuwanie z nich wiadomości obsługiwane jest przez metody oraz klasy wymienione poniżej:

```
def addTemplateToScenario(self):
    """Adds template to Scenario """
def popDescriptionFromTemplate(self):
    """Pulls description out from template file and removes it
    return: str description, str template"""
def removeTemplateFromScenario(self):
    """Removes template from Scenario"""
def getTemplate(self, show_path=None):
    """Get template from file """
def createScenario(self):
    """Create scenario"""
def loadScenario(self):
    """Load scenario from file and execute some checks"""
def checkTemplates(self):
    """During loading scenario checks parameters if not
    missing"""
def formattedScenario(self):
    """Generates scenario stack"""
class Scenario(IO):
    """Wrapper class of IO"""
class IO(QtGui.QFileDialog):
    """Class of I/O processes in graphic mode"""
```

Scenariusze są to listy słowników, które posiadają poniższą strukturę:

```
(
    {"source_ip": str(self.sourceAddrBox.currentText()), \
     "source_port": str(self.sourcePortBox.currentText()), \
     "dest_ip": str(self.destAddressBox.currentText()), \
     "dest_port": str(self.destPortBox.currentText()), \
     "path": self.template_path, \
     "template": template, \
     "description": description}
)
```

Wszystkie klucze w słowniku są obowiązkowe, jeśli którego zabraknie zostanie wyświetlony komunikat o błędzie. Scenariusze przechowywane są w pliku generowanym przy pomocy biblioteki *pickle*, która potrafi przekształcić każdy obiekt Python-a w napis (nazywa się to marynowaniem), a następnie zapisać go do pliku. Dzięki tej właściwości w łatwy sposób można przysyłać i dzielić się utworzonymi scenariuszami z innymi użytkownikami. Tę zachowanie nadzoruje klasa *IO*, która, przy pomocy biblioteki *PyQT* pozwala tę czynność wykonać w sposób graficzny. Klasa *Scenario* jest swoistą klasą opakującą klasy *IO*, konsekwencją czego można w klarowny sposób utworzyć lub wczytać scenariusz, na przykład za pomocą:

```
scenario = Scenario(self.scenario) #utworzenie obiektu
filename = scenario.create() #zapisanie scenariusza do pliku
```


Warto zwrócić również uwagę na zastosowanie funkcji `checkTemplates` oraz `loadParameters`. Podczas wczytywania scenariusza, sprawdza czy wszystkie obowiązujące nagłówki są dostępne, tj.: *To*, *From*, *Cseq*, *Call-ID* oraz *Via* (`checkTemplates`). Dodatkowo zastosowane są te same zabezpieczenia co w zakładce *Input Data* (`checkIps`, `checkPorts`, `checkSenderReceiver`), w celu podwyższenia niezawodności (`loadParameters`).

6.2.3. Zakładka symulacji

Symulacja jest najbardziej rozbudowaną częścią aplikacji, która korzysta z modułu *Sniff.py* służący do przechwytywania wysłanych wiadomości. Z tej przyczyny pominę analizę tego skryptu jako, że jego opis można znaleźć w pracy magisterskiej z pkt. 6 bibliografii.

Zestaw funkcji, klas oraz bibliotek, które są używane do symulacji:

```
def loadTemplate(self):
    """Renders given template file, insert rendered
    description and message to scenario"""
def next(self):
    """Send another message and prepare fields"""
def generateRequest(self, case):
    """Generates request message"""
def send(self, case):
    """Sends generated message to given address"""
def open_sock(self, ip, port):
    """Prepares sockets"""
def reset(self):
    """Stops the simulation"""
def startSniff(self):
    """Starts sniffer"""
def printGraph(self, source_ip, destination_ip, sip_method):
    """Return formatted SIP graph flow and message"""
def turnOnOff(self, label, off):
    """Indicator of simulation status"""
class Sniff(threading.Thread):
    """Sniffing tools for capturing messages"""
class Message:
    """SIP Protocol headers + body."""
class Request(Message):
    """SIP request."""
class Response(Message):
    """SIP response."""
```

Metoda `loadTemplates` służy do renderowania wiadomości SIP, to w niej podmieniane są wartości zapisane w formacie `%(zmienna)s`. Fragment kodu odpowiadający za tą funkcjonalność:

```
for case in self.scenario:
```

```

        try:
            case["description"] = case["description"] %
self.template_vars
            case["message"] = case["template"] %
self.template_vars #replaces %(<variable>)s by template_vars
        except KeyError, e:
            PopupDialog("Some of the paramters are missing in
template: {}".format(e), "Whopsie..", "warning")

```

Do wysyłania oraz odbierania pakietów zaimportowana została biblioteka *socket* (*gniazdo*), który zawiera wszystkie niezbędne definicje i funkcje pozwalające na otwieranie gniazd i komunikację sieciową z ich wykorzystaniem. Ta właściwość została zaimplementowana w funkcji *openSocket* i jest używana podczas wysyłania wiadomości SIP przy pomocy protokołu UDP.

Klasa *Sniff* korzysta z biblioteki *threading*, która tworzy osobny wątek w celu przechwytywania wiadomości, które to następnie są wypisywane w polach: *Connnection flow*, *Current Message* oraz *Description*.

Funkcja *next* jest wywoływana podczas aktywowania przycisku *Next*. Przygotowuje pola interfejsu, generuje wiadomość scenariusza, a następnie wysyła pod odpowiednie adres i port.

Funkcja *reset* zatrzymuje symulację, składa się na to kilka akcji, a najistotniejsze z nich to:

- Wyczyszczenie licznika wiadomości
- Usunięcie obiektu klasy *Transmission* z modułu *Sniff.py*
- Zatrzymanie wątków
- Wysłanie wiadomości kończącej połączenie

```

self.message_counter = -1
try:
    killTransmission()
except NameError:
    return
self.sniff_thread.stop()
self.sniff_thread = None
self.send({'message': 'END TRANSMISSION', 'source_ip':
'127.0.0.1', 'source_port': '6666', \
'dest_port': '6666', 'dest_ip': '127.0.0.1'})

```

Metoda *printGraph* jest zaimplementowana w module *Sniff.py* została dostosowana na potrzeby symulatora w sposób, który można zobaczyć na . Funkcjonalność ta została wydzielona do odrębnej funkcji z powodu rozszerzenia jej działalności i zachowaniu modularności. Rozpoznaje adres źródłowy, docelowy pakietu, rodzaj wiadomości oraz jego numer, dzięki czemu jest w stanie utworzyć odpowiednie odwzorowanie w postaci tekstowej.

Klasy `Message`, `Request` oraz `Response` służą wyłącznie do sprawdzenia poprawności czy dana wiadomość SIP jest żądaniem, czy odpowiedzią oraz czy zostało poprawnie wygenerowane. Zastosowane są w nich wyrażenia regularne, które w połączeniu ze specyfikacją wiadomości SIP pozwalają na tego typu analizę.

6.2.4. Pozostałe

W tej sekcji znajdują się funkcje i klasy, które nie zakwalifikowały się do powyższego podziału:

```
def closeEvent(self, event):
    """This function is called due to press "x"
    Popup widnow is showed up and scenario simulation is
    stoped"""
def enableAddCaseButton(self):
def simulateRadioClicked(self, enabled):
```

Dodatkowe funkcje sprawdzające poprawność wiadomości SIP:

```
def canon_header(s):
def parse_headers(f):
    """Return dict of HTTP headers parsed from a file
    object."""
def parse_body(f, headers):
    """Return SIP body parsed from a file object, given HTTP
    header dict."""
```

Klasy potomne biblioteki *PyQt*:

```
class PopupDialog(QtGui.QMessageBox):
    """This class triggers new popup window"""
class Ui_MainWindow(QtGui.QMainWindow):
    """PyQT class"""
```

Wyjątki:

```
class WrongIp(Exception): pass
class WrongPort(Exception): pass
class WrongSenderReceiver(Exception): pass
class SipError(Exception): pass
class SipUnpackError(SipError): pass
class SipNeedData(SipUnpackError): pass
class SipPackError(SipError): pass
```

Funkcje `closeEvent`, `enableAddCaseButton`, `simulateRadioClicked` odpowiadają za działania podjęte w interfejsie graficznym, a ich nazwy jednocześnie określają ich funkcjonalność. `canon_header`, `parse_headers`, `parse_body` są zestawem funkcji z których korzystają klasy `Response`, `Request` oraz `Message`. Przetwarzają oraz sprawdzają poprawność wiadomości SIP.

Klasa `Ui_MainWindow` jest wygenerowana automatycznie poprzez bibliotekę *PyQT*, która zawiera szereg parametrów odpowiedzialnych za wygląd GUI (Graphical User Interface)

Ostatni zestaw klas jest grupą wyjątków i został stworzony dla zwiększenia czytelności kodu.

6.3. Porównanie symulatorów

Poniższa tabela przedstawia zestaw wymogów postawionych edukacyjnemu symulatorowi protokołu SIP.

	SipInspector	Wireshark	SipP	MAPS	Aplikacja autorska
Rodzaj licencji	Płatna	Darmowa	Darmowa	Płatna	Darmowa
Wspierane protokoły	TCP, UDP ,SIP, IPv4, RTP	SIP, H.323, UDP, TCP, IPv4/6, RTP, TLS, SCTP i dużo więcej	TCP, UDP ,SIP, IPv4, TLS, RTP, SCTP	Wyspecyfikowane w RFC 3261, 3262, 3515 w tym TCP, UDP ,SIP, IPv4, RTP	UDP ,SIP, IPv4
Tworzenie scenariuszy	Tak	Nie	Tak	Tak	Tak
Wsparcie fizycznej architektury sieciowej	Tak	Tylko przechwytywanie pakietów	Tak	Tak	Tak
GUI	Tak	Tak	Nie	Tak	Tak
Wspierane OS	Windows/Unix/OS X	Windows/Unix/OS X	Unix/Windows XP	Windows 7/8	Windows/Unix
Możliwość rozbudowy oprogramowania	Nie	Nie	Tak	Tak, za dopłatą	Tak
Poziom zaawansowania aplikacji	Umiarkowany	Umiarkowany	Umiarkowany	Zawansowany	Podstawowy
Technologia tworzenia aplikacji	Java	C/C++	C++	-	Python

Tabela 2. Porównanie symulatorów SIP

Zestawienie powyższych programów zostało dobrane pod kątem dostępności, funkcjonalności oraz możliwością zastosowania ich do celów edukacyjnych. Jednakże, każda z opisanych aplikacji została dostosowana do innych wymagań, konsekwencją czego, ich parametry są bardzo zróżnicowane. Wybór kryteriów również nie był przypadkowy, są to

wymagania, które według, subiektywnej oceny, autora powinien spełniać dedykowany symulator SIP do celów dydaktycznych.

Istotne jest żeby aplikacja była możliwie prosta i intuicyjna, aby użytkownik, po szybkim zapoznaniu z oprogramowaniem, mógł skupić się wyłącznie na przedstawionej treści. W tej kategorii najlepiej wypada autorska aplikacja ponieważ została skonstruowana pod dedykowane środowisko użytkowników. Interfejs jest przejrzysty bez zbędnych opcji czy konfiguracji, a użytkownik jest w stanie z niej korzystać praktycznie natychmiast. SipInspector równie dobrze się sprawdza w tej kwestii, natomiast SipP poprzez brak interfejsu graficznego jest nieco trudniejszy w obsłudze. Najgorzej wypadły MAPS oraz Wireshark, które pod nadmiarem ustawień tracą na przejrzystości. Jednakże nadrabiają to możliwościami, z których można wyróżnić np. testy infrastruktury sieciowej, ilość wspieranych protokołów czy prowadzenie statystyk.

Korzystną cechą dobrego oprogramowania jest jego możliwość oraz łatwość rozszerzalności, która przejawia się odpowiednią klarownością oraz modularnością kodu czy rodzajem zastosowanej licencji. Pod tym względem najwyższe miejsce zajęły SipP oraz aplikacja autora, a to dzięki licencji Open Source, która pozwala na dowolną modyfikację oraz dostęp do kodu źródłowego. Są to jedyne aplikacje które udostępniają kod źródłowy z całego zestawienia, a dodatkowym atutem autorskiej aplikacji jest technologia w jakim została stworzona. Jest nią wysokopoziomowy język Python, który posiada bardzo intuicyjną składnię oraz wymusza zachowanie przejrzystości kodu. Struktura kodu dodatkowo jest obiektowa i została oparta o zalecenia PEP8, co w jeszcze wyższym stopniu przekłada się na czytelność i łatwość dalszego rozwijania. W tej kategorii słabiej wypada aplikacja MAPS, dla której dodatkowe funkcje są ograniczone i niestety odpłatne. Natomiast o programach SipInspector oraz Wireshark można powiedzieć, że praktycznie nie jest możliwe poszerzenie funkcjonalności.

Jeśli mówimy o oprogramowaniu edukacyjnym czyli stosowanym głównie na uczelniach to kolejnym ważnym aspektem jest jego cena, która w pewnych okolicznościach może być wymaganiem koniecznym. Po raz kolejny w tej kwestii najlepiej wypadają SipP, autorska aplikacja oraz Wireshark, są one w pełni darmowe. Niestety, SipInspector stał się od niedawna oprogramowaniem płatnym jednak w porównaniu do MAPS koszt jedynie 40\$.

Podsumowując, według subiektywnej oceny autora, najwięcej z powyższych kryteriów spełnia aplikacja autorska, która podczas tworzenia opierała się na tych wymaganiach. Wynikiem czego powstało dedykowane, intuicyjne oprogramowanie edukacyjne dla konkretnego protokołu sygnalizacyjnego. Równie dobrze wypada SipInspector, jednak płatna licencja skreśla go z listy dostępnych symulatorów. Z kolei największą wadą SipP jest brak interfejsu graficznego, przez co nie jest tak przejrzysty jak aplikacja autora. Na czwartym miejscu uplasował się Wireshark z dość przyzwoitym interfejsem graficznym, który nie jest dedykowaną aplikacją dydaktyczną, mimo to jest

korzystany na wielu uczelniach w celu analizy przesyłanych protokołów. Na samym końcu znalazł się symulator MAPS, a to ze względu na jego cenę, która przekracza możliwości większości uczelni. Dodatkowo oprogramowanie jest bardzo rozbudowane i ściśle nastawione do symulowania całej infrastruktury sieciowej przed jej fizyczną implementacją. Na skutek tego, ilość opcji konfiguracji może przytłoczyć użytkownika.

7. Podsumowanie

Celem pracy było przedstawienie i analiza porównawcza dwóch najbardziej popularnych protokołów sygnalizacyjnych SIP oraz H.323. Czego efektem jest *Tabela 1. Porównanie SIP i H.323*, w której zostały zawarte najważniejsze cechy obu technologii. Na podstawie wspomnianego zestawienia, zebranej literatury oraz wyników zapytań Google można wyciągnąć konkluzje, że oba te protokoły są bardzo do siebie podobne, ostatecznie obejmują tę samą funkcjonalność, jednak po głębszej analizie dostrzega się niuanse, które znacznie wpływają na działanie danej sieci. Według autora SIP jest znacznie opłacalną technologią ponieważ, dzięki swej skalowalności i prostocie, można w względnie łatwy i szybki sposób zaimplementować go dla konkretnej architektury sieciowej, co również wpływa na obniżenie kosztów. Wyższe wersje protokołu H.323 posiadają już poprawki, dzięki którym zestaw owych zaleceń jest na tyle sprawny, że dorównuje protokołowi SIP. Przy odpowiedniej konfiguracji może być wydajniejszy co jednak w dalszym ciągu przekłada się na większą złożoność i w razie awarii dłuższe, a co za tym idzie kosztowniejsze rozwiązywanie awarii.

Zostało opracowane również zestawienie symulatorów protokołu SIP wliczając w to stworzoną przez autora aplikację. Napisana w wysokopoziomym języku Python oraz dostosowana do kryteriów spełniających stanowisko laboratoryjne, ma za zadanie, w sposób szybki i klarowny, zobrazować użytkownikowi zasadę działania protokołu SIP. W *Tabela 2. Porównanie symulatorów SIP* zostały zebrane najważniejsze cechy dobrego oprogramowania dydaktycznego. Wymagania te opracowane były w sposób subiektywny tak samo jak opis wraz z oceną symulatorów. Jednocześnie autor starał się dobrać aplikacje w sposób obiektywny, tak aby od początku nie można było określić faworyta. W rezultacie aplikacja autora została określona mianem najlepszej, a to za sprawą tworzenia jej od podstaw biorąc pod uwagę kryteria z *Tabela 2. Porównanie symulatorów SIP*. Tak dedykowane oprogramowanie powinno spełniać wszelkie oczekiwania użytkownika.

8. Bibliografia

1. <http://pinkaccordions.homelinux.org/staff/tp/prog/tex/wzmgr/tf-tex-1.pdf>
2. <http://szmarcin.w.interia.pl/text/h323sip.html>
3. <http://www.cs.wustl.edu/~jain/cis788-99/ftp/h323/index.html>
4. Dokumentacja RFC 3665 „*Session Initiation Protocol (SIP) Basic Call Flow Examples*”, <https://tools.ietf.org/html/rfc3665>
5. <http://pl.wikipedia.org/wiki/Wireshark>
6. Mgr. Jacek Szubert, „*Koncepcja stanowiska laboratoryjnego do realizacji i analizy usług multimedialnych*”, Politechnika Wrocławska, 2014
7. http://www.analog.com/library/analogDialogue/archives/40-04/blackfin_voip.html

9. Instrukcja laboratoryjna oraz obsługi aplikacji

9.1. Instrukcja laboratoryjna

9.2. Instrukcja obsługi

