Politechnika Wrocławska
Wydział Elektroniki

Kierunek: Telekomunikacja
Specjalność: Telekomunikacyjne sieci mobilne

Praca dyplomowa
Magisterska

Koncepcja stanowiska laboratoryjnego
do realizacji i analizy usług multimedialnych
A concept of test bench for implementation
and analysis multimedia services

Autor:
Jacek Szubert

Prowadzący pracę:

dr inż. Tomasz Długosz, Katedra Telekomunikacji i Teleinformatyki

Ocena pracy:

WROCŁAW 2014

Spis treści

1. Wprowadzenie

Wpływ infrastruktury na kulturę masową oraz generalny rozwój społeczeństwa jest ogromy, co nieraz już potwierdzały przełomowe dla ludzkości wynalazki czy ich wdrożenia. Począwszy od sieci komunikacyjnej łączącej wszystkie ważniejsze prowincje Cesarstwa Rzymskiego, przez odkrycie elektryczności i powszechną elektryfikację kilkanaście stuleci później, po odkrycie telefonu. Obecny rozwój dyktują technologie pakietowe, z Internetem na czele.

Dzisiejszy rozwój sieci zmierza w kierunku wyeliminowania podziału na sieci dające możliwość świadczenia usług głosowych i transmisję danych — dochodzi do konwergencji różnego typu usług. Przenikanie się owych za sprawą protokołu IP już dziś jest codziennością — transmisja dźwięku czy obrazu w sieci Internet jest ogromnie popularne i ciągle obserwujemy spadek systemów dających możliwość transmisji jednego rodzaju mediów na rzecz systemów konwergentnych.

Niniejsza praca skupia się na usługach typu VoIP oraz pokrewnych, które stanowią znakomity przykład konwergencji usług, a także zdobywają coraz większą popularność, wypierając sieci konwencjonalnej telefonii. Zaprezentowana zostanie aplikacja do analizowania ruchu sieciowego generowanego przez połączenia wykorzystujące najpopularniejsze protokoły, a także koncepcja stanowiska laboratoryjnego umożliwiającego analizę owych.

2. Usługi multimedialne

Zgodnie z definicją proponowaną przez Sektor Normalizacji Telekomunikacji ITU[11], jako medium określić możemy sposób w jaki informacja jest wyrażana, przechowywana lub transmitowana. Spośród form reprezentacji mediów wspomnieć warto o tekście, grafice, dźwięku oraz obrazie ruchomym. ITU-T jako usługę multimedialną rozumie możliwość transmisji co najmniej dwóch z wyżej wymienionych, w sposób z punktu widzenia użytkownika końcowego, odpowiednio zsynchronizowany. Analogicznie zdefiniowana jest aplikacja multimedialna — wymagająca dwóch lub więcej rodzajów mediów jednocześnie.

Multimedia znajdują szerokie zastosowanie we wielu sektorach gospodarki, nauki czy rozrywki. W tym ostatnim temacie znakomitym przykładem interaktywnych usług multimedialnych mogą być gry, łączące w sobie przekaz obrazu, dźwięku i tekstu. Multimedia jednak, równie często jak w mniej ważnych dziedzinach, takich jak wspomniane wcześniej gry czy też reklamy, wykorzystywane są w kluczowych dla społeczeństwa sektorach. Już teraz ilość dostępnych materiałów edukacyjnych, szczególnie pochodzących z zagranicznych uczelni, jest przytłaczająca. Prezentacje łączące w sobie kilka rodzajów mediów, często dodające do procesu nauki interakcję zarówno szkolonego jak i szkolącego, stają się codziennością i przyszłością. W inżynierii natomiast najprostszym przykładem będą aplikacje typu CAD (ang. *Computer Aided Design*), CAE (ang. *Computer Aided Engineering*) czy inne, stosowane także w naukach matematycznych, chemicznych czy biologicznych, do tworzenie zaawansowanych symulacji.

Usługami na których skupiono się w owej pracy jednak będą te, dające możliwość użytkownikowi końcowemu transmisji obrazu (również ruchomego), dźwięku czy tekstu, do wybranych przez niego odbiorców. Funkcjonalności takie dają współczesne telefony komórkowe, ogrom aplikacji internetowych, czy też fizycznych urządzeń. Technologia określana jako VoIP (ang. *Voice over Internet Protocol*), pozwala na integrację wspomnianych, z istniejącą już siecią globalną, lub też sieciami lokalnymi. Wykorzystanie protokołu IP, oprócz wspomnianej integracji, daje znaczne profity, począwszy od eliminacji efektu zajętości łącza podczas gdy rozmówcy milczą, poprzez ogrom usług dodatkowych, niedostępnych w konwencjonalnych sieciach,

po koszty wdrożenia, które przy implementacji systemowej i zastosowaniu wolnego oprogramowania, mogą być potężnie zminimalizowane.

2.1. MMS

MMS (ang. *Multimedia Messaging Service*) jest rozszerzeniem funkcji SMS (ang. *Short Messaging Service*) i EMS (ang. *Enhanced Message Service*) i zarazem idealnym przykładem usługi multimedialnej. Daje możliwości przesyłania tekstu, grafiki, animacji, dźwięku, a nawet aplikacji czy innych, dowolnych plików z danymi. Przez możliwość podziału na "sceny", daje nawet możliwość tworzenia swoistych prezentacji.

2.2. Streaming

W związku z rosnąca przepustowością połączeń, standardowy sposób pobierania plików multimedialnych, takich jak muzyka czy film, odchodzi do lamusa. Pobieranie i możliwość ich oglądania dopiero po ukończeniu transferu, zastępowane zostaje buforowaniem jedynie części pliku, która wkrótce ma zostać odtworzona. Bezkonkurencyjne niegdyś sieci typu P2P (ang. *Peer-to-peer*), takie jak Napster, Audiogalaxy, Gnutella, DirectConnect, eDonkey, czy nawet powoli BitTorrent, wypierane zostają serwisami streamingowymi. Również portale oferujące kiedyś wyłącznie hosting plików, aby utrzymać się na rynku, wprowadzają ich odtwarzanie w czasie rzeczywistym.

2.2.1. Historia

Strumieniowy przekaz do niedawna ograniczony był kilkoma czynnikami. Serwery generujące tego typu dane posiadać muszą ogromną moc obliczeniową, co nie jest wymagane przy standardowym przekazie plików. Kolejnym problemem była przepustowość sieci — aby na bieżąco zbuforować odpowiednią ilość danych, musi być ona odpowiednio duża, co znowu nie jest wymagane przy konwencjonalnym sposobie transferu, gdzie nawet bardzo wysoki czas oczekiwania nie wpływa bezpośrednio na sposób odbioru samego medium. Pierwsze próby strumieniowania danych na większa skale podjęte zostały około roku 1993, natomiast w roku 1995 oraz 1999 kolejno firmy Microsoft oraz Apple wydały gotowe rozwiązania, które posłużyć mogłyby popularyzacji idei — ActiveMovie (obecnie DirectShow) oraz QuickTime. Około roku 2002

wzrosło zainteresowanie wykorzystaniem Adobe Flash, który dominuje na rynku do dziś.

2.2.2. Kodeki i protokoły

Dane dźwiękowe kompresowane są na ogół przy pomocy MP3, Vorbis czy AAC, do wideo natomiast stosuje się H.264 oraz VP8. Strumień transmitowany jest protokołem MMS (ang. *Microsoft Media Server*) oraz RTP (ang. *Real-time Transport Protocol*). W niższej warstwie najbardziej efektywnym protokołem jest UDP (ang. *User Datagram Protocol*), który w razie problemów z transportem porzuca próby ponowienia, nie blokując tym samym kolejnych datagramów. Za zapewnienie QoS odpowiadać w takim przypadku musi inny protokół, taki jak RTCP (ang. *Real Time Control Protocol*). Coraz popularniejszym rozwiązaniem są jednak technologie oparte na HTTP (ang. *Hypertext Transfer Protocol*). QoS zapewniany jest poprzez kontrolę przepustowości łącza i dostosowywanie do niej jakości transmitowanego medium. Obecnie wykorzystywane są takie techniki, jak MPEG-DASH (ang. *Dynamic Adaptive Streaming over HTTP*), HLS (ang. *HTTP Live Streaming*), Microsoft Smooth Streaming czy QuavStreams.

2.2.3. Zastosowanie

Wyróżnia się dwa sposoby dostępu do mediów strumieniowych:

— na żywo (ang. *live*) — dane docierają do odbiorców w czasie rzeczywistym, bezpośrednio po ich wygenerowaniu, np. radia internetowe,

— na zadanie (ang. *on demand*) — odbiorcy decydują, które dane, spośród dostępnych w danym serwisie, w którym momencie chcą odtworzyć, np. Youtube.

Streaming obecnie powszechny jest we większości dziedzin, takich jak edukacja, rozrywka, czy reklama.

2.3. VoIP

W porównaniu z telefonią tradycyjną, technologia VoIP zapewnia przede wszystkim niezależność od operatora, co w jasny sposób przekłada się na koszty eksploatacji. Wewnątrz sieci rozmowy będą darmowe, chcąc komunikować się z siecią konwencjonalną natomiast skorzystać możemy z odpowiednich bramek. W przypadku sieci pakietowej również lokalizacja geograficzna ma nikłe znaczenie. Jednakże w kontekście tematu pracy, kluczową jest możliwość integracji z przesyłem danych czy

obrazu. Odpowiednia implementacja zapewnić może płynną i darmową możliwość transmisji wszystkich tych rodzajów mediów, między całym światem.

2.3.1. Protokoły

Spośród protokołów stosowanych w technologii VoIP wysoką popularnością cieszy się H.323, jednakże przez swój poziom skomplikowania, wypierany jest przez SIP. Stosowane, lecz znacznie mniej popularnie, są również takie protokoły jak: MGCP (ang. *Media Gateway Control Protocol*), H.248 (znany również jako Megaco — ang. *Media Gateway Control*), IAX (ang. *Inter-Asterisk eXchange*) czy Jingle (będącym rozszerzeniem XMPP — ang. *Extensible Messaging and Presence Protocol*).

2.3.2. Implementacje

Dzięki funkcjonalności zwanej DDI (ang. *Direct Dial-In*), polegającej na możliwości dzwonienia bezpośrednio do abonenta wewnętrznego, a także niskim kosztom zarówno wdrożenia jak i utrzymania, VoIP zdobywa na popularności zarówno wśród użytkowników prywatnych jak i korporacyjnych. Już w 2008 roku 80% nowych, instalowanych centralek PBX (ang. *Private branch exchange*) to VoIP. Niezależność od zewnętrznych operatorów, niższy koszt połączeń z siecią PSTN, kompletna mobilność użytkowników czy w końcu możliwość transportu wszelkiego rodzaju danych, dzięki jednej infrastrukturze, w jasny sposób daje przewagę sieciom VoIP nad konwencjonalnymi rozwiązaniami.

3. Protokoły sygnalizacyjne

3.1. SIP

SIP (and. *Session Initiation Protocol*) jest protokołem warstwy aplikacji, służącym do ustanawiania, modyfikacji oraz zakańczania sesji multimedialnych. Wykorzystywany może być zarówno do transmisji typu *unicast*, jak i *multicast*. Korzystają z niego aplikacje oferujące usługi takie jak: wideokonferencje, streaming multimediów, komunikatory, transfer plików, fax czy gry online. Głównymi zadaniami SIP są:

— Lokalizacja użytkownika — ustalenie użytkownika końcowego.
— Dostępność użytkownika — ustalenie chęci do nawiązania połączenia.
— Możliwości użytkownika — negocjacja transmitowanych mediów oraz ich parametrów.
— Konfiguracja sesji — "dzwonienie", ustalenie parametrów sesji zarówno dla strony wybierającej jak i wybieranej.
— Zarządzanie sesją — przenoszenie i kończenie sesji, modyfikacja jej parametrów, wywoływanie usług.

Obecnie jest dominującym rozwiązaniem w sieciach VoIP i stopniowo wypiera H.323.

3.1.1. Zasada działania

Działa w warstwie aplikacji, na protokołach takich jak: TCP (ang. *Transmission Control Protocol*), UDP (ang. *User Datagram Protocol*), SCTP (ang. *Stream Control Transmission Protocol*). Jest oparty na transmisji czystego tekstu. Wiele elementów czerpie z protkołu HTTP (ang. *Hypertext Transfer Protocol*) oraz SMTP (ang. *Simple Mail Transfer Protocol*). Aby utworzyć pełnowartościowy system multimedialny współpracuje z innymi protokołami, wśród których najczęściej występują[3]:

— (S)RTP (ang. *(Secure) Real-time Transport Protocol*) — funkcje transportowe oraz QoS.
— RTSP (ang. *Real-Time Streaming Protocol*) — kontrola doręczenia.
— MEGACO (ang. *Media Gateway Control Protocol*) — nadzór nad bramkami do publicznej komutowanej sieci telefonicznej (PSTN).

— SDP (ang. *Session Description Protocol*) — opis sesji multimedialnych.
— TLS (ang. *Transport Layer Security*) — poufność i integralność danych.

Podobnie jak w protokole HTTP, zastosowano model wymiany danych na zasadzie żądanie-odpowiedź. Domyślnymi portami we wielu implementacjach są 5060 dla połączeń nieszyfrowanych i 5061 dla wykorzystujących TLS[16].

3.1.2. Elementy sieciowe

Urządzenia końcowe SIP komunikować mogą się ze sobą w sposób bezpośredni. Często jest to jednak niepraktyczne, dlatego też zdefiniowano następujące elementy sieciowe.

Agenty użytkowników

Agenty użytkowników (UA, ang. *users agents*) są końcowymi elementami używanymi do generowania lub otrzymywania wiadomości SIP. Na tej podstawie wyróżnia się dwa typy agentów: klient (UAC, ang. *User Agent Client*), wysyłający żądania, oraz serwer (UAS, ang. *User Agent Server*), otrzymujący żądania oraz wysyłający odpowiedzi.

Serwery proxy

Serwery proxy są jednostkami pośredniczącymi, działającymi zarówno jak serwery i klienty, w imieniu innych elementów. Interpretują one, i jeśli to konieczne, zmieniają część żądania przed przekazaniem dalej. Pełnią głównie funkcję routingowe, przekazując wiadomość do jednostki leżącej bliżej użytkownika końcowego. Wykorzystywane mogą być również w kwestiach bezpieczeństwa, przykładowo weryfikując uprawnienia.

Serwery przekierowań

Odpowiadają za wysyłanie do klienta wiadomości 3xx (*redirection*), zawierających adres następnego serwera. Nie akceptują połączeń, nie przetwarzają i nie przekazują wiadomości SIP.

Serwery rejestrujące

Po translacji URI na adres IP rejestrują obecność obsługiwanego użytkownika w sieci, przez co zwiększają jej efektywność.

Serwery lokalizacji

Odpowiadają za przechowywanie informacji o bieżącym stanie użytkownika, udostępniając je serwerom przekierowań oraz proxy.

Serwery aplikacji

Związane są ze świadczeniem konkretnych usług dodatkowych.

3.1.3. Wiadomości

Wiadomości SIP przekazywane są w formie tekstu kodowanego UTF-8[16], może nim być zarówno żądanie od klienta do serwera, jak i odpowiedź od serwera do klienta. Oba te typy komunikatów składają się z linii wstępnej (*Request-Line* lub *Status-Line*; opisane poniżej), jednego lub więcej pól nagłówka, pustej linii oraz opcjonalnie wiadomości. Składnia większości z nich jest analogiczna do komunikatów znanych z HTTP/1.1.

Żądania

Wiadomości typu *request* zaczyna następująca linia:

```
Method SP Request-URI SP SIP-Version CRLF
```

gdzie:

- Method — dozwolonych jest sześć metod:
  - REGISTER — rejestracja informacji o kontakcie.
  - INVITE — zestawienie sesji między UA.
  - ACK — potwierdzenie poprawnej wymiany wiadomości.
  - CANCEL — odrzucenie oczekującego żądania.
  - BEY — zakańczanie sesji.
  - OPTIONS — odpytywanie o możliwości serwera.
  - PRACK[7] (ang. *Provisional Response Acknowledgement*) — potwierdzenie odbioru tymczasowej wiadomości 1xx.

  Definiowane mogą być również dodatkowe metody:
- Request-URI — SIP lub SIPS URI (ang. *Uniform Resource Indicator*), wskazujący na użytkownika lub usługę do której *request* jest kierowany.
- SIP-Version — znajdująca się zarówno w wiadomości typu *request*, jak i *response*, wersja protokołu SIP.
- SP (ang. *Single Space*) — pojedynczy znak spacji.
- CRLF (ang. *Carriage-Return Line-Feed*) — znak nowej linii.

Odpowiedzi

Pierwsza linia komunikatu typu *response* prezentuje się następująco:

```
SIP-Version SP Status-Code SP Reason-Phrase CRLF
```

gdzie:

— Status-Code — trójcyforowa liczba, będąca formą *Reason-Phrase* przeznaczoną do interpretacji dla automatów. Pierwsza cyfra stanowi o klasie odpowiedzi, pozostałe dwie nie mają zestandaryzowanej funkcji. SIP/2.0 dopuszcza sześć klas:
  — 1xx: Provisional — żądanie otrzymane, przetwarzanie w toku.
  — 2xx: Success — żądanie zostało otrzymane, zrozumiane oraz zaakceptowane.
  — 3xx: Redirection — aby spełnić żądanie wymagane są dalsze działania.
  — 4xx: Client Error — żądanie o złej składni, lub niemogące być wypełnionym przez dany serwer.
  — 5xx: Server Error — serwer nie wypełnił poprawnego żądania.
  — 6xx: Global Failure — żądanie nie może zostać wypełnione przez żaden serwer.

— Reason-Phrase — opisowa forma odpowiedzi, dedykowana do analizy dla ludzi.

Nagłówki

Nagłówki podobne są do nagłówków HTTP — zarówno składnią jak i semantyką.

```
"header-name" HCOLON header-value *(COMMA header-value)
```

Mogą być rozszerzone na kilka linii, pod warunkiem, że każda kolejna na początku zawiera chociaż jedną spację. Kolejność pól nie ma znaczenia, jednak zalecanym jest, aby wykorzystywane w procesie przetwarzania przez proxy były na początku wiadomości[1]. Zarówno nazwy pól, jak i ich wartości (oprócz tych w cudzysłowiach), nie są czułe na wielkość liter. W żądaniach wymagane jest co najmniej sześć pól: To, From, CSeq, Call-ID, Max-Forwards oraz Via.

— Accept — akceptowalne w ciele wiadomości rodzaje mediów. Składnia identyczna z HTTP/1.1 z zastrzeżeniem, że jeśli pole nie zostało zdefiniowane, przyjmnowana jest wartość *application/sdp*. Stosowany może być wieloznacznik "*". Przykładowo:

```
Accept: application/sdp;level=1, application/x-private, text/html
```

— Accept-Encoding — pole podobne do poprzedniego, określa jednak akceptowalne kodowania. Przykładowo:

```
Accept-Encoding: gzip
```

— Accept-Language — preferowany język dla opisu sesji i innych pól. Przykładowo:

```
Accept-Language: da, en-gb;q=0.8, en;q=0.7
```

— Alert-Info — alternatywny ton dzwonka. Przykładowo:

```
Alert-Info: <http://www.example.com/sounds/moo.wav>
```

— Allow — lista metod wspierana przez UA. Przykładowo:

```
Allow: INVITE, ACK, OPTIONS, CANCEL, BYE
```

— Allow-Events — wspierane pakiety. Skróconą wersją jest *u*. Przykładowo:

```
Allow-Events:
u: conference dialog
```

— Answer-Mode — akceptowalne są dwie wartości: *Manual* oraz *Auto*. Podczas gdy ta pierwsza odpowiada za normalne zachowanie, druga stanowi o natychmiastowym przetwarzaniu i akceptacji metody INVITE. Przykładowo:

```
Answer-Mode: Auto;require
```

— Authentication-Info — pole związane z autentykacją HTTP Digest. Przykładowo:

```
Authentication-Info: nextnonce="47364c23432d2e131a5fb210812c"
```

— Authorization — wartości, którymi autentykuje się UA. Przykładowo:

```
Authorization: Digest username="Alice", realm="atlanta.com",
nonce="84a4cc6f3082121f32b42a2187831a9e",
response="7587245234b3434cc3412213e5f113a5432"
```

— Call-ID — pole jednoznacznie identyfikujące konkretne zaproszenie. Skróconą wersją jest *i*. Przykładowo:

```
Call-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6@biloxi.com
i:f81d4fae-7dec-11d0-a765-00a0c91e6bf6@192.0.2.4
```

— Call-Info — dodatkowe informacje o nadawcy lub odbiorcy. Przykładowo:

```
Call-Info: <http://wwww.example.com/alice/photo.jpg> ;purpose=icon,
<http://www.example.com/alice/> ;purpose=info
```

— Contact — URI, prezentowana nazwa lub parametry nagłówka. Skróconą wersją jest *m*. Przykładowo:

```
Contact: "Mr. Watson" <sip:watson@worcester.bell-telephone.com>
;q=0.7; expires=3600,
"Mr. Watson" <mailto:watson@bell-telephone.com> ;q=0.1
m: <sips:bob@192.0.2.4>;expires=60
```

— Content-Disposition — opis, w jaki sposób część informacyjna ma być zinterpretowana. Przykładowo:

```
Content-Disposition: session
```

— Content-Encoding — modyfikator wartości *media-type*, określa jakie dodatkowe kodowanie zostało zastosowane w części informacyjnej. Skróconą wersją jest *e*. Przykładowo:

```
Content-Encoding: gzip
e: tar
```

— Content-Language — określa język części informacyjnej. Przykładowo:

```
Content-Language: fr
```

— Content-Length — długość części informacyjnej — podana w oktetach. Skróconą wersją jest *l*. Przykładowo:

```
Content-Length: 349
l: 173
```

— Content-Type — Zawartość części informacyjnej wiadomości. Skróconą wersją jest *c*. Przykładowo:

```
Content-Type: application/sdp
c: text/html; charset=ISO-8859-4
```

— CSeq — zawiera nazwę metody oraz liczbę dziesiętną, umożliwiającą umiejscowienie wiadomości w całym dialogu oraz rozróżnienie nowego żądania od retransmisji. Przykładowo:

```
CSeq: 4711 INVITE
```

— Date — zawiera datę oraz godzinę (w GMT). Przykładowo:

```
Date: Sat, 13 Nov 2010 23:29:00 GMT
```

— Error-Info — informacja o błędzie. Przykładowo:

```
SIP/2.0 404 The number you have dialed is not in service
Error-Info: <sip:not-in-service-recording@atlanta.com>
```

— Expires — relatywny czas (w sekundach), po którym wiadomość przeterminowywuje się. Przykładowo:

```
Expires: 5
```

— From — zawiera nazwę nadawcy, jego URI oraz tag (jeden ze składników identyfikujących sesję, tworzy go ciąg losowych znaków). Skróconą wersją jest *f* Przykładowo:

```
From: "A. G. Bell" <sip:agb@bell-telephone.com> ;tag=a48s
From: sip:+12125551212@server.phone2net.com;tag=887s
f: Anonymous <sip:c8oqz84zk7z@privacy.org>;tag=hyh8
```

— In-Reply-To — ID rozmów do których odnosi się nagłówek. Przykładowo:

```
In-Reply-To: 70710@saturn.bell-tel.com, 17320@saturn.bell-tel.com
```

— Max-Forwards — limit przejść między serwerami, każdy z nich dekrementuje tę wartość. W przypadku wyzerowania liczby przed dotarciem do adresata, żądanie jest anulowane z kodem błędu 483 (*Too Many Hops*). Przykładowo:

```
Max-Forwards: 6
```

— Min-Expires — pole używane w odpowiedzi na REGISTER z przekroczonym czasem wygaśnięcia. Przykładowo:

```
Min-Expires: 60
```

— MIME-Version — określa wersję protokołu MIME użytego przy konstrukcji ciała wiadomości. Przykładowo:

```
MIME-Version: 1.0
```

— Organization — określa organizację, do której kierowana jest wiadomość. Może zostać użyte przez klienta do filtrowania połączeń. Przykładowo:

```
Organization: Boxes by Bob
```

— Priority — priorytet wiadomości, użyty może być do decyzji o routingu czy akceptacji rozmowy. Brak pola oznacza *normal*, zalecanymi wartościami są: *non-urgent*, *normal*, *urgent*, *emergency*, lecz zdefiniowane mogą być również inne. Przykładowo:

```
Subject: A˜tornado is heading our way!
Priority: emergency
```

— Proxy-Authenticate — definiuje kwestie związane z autentykacją. Przykładowo:

```
Proxy-Authenticate: Digest realm="atlanta.com",
domain="sip:ss1.carrier.com", qop="auth",
nonce="f84f1cec41e6cbe5aea9c8e88d359",
opaque="", stale=FALSE, algorithm=MD5
```

— Proxy-Authorization — pozwala klientowi na identyfikację siebie lub swoich użytkowników na serwerze proxy. Przykładowo:

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

— Proxy-Require — wymagane do poprawnego przetworzenia wiadomości przez proxy rozszerzenia. Przykładowo:

```
Proxy-Require: timer
```

— Record-Route — stanowi o wymogu transmisji kolejnych żądań w sesji przez serwer proxy. Przykładowo:

```
Record-Route: <sip:server10.biloxi.com;lr>,
<sip:bigbox3.site3.atlanta.com;lr>
```

— Reply-To — URI, które powinno zostać użyte przy odpowiedzi na dane żądanie. Przykładowo:

```
Reply-To: Bob <sip:bob@biloxi.com>
```

— Require — rozszerzenia wymagane przez UAC i UAS do przetworzenia żądania. Przykładowo:

```
Require: 100rel
```

— Retry-After — definiuje przewidywany czas niedostępności w przypadku kodu błędu. Przykładowo:

```
Retry-After: 18000;duration=3600
Retry-After: 120 (I'm in a~meeting)
```

— Route — wymuszenie routingu przez wylistowaną listę proxy. Przykładowo:

```
Route: <sip:bigbox3.site3.atlanta.com;lr>,
<sip:server10.biloxi.com;lr>
```

— Server — informacja o oprogramowaniu UAS wykorzystywanym przy danym żądaniu. Przykładowo:

```
Server: HomeServer v2
```

— Subject — podsumowanie zestawianego połączenia, umożliwia ich filtrowanie bez potrzeby parsowania opisu sesji. Skróconą wersją jest *s*. Przykładowo:

```
Subject: Need more boxes
s: Tech Support
```

— Supported — rozszerzenia wspierane przez UAC lub UAS. Przykładowo:

```
Supported: 100rel
```

— Timestamp — opisuje czas, w którym żądanie zostało wygenerowane. Przykładowo:

```
Timestamp: 54
```

— To — zawiera URI adresata oraz tag. Skróconą wersją jest *t*. Przykładowo:

```
To: The Operator <sip:operator@cs.columbia.edu>;tag=287447
t: sip:+12125551212@server.phone2net.com
```

— Unsupported — właściwości niewspierane przez UAS. Przykładowo:

```
Unsupported: foo
```

— User-Agent — informacja o UA, który wygenerował żądanie. Przykładowo:

```
User-Agent: Softphone Beta1.5
```

— Via — określa trasę wiadomości, od źródła poprzez serwery pośrednie. Przykładowo:

```
Via: SIP/2.0/UDP erlang.bell-telephone.com:5060;branch=z9hG4bK87asdks7
Via: SIP/2.0/UDP 192.0.2.1:5060 ;received=192.0.2.207
;branch=z9hG4bK77asjd
```

— Warning — dodatkowe informacje o statusie odpowiedzi. Przykładowo:

```
Warning: 307 isi.edu "Session parameter 'foo' not understood"
```

— WWW-Authenticate — kwestie związane z autentykacją. Przykładowo:

```
WWW-Authenticate: Digest realm="atlanta.com",
domain="sip:boxesbybob.com", qop="auth",
nonce="f84f1cec41e6cbe5aea9c8e88d359",
opaque="", stale=FALSE, algorithm=MD5
```

Sekcja SDP

Informacje, których przekazanie jest konieczne, a niemożliwe lub niechciane w formie nagłówków SIP, przenosi część informacyjna wiadomości, nazywana ciałem (ang. *body*). Najczęstszym protokołem opisu sesji multimedialnej jest SDP. (Session Description Protocol). Opis ten zawiera wiele pól w formacie `<character>=<value>`, rozdzielonych znakiem nowej linii. Wyróżnia się trzy sekcje, odpowiedzialne za opis sesji, czasu oraz mediów. Dwie pierwsze są obowiązkowe. Poniżej scharakteryzowano owe pola z podziałem na sekcje, zaznaczając znakiem "*" pola obligatoryjne.

— Sesja

```
v=  wersja protokołu
o=  pochodzenie oraz ID sesji
s=  nazwa sesji
i=* tytuł lub krótka informacja o sesji
u=* URI opisu
e=* adresy email
p=* numery telefonów
c=* informacje o połączeniu
b=* informacje o przepustowości
Pola opisu czasu (patrz poniżej).
z=* korekty zgodnie ze strefą czasową
k=* klucz szyfrowania
a=* atrybuty sesji – profil, kodek, częstotliwość próbkowania
Pola opisu mediów.
```

— Czas

```
t=  czas aktywacji sesji
r=* czas powtórzeń
```

— Medium

```
m=  media – typ, port, profil RTP/AVP
i=* tytuł oraz pole informacyjne
c=* dane połączenia – opcjonalne, jeśli zawarte w opisie sesji
b=* informacje o przepustowości
k=* klucz szyfrowania
a=* atrybuty mediów, nadpisujące atrybuty sesji
```

Przykładowy opis przy użyciu SDP:

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
```

Protkół SIP dokonuje analizy wyłącznie adresu IP, numeru portu oraz trybu kodowania treści medialnych, analizę pozostałych pól zostawiając aplikacjom wyższego poziomu.

3.1.4. Identyfikator

Identyfikator SIP URI (ang. Uniform Resource Indicator), który przybiera następującą postać[8]

```
user@host
```

służy do wskazania określonego zasobu systemu, którym może być użytkownik, terminal, skrzynka pocztowa czy numer PSTN przyłączony za pomocą bramki. Podczas gdy informacja przekazana powinna być w trybie bezpiecznym, identyfikator przyjmuje formę SIPS URI (sips:). Pełen identyfikator SIP prezentuje się następująco:

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

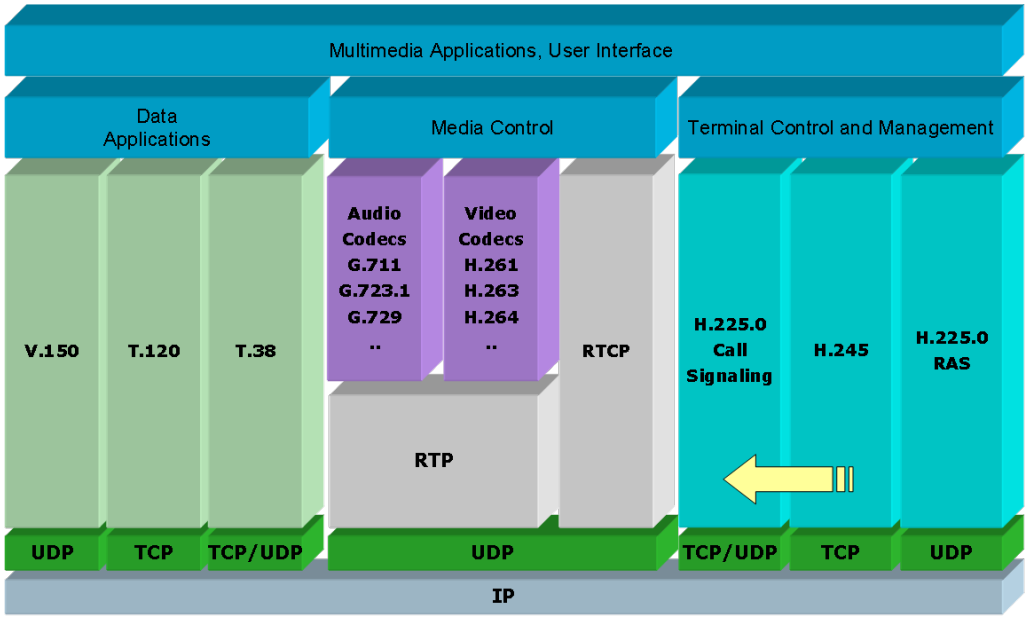
gdzie:

— *user* — ciąg znakowy identyfikujący użytkownika. Również numer telefoniczny, jeśli host jest je w stanie przetwarzać.
— *password* — hasło powiązane z użytkownikiem, w związku z przesyłem w formie czystego tekstu, pole to nie jest rekomendowane.
— *host* — pełna nazwa domenowa hosta, lub adres IP (zarówno w wersji 4. jak i 6.).
— *port* — port przeznaczenia wiadomości.
— *uri-parameters* — parametry w formie parametr=wartość, rozdzielone średnikiem.
— *headers* — elementy wykorzystywane podczas konstruowania żądania SIP na podstawie URI, rozdzielane znakiem *et*. Dopuszczane są następujące parametry:
  — *transport* — określa protokół warstwy transportowej.
  — *madrr* — adres serwera, który uczestniczy w transmisji.
  — *ttl* (ang. *time to live*) — maksymalna ilość przejść pakietu UDP rozsyłanego w trybie wielopunktowym.
  — *user* — gdy przyjmuje wartość "phone" jest to informacja, że pole "user" jest adresem telefonicznym.
  — *method* — typ żądania SIP konstruowanego przy użyciu elementów URI.
  — *lr* — parametr zapewniający wsteczną kompatybilność.

3.2. H.323

H.323 jest zaleceniem definiowanym przez ITU-T, opisującym zastosowanie protokołów ITU-T oraz IETF przeznaczonych do transmisji siecią pakietową danych audiowizualnych. Obejmuje kontrolę połączenia, sygnalizację, transport i kontrolę multimediów czy też nadzór nad pasmem. Działa na protokołach TCP/IP oraz UDP. Składa się z następujących protokołów[1]:

— H.225 — sygnalizacja, RAS (ang. *Registration, Admission and Status*),
— H.235 — bezpieczeństwo,
— H.239 — wideokonferencje,
— H.245 — negocjacje odnośnie kanału,
— H.450 — usługi dodatkowe,
— H.460 — rozszerzenia, które mogą być zaimplementowane w gatekeeperze.
— Q.931 — sygnalizacja,
— G.711, G.722, G.723, G.728, G.729 — kodeki dźwięku,
— H.261, H.263 — kodeki obrazu,
— RTP — transport mediów.



Rysunek 3.1. Struktura protokołów rodziny H.323[18].

3.2.1. Budowa protokołów

Większość implementacji wykorzystuje TCP jako protokół transportowy dla sygnalizacji, H.323 od wersji drugiej umożliwia jednak zastosowanie UDP. Rodzinę protokołów H.323 dzieli się na trzy główne obszary.

H.225.0

Po rozwiązaniu adresu, celem ustanowienia połączenia, wykorzystywany jest protokół H.225. Wiadomościami wykorzystywanymi przez ten protokół są:
— *Setup and Setup acknowledge*,
— *Call Proceeding*,
— *Connect*,
— *Alerting*,
— *Information*,
— *Release Complete*,
— *Facility*,
— *Progress*,
— *Status and Status Inquiry*,
— *Notify*.

RAS

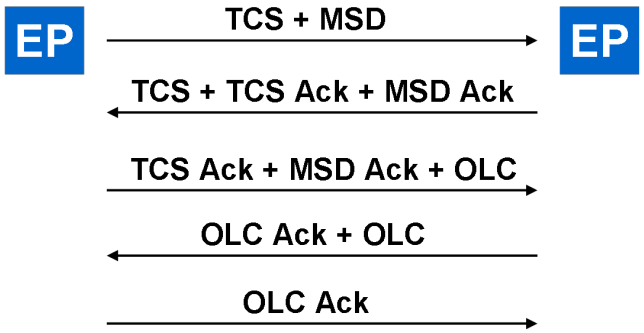
Sygnalizacja RAS (ang. *Registration, Admission and Status*) służy do komunikacji ze strażnikami. Kanał RAS jest niezależny od innych kanałów, zestawiany jest przed nimi. Dane przenoszone są protokołem UDP. Wykorzystuje następujące wiadomości:
— xRQ — *Request*,
— xRJ — *Reject*,
— xCF — *Confirm*,
— *nonStandardMessage*,
— *unknownMessage*,
— RIP — *Request in Progress*,
— RAI/RAC — *Resource Availability Indication/Confirm*,
— SCI/SCR — *Service Control Indication/Response*.
Kiedy podłączany jest terminal, wysyła on wiadomość GRQ, aby sprawdzić dostępność strażników, którzy odpowiedzą potwierdzeniem GCF. Po otrzymaniu potwierdzenia generowane jest żądanie rejestracji RRQ, po uzyskaniu na które odpowiedzi RCF, możliwe będą połączenia. W przypadku takiej konieczności wysyłane jest żą-

danie wstępne ARQ, na które, po rozwinięciu adresu przez gatekeepera, zwracane jest potwierdzenie ACF.

H.245

Podstawowym zadaniem protokołu H.245 jest transmitowanie wiadomości sygnalizacyjnych, potrzebnych do takich funkcji jak szyfrowanie, kontrola nad jitterem, określenie relacji master i slave, otwieranie i zamykanie kanałów służących do transportu mediów. Prawdopodobnie najważniejszą z nich jest jednak negocjacja parametrów transmisji. Ustalane są między innymi kodeki, którymi mogą być:

— audio — G.711, G.729, G.723.1, G.726, G.722, G.728, Speex, AAC-LD,
— wideo — H.261, H.263, H.264,
— tekst — T.140.



Rysunek 3.2. Typowa sesja H.245[18].

3.2.2. Architektura

Zalecenie opisuje elementy sieciowe, które współpracując ze sobą, zapewnić mogą kompleksową sieć wymiany danych multimedialnych. Aby komunikacja była możliwa, zaimplementować wystarczy dwa terminale, większość sieci wykorzystuje również gatekeepera, pozostałe elementy zapewniają kolejne, dodatkowe funkcjonalności.

Terminale

Mogą nimi być zarówno aplikacje (ang. *softphone*), jak i fizyczne telefony czy przystawki konferencyjne. Wykorzystywać mogą jednocześnie więcej niż jeden kanał transmisyjny. Dzięki odpowiednim specyfikacjom współpracować mogą z systemami ISDN, PSTN oraz innymi.

Brama

Brama sieciowa (ang. *gateway, GW*) jest interfejsem do sąsiednich sieci czy domen, wykorzystujących inne technologie, takie jak wspomniane wcześniej PSTN czy ISDN. Dokonują translacji między formatami transmisji danych, dźwięku czy obrazu. Używane również są w celu umożliwienia wideokonferencji opartych na H.320 czy H.324 w systemie H.323[16].

MCU

Jednostka MCU (ang. *Multipoint Control Units*) odpowiedzialna jest za połączenia wielopunktowe — konferencje. Wewnątrz pojęcia MCU wyróżnia się sterownik MC (ang. *Multipoint Controller*), negocjujący z terminalem warunki połączenia, oraz procesor MP (ang. *Multipoint Processor*), który odpowiedzialny jest za transfer mediów oraz danych sygnalizacyjnych. Podstawowym zadaniem MCU jest odbiór sygnałów audio, ich miksowanie oraz rozsyłanie do uczestników konferencji. W przypadku wideokonferencji, miksowanie danych nie wchodzi w grę, proponowane więc są inne rozwiązania. Pierwszym z nich jest rozsyłanie sygnału generowanego przez stronę o najwyższym w danym momencie poziomie sygnału dźwięku. Kolejnym — decydowanie o źródle transmitowanego obrazu przez nadzorcę. Najpopularniejszym obecnie jednak wydaje się być tryb ciągłej dostępności wszystkich stron konferencji. MCU przyjmuje postać sprzętową lub programową czy mieszaną.

Gatekeeper

Strażnik bramy, popularnie nazywany gatekeeperem lub skrótowo GK, odpowiada za zarządzanie przydzielonym mu obszarem sieci, wydzielonym od pozostałych jej elementów. Realizuje takie funkcje jak:

— translacja adresów — na podstawie aliasów H.323 lub adresów E.164 określa adres IP,

— zarządzanie dostępem — autoryzacja przy użyciu komunikatów żądania, potwierdzenia i odrzucenia (ang. *ARQ - Admission Request, ACF - Admission Confirm, ARJ - Admission Reject*),

— kontrola pasma — zapewniana przy użyciu komunikatów żądania, potwierdzenia i odrzucenia pasma (ang. *BRQ - Bandwidth Request, BCF - Bandwidth Confirm, BRJ - Bandwidth Reject*),

— zarządzanie strefą.

Może realizować również funkcje routingowe, co daje asumpt do systemu bilingowego, przekierowania sesji do właściwych bramek czy innych funkcjonalności.

Serwer proxy

Działa na poziomie warstwy aplikacji, stosowany do sprawdzania pakietów transmitowanych w danym punkcie sieci i przekierowania ich w odpowiednie miejsce.

3.3. XMPP/Jingle

XMPP (ang. *Extensible Messaging and Presence Protocol*), wywodzący się z Jabbera, jest otwartym protokołem opartym na XML (ang. E*xtensible Markup Language*), wykorzystywanym miedzy innymi w komunikatorach internetowych.

3.3.1. Opis

Wykorzystuje relacje klient-serwer — dane przesyłane miedzy klientami przechodzą przez serwer. Format nazw użytkowników (JID, ang. *Jabber ID*), jest analogiczny z adresami e-mail. Dane przesyłane są w formie XML, co definiują odpowiednie zalecenia. Jingle natomiast jest rozszerzeniem protokołu XMPP dla sesji multimedialnych. Dodaje funkcje kontrolne dla przekazu typu peer-to-peer, wideokonferencji i innych. Zdefiniowany został przez Google oraz XMPP Standards Foundation. Dane sygnalizacyjne przesyłane są w formacie XML, natomiast dane użytkownika po protokole RTP, co implikuje możliwość integracji z istniejącymi już systemami. Składnia Jingle zaprojektowana została wręcz w ten sposób, aby współpraca z SIP oraz SDP była prosta i niekosztowna.

3.3.2. Wykorzystanie

Na protokole XMPP/Jingle oparte są komunikatory takich witryn jak Facebook, Nk.pl czy niegdyś Google Talk. Na szczególna uwagę zasługuje ten ostatni — kombinacja idei komunikatora internetowego (IM, ang. Instant messaging) z VoIP. Obecnie znany jest jako Google Hangout i odszedł od protokołu Jingle, lecz w czasach jego istnienia, idealnie ukazywał możliwości tego protokołu. Komunikator umożliwiał konwersacje — zarówno tekstowe, dźwiękowe oraz wideo — miedzy wieloma użytkownikami, zapewniając przy tym płynność przekazu oraz wiele dodatkowych możliwości, takich jak możliwość integracji z innymi, licznymi i również opartymi na XMPP usługami świadczonymi przez równie liczne serwisy.

3.4. Porównanie

Dwa z omówionych powyżej protokołów walczą obecnie o dominacje na rynku — lecz H.323 stopniowo wypierany jest przez SIP. Mimo, ze ten drugi generalnie wydaje się być prostszym i zarazem łatwiej rozszerzalnym, również H.323 ma swoich zwolenników oraz zalety. Tabela 3.1 prezentuje pokrótce podobieństwa i różnice między nimi.

Tabela 3.1. Porównanie SIP i H.323.

Definicja protokołu	Zdefiniowane w 1996 przez ITU-T[10], w oparciu o Q.931[9]. Ostatnia wersja protokołu z roku 2009[12].	Zdefiniowane w 1996 przez IETF w RFC 3261[6]. Ostatnia wersja protokołu z 2002[8].
Format wiadomość	ASN.1, nieczytelny dla człowieka format binarny.	ABFN, czytelne dla człowieka ASCII, co czyni wiadomości większymi.
Transport mediów	RTP/SRTP.	RTP/SRTP.
Rozszerzalność	Kolejne wersje dodają nowe funkcjonalności, zapewniając kompatybilność wsteczną i bezkolizyjność. Unikalne identyfikatory zapewniają brak problemów we współpracy między rozszerzeniami różnych firm.	Kolejne wersje dodają nowe funkcjonalności, zapewniając bezkolizyjność z obecnymi, lecz kompatybilność wsteczna nie zawsze jest zapewniona. Łatwo rozszerzalny poprzez dodawanie kolejnych nagłówków, lecz bez wymaganych różnic w identyfikatorach, co może skutkować konfliktami.
Adresy	Wsparcie dla wielu typów adresów, m.in. URI, e-mail, ISUP, H.323 ID, E.164.	Jedynie URI, w przypadku współpracy z innymi urządzeniami konieczna jest translacja adresów.
Protokół transportowy	Połączeniowy lub bezpołączeniowy, np. TCP lub UDP, głównie stosowany połączeniowy.	Połączeniowy lub bezpołączeniowy, np. TCP lub UDP, głównie stosowany bezpołączeniowy.
Topologia	Unicast, multicast, gwiazda, scentralizowana.	Unicast, multicast, gwiazda, scentralizowana.
Multicast	Przez wiadomości LRQ i GRQ.	Przez grupowe wiadomości INVITE.
Bezpieczeństwo	Mechanizmy bezpieczeństwa zdefiniowane w H.235, możliwe również użycie SSL	Autentykacja przez mechanizmy HTTP, szyfrowanie SSL/TSL (a- lub symetryczne), możliwe wykorzystanie innych mechanizmów warstwy transportowej. Klucze przekazywane SDP. Możliwe wykorzystanie PGP lub S/MIME.
Kontrola przez stronę trzecią	Dostępna przez sam H.323 jak i H.450.	Dostępna w SIP.
Wykrywanie zapętleń	Zapętlenia wykryte mogą być za pomocą ID oraz adresu docelowego.	Nagłówek Via oraz Max-Forwards zapobiegają pętlom.

4. Aplikacje analizujące działanie protokołów

Aby zaprezentować działanie jakiegokolwiek protokołu, użyć należy programu typu sniffer. Aplikacje tego typu są w stanie przechwycić i ewentualnie przeanalizować ruch sieciowy. Stanowią często niezastąpione narzędzie diagnostyczne sieci, lecz zabraknąć ich nie może również w procesie nauki o danym protokole.

Poniżej przedstawiono najpopularniejsze aplikacje tego typu[17].

4.1. Ogólnodostępne aplikacje

4.1.1. *tcpdump* i WinDump

Podstawowym snifferem na większości systemów uniksowych jest *tcpdump*. Jest aplikacją konsolową, udostępniającą szereg opcji. Podstawową z nich jest wyświetlanie zawartości pakietów sieciowych, które przechwyci ze wskazanego (poprzez parametr *-i*) interfejsu sieciowego. Aplikacja służy do przechwytywanie nie tylko, na co wskazywać mogłaby nazwa, pakietów TCP/IP. Jej portem na Windowsa jest WinDump.

Przykład zastosowania podczas analizy protokołu SIP:

```
root@localhost:~# tcpdump -vv -c 10 -i wlan0 '((udp) and \
> (src host 87.99.19.238) and (src port 32188) and \
> (dst host 192.168.1.100) and (dst port 5060))' \
> -w packets_from_tcpdump
    tcpdump: listening on wlan0, link-type EN10MB (Ethernet), \
        capture size 65535 bytes
    10 packets captured
    13 packets received by filter
    0 packets dropped by kernel
```

Wywołanie tego typu przechwyci 10 kolejnych pakietów protokołu UDP z interfejsu *wlan0*, odfiltrowując jedynie te, pochodzące z IP 87.99.19.238 oraz portu 32188,

których adresatem jest 192.168.1.100 i port 5060. Dane zapisane zostaną do pliku typu pcap[20] o nazwie *packets_from_tcpdump*:

```
root@localhost:~# file packets_from_tcpdump
packets_from_tcpdump: tcpdump capture file (little-endian) - \
    version 2.4 (Ethernet, capture length 65535)
```

Odczytane mogą być zarówno za pomocą podstawowych narzędzi do przetwarzania tekstu (lecz w tym przypadku dane w formie czystego tekstu wymieszane będą z danymi binarnymi, zaciemniającymi obraz), ale także przy pomocy *tcpdump* i przełącznika *-r*, który dokona wstępnej analizy i zaprezentuje pakiety w formie bardziej przystępnej:

```
root@localhost:~# head -25 packets_from_tcpdump
    PUBLISH sip:user1@87.99.19.238;transport=UDP SIP/2.0
    Via: SIP/2.0/UDP 87.99.19.238:32188;\
        branch=z9hG4bK-d8754z-bc3fcace7bc791a6-1---d8754z-
    Max-Forwards: 70
    Contact: <sip:user1@87.99.19.238:32188;transport=UDP>
    To: <sip:user1@87.99.19.238;transport=UDP>
    From: <sip:user1@87.99.19.238;transport=UDP>;tag=63c34008
    Call-ID: YjE4NDExM2Q0ZTExNWU0NTY0YjZmMjljY2U1YWUzMWM.
    CSeq: 1 PUBLISH
    Expires: 600
    Allow: INVITE, ACK, CANCEL, BYE, NOTIFY, REFER, \
        MESSAGE, OPTIONS, INFO, SUBSCRIBE
    Content-Type: application/pidf+xml
    Supported: replaces, norefersub, extended-refer, \
        timer, X-cisco-serviceuri
    User-Agent: Z 3.2.21357 r21103
    Event: presence
    Allow-Events: presence, kpml
    Content-Length: 268

    <?xml version="1.0" encoding="UTF-8"?>
    <presence xmlns="urn:ietf:params:xml:ns:pidf"
            entity="sip:user1@87.99.19.238;transport=UDP">
    <tuple id="user1" >
        <status><basic>open</basic></status>
        <note>On the phone</note>
    </tuple>
    </presence>

root@localhost:~# tcpdump -r packets_from_tcpdump -vv |head -6
```

```
reading from file packets_from_tcpdump, link-type EN10MB (Ethernet)
13:20:46.876107 IP (tos 0x0, ttl 63, id 58919, offset 0, \
    flags [DF], proto UDP (17), length 997)
    87-99-19-238.internetia.net.pl.32188 > linux.local.sip: \
        [udp sum ok] SIP, length: 969
        PUBLISH sip:user1@87.99.19.238;transport=UDP SIP/2.0
        Via: SIP/2.0/UDP 87.99.19.238:32188;\
            branch=z9hG4bK-d8754z-bc3fcace7bc791a6-1---d8754z-
        Max-Forwards: 70
        Contact: <sip:user1@87.99.19.238:32188;transport=UDP>
```

4.1.2. *ettercap*

Program ettercap dostępny jest zarówno pod systemy uniksowe, jak i Windowsy, lecz w przeciwieństwie do poprzednika, udostępnia interfejs graficzny (zarówno Ncurses, jak i GTK+). Zapewnia możliwość filtrowania względem IP, portu, MAC, czy konkretnych ciągach znaków, a także możliwość zamiany w locie danych ciągów, wykonania zdefiniowanej aplikacji przy każdym pakiecie spełniającym zadane kryterium oraz wiele innych możliwości. Filtry pisane są w składni niemal identycznej do języka C, a następnie kompilowane za pomocą narzędzia *etterfilter*, który zapisuje dane binarne do pliku *.ef*:

```
root@localhost:~# cat filter
if (ip.proto == UDP && udp.src == 32188 && udp.dst == 5060 && \
        ip.src == '87.99.19.238' && ip.dst == '192.168.1.100')
{
    msg("UDP package from 87.99.19.238 to 192.168.1.100\n");
    log(DATA.data, "./packets_from_ettercap");
}

root@localhost:~# etterfilter filter
etterfilter 0.8.0 copyright 2001-2013 Ettercap Development Team
 12 protocol tables loaded:
        DECODED DATA udp tcp gre icmp ip arp wifi fddi tr eth
 11 constants loaded:
        VRRP OSPF GRE UDP TCP ICMP6 ICMP PPTP PPPoE IP ARP
 Parsing source file 'filter'  done.
 Unfolding the meta-tree  done.
 Converting labels to real offsets  done.
 Writing output to 'filter.ef'  done.
 -> Script encoded into 13 instructions.
```

```
root@localhost:~# file filter.ef
filter.ef: data
```

Przykładowe dane otrzymane podczas analizy protokołu SIP:

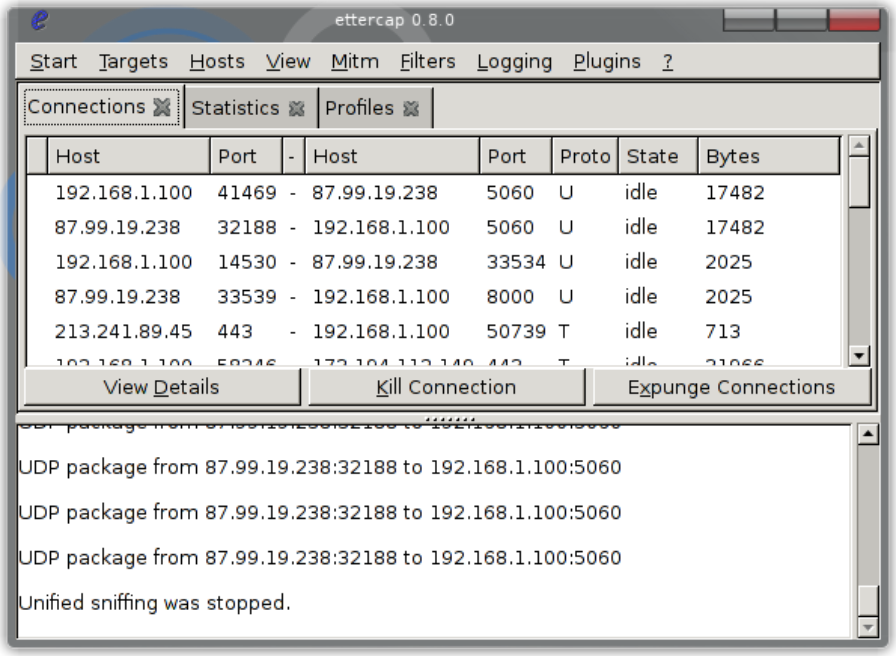
```
root@localhost:~# ettercap -i wlan0 -T -F filter.ef |head -100
ettercap 0.8.0 copyright 2001-2013 Ettercap Development Team
Content filters loaded from filter.ef...
Listening on:
 wlan0 -> 24:FD:52:9D:BB:AA
          192.168.1.100/255.255.255.0
          fe80::26fd:52ff:fe9d:bbaa/64
(...)
Starting Unified sniffing...

Sun Jun  1 14:35:59 2014
UDP  87.99.19.238:32188 --> 192.168.1.100:5060 |

SUBSCRIBE sip:user1@87.99.19.238;transport=UDP SIP/2.0.
Via: SIP/2.0/UDP 87.99.19.238:32188;branch=z9hG4bK-d8754z.
Max-Forwards: 70.
Contact: <sip:user1@87.99.19.238:32188;transport=UDP>.
```

Dostępne są również dwie wersje graficzne: GTK+ (rys. 4.1) oraz curses (rys. 4.2), dające podobne możliwości, lecz niewymagające od użytkownika znajomości przełączników i ich funkcji.

W obu nakładkach graficznych wczytane mogą zostać opisane uprzednio filtry, również dane wyeksportowane mogą zostać do plików.

4.1.3. *dSniff*

dSniff jest zestawem narzędzi służących do sniffowania i analizowania ruchu sieciowego. Dostępny pod systemy uniksowe. Zestaw ten składa się z następujących aplikacji: *dsniff*, *filesnarf*, *mailsnarf*, *msgsnarf*, *urlsnarf*, *webspy*, *arpspoof*, *dnsspoof*, oraz *macof*. Służą do tak zróżnicowanych operacji, jak przekierowywanie pakietów, wysyłanie sfałszowanych odpowiedzi na zapytania DNS, przechwytywanie danych przesyłanych protokołem NFS, przepełnianie pamięci switcha, przechwytywanie maili, wiadomości IRC, adresów URL z zapytań HTTP i innych. Najważniejszą dla prezentowanej tematyki jest jednak aplikacja *dsniff*, która jest snifferem wraz z interpreterem wielu popularnych protokołów. Nie ma jednak wśród nich protokołów VoIP.



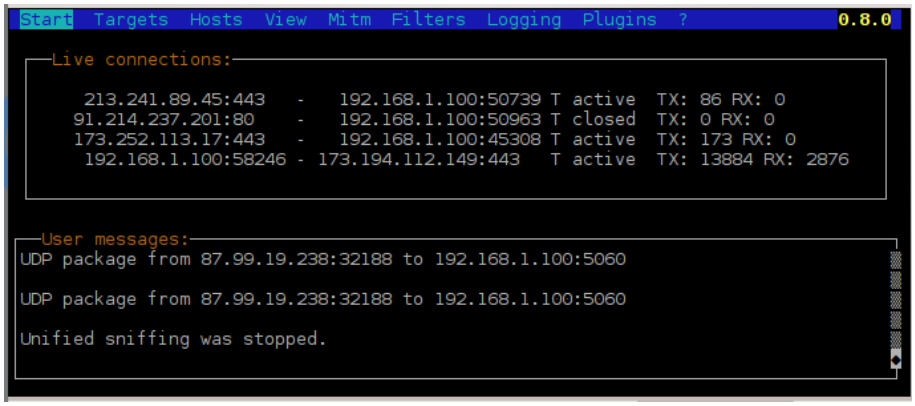
Rysunek 4.1. Okno programu *ettercap* GTK+.

4.1.4. *snort*

Dostępny zarówno pod Linuksy jak i Windowsy. Oferuje możliwość zarówno przechwytania pakietów, jak i ich analizy, wyszukiwania oraz dopasowywania zawartości. Ma to praktyczne zastosowanie, przykładowo do wykrywania prób ataków, lecz równie użyteczne może być podczas analizy protokołów VoIP.

Przykład zastosowania podczas analizy protokołu SIP:

```
root@localhost:~# snort -i wlan0 -v -d '((udp) and \
> (src host 87.99.19.238) and (src port 32188) and \
> (dst host 192.168.1.100) and (dst port 5060))'
06/01-18:14:32.174015 87.99.19.238:32188 -> 192.168.1.100:5060
UDP TTL:63 TOS:0x0 ID:63653 IpLen:20 DgmLen:913 DF
Len: 885
53 55 42 53 43 52 49 42 45 20 73 69 70 3A 75 73  SUBSCRIBE sip:us
65 72 31 40 38 37 2E 39 39 2E 31 39 2E 32 33 38  er1@87.99.19.238
3B 74 72 61 6E 73 70 6F 72 74 3D 55 44 50 20 53  ;transport=UDP S
49 50 2F 32 2E 30 0D 0A 56 69 61 3A 20 53 49 50  IP/2.0..Via: SIP
2F 32 2E 30 2F 55 44 50 20 38 37 2E 39 39 2E 31  /2.0/UDP 87.99.1
39 2E 32 33 38 3A 33 32 31 38 38 3B 62 72 61 6E  9.238:32188;bran
63 68 3D 7A 39 68 47 34 62 4B 2D 64 38 37 35 34  ch=z9hG4bK-d8754
7A 2D 39 63 33 31 35 61 66 32 33 39 65 63 65 38  z-9c315af239ece8
63 34 2D 31 2D 2D 2D 64 38 37 35 34 7A 2D 0D 0A  c4-1---d8754z-..
4D 61 78 2D 46 6F 72 77 61 72 64 73 3A 20 37 30  Max-Forwards: 70
```



Rysunek 4.2. Okno programu *ettercap* curses.

```
0D 0A 43 6F 6E 74 61 63 74 3A 20 3C 73 69 70 3A   ..Contact: <sip:
75 73 65 72 31 40 38 37 2E 39 39 2E 31 39 2E 32   user1@87.99.19.2
33 38 3A 33 32 31 38 38 3B 74 72 61 6E 73 70 6F   38:32188;transpo
72 74 3D 55 44 50 3E 0D 0A 54 6F 3A 20 3C 73 69   rt=UDP>..To: <si
70 3A 75 73 65 72 31 40 38 37 2E 39 39 2E 31 39   p:user1@87.99.19
2E 32 33 38 3B 74 72 61 6E 73 70 6F 72 74 3D 55   .238;transport=U
44 50 3E 0D 0A 46 72 6F 6D 3A 20 3C 73 69 70 3A   DP>..From: <sip:
```

Domyślną formą zapisu do pliku, uzyskiwaną przez przełącznik *-l*, jest forma binarna. Dodając natomiast przełącznik *-b*, uzyskujemy format identyczny z używanym przez *tcpdump*. Również format filtru jest identyczny w obu aplikacjach.

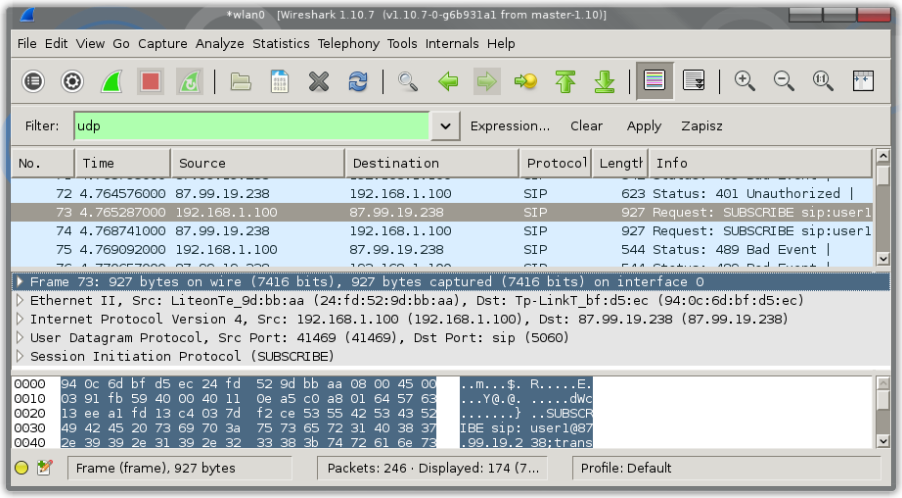
4.1.5. Wireshark

Prawdopodobnie najbardziej rozbudowana z do tej pory wymienionych aplikacji, a zarazem jako pierwsza oferująca rozbudowane GUI. Dostępna jest również wersja konsolowa — TShark, która jednak, przez wzgląd na szereg omówionych uprzednio aplikacji konsolowych, nie zostanie obecnie przybliżona. Wireshark jest aplikacją wieloplatformową. Oprócz funkcjonalności oferowanych przez *tcpdump* oraz graficznej na nie obudowy, oferuje bardzo użyteczne funkcje sortowania, filtrowania i przetwarzania pakietów. Oprócz przechwytywania bezpośrednio z interfejsu sieciowego, jest w stanie analizować ruch zapisany przy użyciu większości popularnych snifferów. Pakiety wyświetla w uporządkowanej i czytelnej formie.

Wspiera następujące protokoły VoIP: SIP, H.323, ISUP, MGCP oraz UNISTIM[21] i wyświetla wyłuskane z nich informacje, również w formie grafu. Jest również w stanie odtworzyć przesłany dźwięk (G.711 lub RTP).

Po uruchomieniu, wybraniu odpowiedniego interfejsu sieciowego oraz włączenia przechwytywania pakietów, na głównym ekranie ukazują się one w uporządkowanej

formie — w tabeli z kolumnami zawierającymi najważniejsze informacje. Po wybraniu danego pakietu, ukazują się jego szczegóły zarówno w formie heksadecymalnej, jak bardziej przystępnej dla człowieka, wraz z podziałem na warstwy (rys. 4.3).



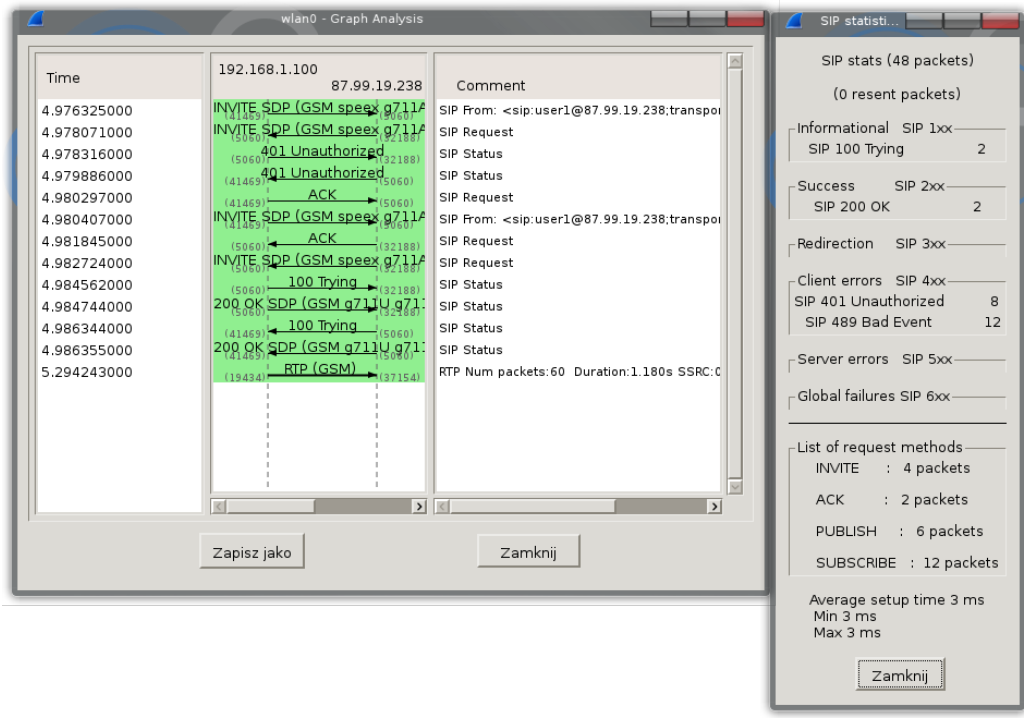
Rysunek 4.3. Główne okno programu Wireshark.

Po rozwinięciu danej warstwy, otrzymujemy szczegółowe informacje o pakiecie (rys. 4.4).

Rysunek 4.4. Wireshark — analiza pakietu SIP.

Wygenerować możemy również statystyki dla danego protokołu, zawierające informacje o ilości pakietów w analizowanej transmisji danego typu (dla SIP będzie

to podział zgodnie z 1xx, 2xx, itd.). Dostępny jest również graf przebiegu rozmowy, który znacznie ułatwia zrozumienie protokołu, przedstawiając krok po kroku czynności jakie zostały poczynione w warstwie aplikacji (rys. 4.5).



Rysunek 4.5. Wireshark — graf oraz statystyki SIP.

4.2. Program autorski

Przeanalizowane powyżej aplikacje dają ogromne możliwości. W przypadku znacznej części z nich jednak wymagana jest znajomość odpowiednich przełączników, rozpoznanie w kwestii logowania zdarzeń do pliku, a nawet znajomość języka C. Najbliżej ideałowi w kwestii dydaktyki wydaje się być Wireshark, udostępniający multum opcji w formie przejrzystego GUI. Dedykowany jest on jednak głównie administratorom sieci czy pasjonatom, chcącym dogłębnie poznać procesy zachodzące w niższych warstwach. Stosowany jest również w procesie diagnostyki problemów sieciowych. Ogrom możliwości jakie daje Wireshark w procesie edukacji na temat protokołów VoIP może być przytłaczający i działać na niekorzyść. Postanowiono więc stworzyć własną aplikację, dedykowaną właśnie protokołom VoIP.

4.2.1. Zasada działania

Aplikacja stworzona została w wysokopoziomowym języku Python, co przekłada się na klarowność i łatwą rozszerzalność kodu. Jest również wieloplatformowa, więc przy zapewnieniu odpowiednich bibliotek działała będzie zarówno pod systemami uniksowymi, jak i Windowsami.

Kluczową część aplikacji stanowi moduł *sniff.py*, posiadający implementację zarówno sniffera, jak i parsera przechwyconych pakietów. Może być on włączany zarówno bezpośrednio (z konsoli), jak i pośrednio (z modułu *window.py*). W części dotyczącej sniffera, otwierany jest socket, a następnie przechwytywane są z niego dane. Dane te są dekodowane oraz dzielone za pomocą podstawowych narzędzi Pythona na zmienne reprezentujące takie wartości, jak adresy IP, TTL, czy nagłówek TCP. Po rozpakowaniu owego następuje dalsze przetwarzanie, tym razem dotyczące wyż-szych warstw. Całość wyświetlana jest w formie graficznej przez moduł *window.py*, bazujący na PyQt.

sniff.py

Moduł korzysta ze standardowych systemowych bibliotek, takich jak *socket*, *sys*, *struct*, *re* oraz *math*. Zbudowany jest z klasy *transmission* oraz funkcji *sniff*. Funkcja ta, przy pierwszym wywołaniu, tworzy obiekt klasy *tramsmission*, odpowiedni dla danego protokołu VoIP (obecnie zaimplementowany jedynie SIP). Przyjmuje ona jako argument protokół warstwy transportu (TCP lub UDP), a następnie za pomocą socketa przechwytuje pakiet:

```
transmission_protocols={'UDP': socket.IPPROTO_UDP, 'TCP': socket.IPPROTO_TCP}
if transmission_protocol not in transmission_protocols.keys():
    sys.exit()
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_RAW, \
    transmission_protocols[transmission_protocol])
except socket.error, msg:
    print 'Socket could not be created. Error code: ' + str(msg[0]) \
    + ' Message ' + msg[1]
    sys.exit()
packet = s.recvfrom(65565)
```

Funkcja socket, jako trzeci parametr przyjąć może następujące wartości: *IPPROTO_RAW*, *IPPROTO_ICMP*, *IPPOROTO_IGMP*, *IPPROTO_TCP*, *IPPROTO_UDP* i podobne. W przypadku protokołów VoIP będą to jedynie dwa ostatnie. Wartość ta definiowana jest ze słownika *transmission_protocols* z wartości przekazanej

przez użytkownika — w przypadku bezpośredniego użycia modułu, jako parametr, a w przypadku użycia *window.py*, wybierany zostaje z elementu typu *combobox*. Kolejnymi krokami jest rozpakowanie nagłówka IP oraz jego przetworzenie:

```
try:
    ip_header = unpack('!BBHHHBBH4s4s', ip_header_packed)
except:
    print 'unpacking error'
    return None

version_ihl = ip_header[0]
version = version_ihl >> 4
ihl = version_ihl & 0xF
ip_header_length = ihl * 4

ttl = ip_header[5] #time to leave
protocol = ip_header[6]
source_address = socket.inet_ntoa(ip_header[8])
destination_address = socket.inet_ntoa(ip_header[9])
```

Z nagłówka otrzymywane są kolejno TTL (ang. *Time To Live*), protokół, IP źródła oraz celu. Następnie, analogicznie, przetwarzany jest nagłówek TCP:

```
tcp_header_packed = packet[ip_header_length:ip_header_length+20]
try:
    tcp_header = unpack('!HHLLBBHHH', tcp_header_packed) #unpack it
except:
    print 'unpacking error'
    return None

source_port = tcp_header[0]
destination_port = tcp_header[1]
sequence = tcp_header[2]
acknowledgement = tcp_header[3]
thl = tcp_header[4]
tcp_header_lenght = thl >> 4

header_size = ip_header_length + tcp_header_lenght * 4
data_size = len(packet) - header_size
data = packet[header_size:]
```

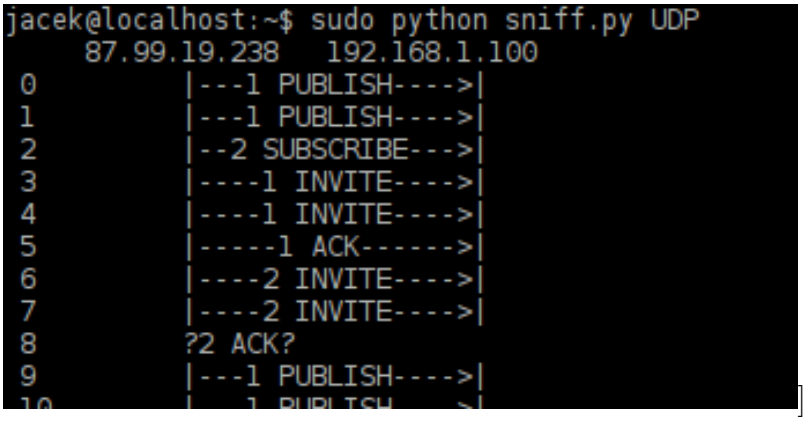
Jeśli w zmiennej *data* odnaleziona zostaje odpowiednia dla danego protokołu VoIP sekwencja, cały pakiet (w formie słownika z powyższymi danymi) przekazywany jest funkcji *analyse* odpowiedniej instancji klasy *transmission*.

Klasa *transmission* składa się z konstruktora, definiującego początkowe wartości zmiennych, oraz funkcji *analyse*, przyjmującej jako parametr pakiet wyłuskany z socketa przez funkcję *sniff*. Funkcja ta do listy słowników zawierających wszystkie przekazywane w danej transmisji pakietów dodaje obecny, a następnie go przetwarza. Z przekazywanej wiadomości wyłuskuje takie parametry jak *request* oraz konstruuje dwa typy ostatecznych, prezentowanych użytkownikowi form prezentacji zdarzeń sieciowych.

Graf Na bazie adresów IP (z rozróżnieniem na nadawców i adresatów) konstruowany jest graf zdarzeń sieciowych, dający użytkownikowi ogólne rozeznanie o zdarzeniach zachodzących podczas korzystania z danych usług VoIP. Adresy IP uczestniczące w konwersacji przechowywane są w liście *ips*. Przy każdym kolejnym pakiecie w liście tej sprawdzana jest obecność adresu źródłowego i docelowego, nieobecne tam są dopisywane i również na tej podstawie ustalana jest szerokość grafu czy pola go rozpoczynające i kończące:

```
for ip in (source_ip,destination_ip):
    if ip not in self.ips:
        self.ip_list=self.ip_list+ip.center(16, ' ')
        graph=' '+self.ip_list+'\n'
        self.ips.append(ip)
    request=str(cseq.group(1).strip() + ' ' + cseq.group(2).strip())
    source_ip_id=self.ips.index(source_ip)
    destination_ip_id=self.ips.index(destination_ip)
    if source_ip_id < destination_ip_id:
        delimeter_first = '|'
        delimeter_second = '>|'
    elif source_ip_id == destination_ip_id:
        delimeter_first = '?'
        delimeter_second = '?'
    else:
        delimeter_first = '|<'
        delimeter_second = '|'
    distance=int(fabs(int(destination_ip_id-source_ip_id)))
    graph+=str(self.package_number).center(3,' ')+8*' ' + delimeter_first + request.c
```

Do każdej z linii dodawany jest numer pakietu z danej transmisji, dzięki czemu przełożenie grafu na konkretne wiadomości staje się łatwiejsze. Przykładowy graf uzyskany w ten sposób przedstawiony jest na rys. 4.6.



Rysunek 4.6. Aplikacja w konsoli.

Wiadomość Zwracana jest również wiadomość w formie tekstowej, zawierająca wszystkie niezbędne dane do analizy pakietu VoIP. Zdefiniowana jest w następujący sposób:

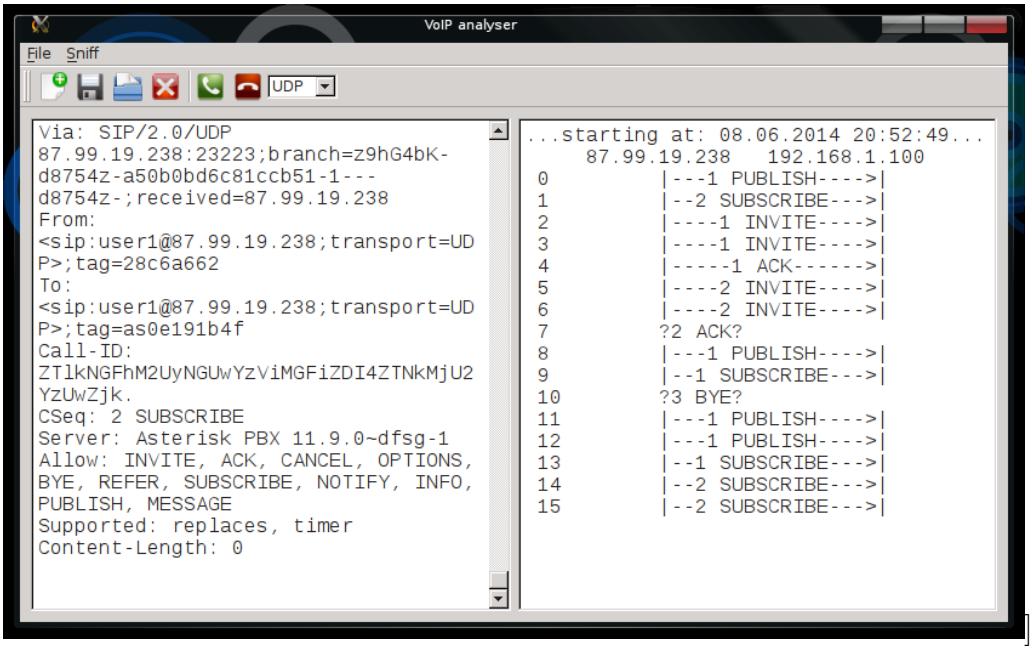
```
message=str(self.package_number).center(8,'-')+'\n'+ \
str(self.package[self.package_number]['data'])
```

Wiadomości prezentowane są w następującej formie:

```
---1---
sip:user1@127.0.0.1;transport=UDP SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:51662;branch=z9hG4bK-d8754z-48d7b3ace3e4811f-1---d8754z-
Max-Forwards: 70
Contact: <sip:user1@127.0.0.1:51662;transport=UDP>
To: <sip:user1@127.0.0.1;transport=UDP>
From: <sip:user1@127.0.0.1;transport=UDP>;tag=4935d21b
Call-ID: YjE1ZGMzYjc2Yzk2NTlkZjA0OTQyMDIxYzkwZTg4ODc.
CSeq: 1 PUBLISH
Expires: 600
Allow: INVITE, ACK, CANCEL, BYE, NOTIFY, REFER, MESSAGE, OPTIONS, INFO, SUBSCRIBE
Content-Type: application/pidf+xml
Supported: replaces, norefersub, extended-refer, timer, X-cisco-serviceuri
User-Agent: Z 3.2.21357 r21103
Event: presence
Allow-Events: presence, kpml
Content-Length: 285
```

Numer znajdujący się w pierwszej linii pozwala odnieść wiadomość do grafu, natomiast następnie zaprezentowana jest pełna wiadomość SIP.

Rys. 4.7 przedstawia główne okno aplikacji.



Rysunek 4.7. Główne okno aplikacji autorskiej.

sip.py

Na podstawie przykładów udostępnianych przez deweloperów modułów PJSIP[14], utworzono prosty symulator połączenia SIP. Moduł ten załączono do pracy, lecz w związku z koniecznością dopracowania, nie zaimplementowano w obecnej jej wersji. Jako parametr przyjmuje on URI i wykonuje na nie połączenie SIP, generując tym samym ruch sieciowy analizowany w tym samym czasie przez zaprojektowany sniffer. Dzięki rozwinięciu tego rozwiązania, uwzględniając możliwości jakie daje PJSIP, możliwym może być analiza protokołu SIP z jedynie niewielkim udziałem aplikacji czy urządzeń zewnętrznych.

Interfejs graficzny

Interfejs graficzny oparto o moduł PyQt4, z którego importowane są QtGui oraz QtCore. Oprócz tego wykorzystyane są *sys*, *os*, *time*, *thread* oraz omówiony powyżej *sniff*. Składa się z dwóch klas oraz kilku niezależnych od nich funkcji.

MainWindow Klasa ta dziedziczy po *QtGui.QMainWindow* i składa się z konstruktora oraz kilku funkcji odpowiadających za utworzenie głównego okna programu. Wewnątrz funkcji *windowInitialize* znajdują się kolejne, analogiczne do poniższego, instrukcje odpowiadające za przygotowania kolejnych interfejsów:

```
newAction = QtGui.QAction(QtGui.QIcon('img/new.png'), 'New', self)
newAction.setShortcut('Ctrl+N')
```

```
newAction.setStatusTip('New file')
newAction.triggered.connect(self.newFile)
```

Do akcji kolejno przypisywana jest ikona, opis, skrót, oraz wywoływana funkcja. Oprócz powyższych, zdefiniowany został *QComboBox* służący do wyboru między TCP a UDP. Wszystko umieszczone zostało na elementach typu *menuBar* oraz *toolBar*. Jako główny widget ustalony zostaje obiekt klasy MainWidget.

MainWidget Klasa, również dziedzicząca po *QtGui.QMainWindow*, definiuje layout programu. Składają się na niego dwa widgety — prawy i lewy, na którym kolejno przedstawiane będą wiadomości i grafy. Klasa zawiera funkcję *printText*, przyjmującą jako parametry identyfikator pola do którego ma zostać wpisany tekst ('left' lub 'right') oraz ów tekst.

Zdefiniowano również trzy funkcje, służące obsłudze pól tekstowych:

— *newFile* — funkcja czyszcząca oba pola.

— *saveFile* — zachowanie zawartości pól do plików tekstowych. Wiadomość (zawartość pola lewego) zapisana zostanie do zdefiniowanego przez użytkownika pliku, natomiast plik grafu (pole prawe) otrzyma sufiks ".schema".

— *openFile* — wczytywanie danych z plików o nazwach omówionych powyżej.

Obsługa sniffera W związku z omówioną wcześniej formułą przygotowanego sniffera, a więc funkcją, która po przechwyceniu jednego pakietu kończy działanie, koniecznym było uruchomienie jej w pętli. Pętla ta włączana zostaje jako osobny wątek i jest wykonywana, dopóki wartość zmiennej *doSniff* wynosi "1". Wartość ta zmieniana jest na "0" po użyciu przyciska przerywającego analizę. W pętli owej, za pomocą funkcji *printText*, zarówno wiadomość jak i graf, na bieżąco wyświetlany jest w głównym oknie programu. W czasie uruchamiania oraz zatrzymywania sniffera dodatkowo w obu oknach pojawia się informacja o zdarzeniu wraz z datą.

```
def startLongSniff():
    infoStartStop('starting')
    thread.start_new_thread(startSniff,())

def startSniff():
    global doSniff
    doSniff = 1
    while doSniff:
        sniff_results=sniff('UDP')
        if sniff_results:
            window.main_widget.printText('left', sniff_results['message'])
```

```
            window.main_widget.printText('right', sniff_results['graph'])

def stopSniff():
    infoStartStop('stopping')
    global doSniff
    doSniff = 0

def infoStartStop(state):
    for side in ('left','right'):
        window.main_widget.printText(side,'... '+state+' at: '+ \
        time.strftime("%d.%m.%Y %H:%M:%S")+' ... ')
```

Dalszy rozwój

W związku z modularnością, łatwą dostępnością gotowych rozwiązań, oraz ciągle rosnącą popularnością języka Python, aplikacja może być w łatwy sposób rozwinięta, aby w pełni zastąpić mogła inne rozwiązania podczas zajęć laboratoryjnych prezentujących protokoły VoIP. Podstawowym proponowanym rozszerzeniem byłoby wsparcie dla protokołu H.323, dla którego otwarte rozwiązania są znacznie gorzej dostępne w porównaniu z SIP, co utrudniło implementację w obecnej pracy. Implementacja owego jednak, przy dostępie do urządzeń generujących odpowiedni ruch sieciowy, sprowadzała się będzie do definicji kilku metod. Również zastosowanie modułu PJSIP wydaje się być sensownym rozwiązaniem. Dzięki niemu w prosty sposób zaimplementować może zostać symulator połączenia SIP.

4.3. Porównanie

Tabela 4.1 prezentuje porównanie podstawowych cech większej niż przedstawiona powyżej ilości aplikacji z dziedziny snifferów i okolicznych. W związku z ich mnogością oraz różnorodnością funkcji jakie zapewniają, dokonanie bardziej dogłębnej analizy nie mieściło by się w temacie obecnej pracy.

Tabela 4.1. Porównanie snifferów.

Aplikacja	GUI	Konsola	Unix	Windows
Capsa	+	-	-	+
CommView	+	-	-	+
EtherApe	+	-	-	-
ettercap	+ (GTK+/Ncurses)	+	+	+
justniffer	-	+	+	-
Kismet	-	+	+	+
netsniff-ng	-	+	+	+
ngrep	-	+	+	+
MS Network Monitor	+	-	-	+
OmniPeek	+	-	-	+
snort	-	+	+	+
tcpdump	-	+	+	+ (Windump)
Wireshark	+	+	+	+

5. Stanowisko laboratoryjne

Celem projektu jest przedstawienie koncepcji stanowiska do realizowania i analizy działania protokołów obsługujących usługi multimedialne. W związku z faktem, iż wybór urządzeń, szczególnie końcowych, nie wpływa znacząco na ruch sieciowy generowany przez dane protokołu, skupiono się właśnie na części stanowiska, która ten ruch będzie analizowała. Przygotowano również przykładową instrukcję do wykonania laboratorium (załącznik).

Niezależnie od wyboru urządzeń, w niższych warstwach, na których skupić się powinno w procesie edukacji, zachodzą te same procesy. Procesy te zdefiniowane są przez dane protokoły, których implementacją są zarówno telefony sprzętowe (ang. *hardphone*) jak i systemowe (ang. *softphone*). W związku z tym, w temacie wyboru tych pierwszych, zostawiona zostaje pełna dowolność. Jeśli chodzi o oprogramowanie, zaproponowane zostaną przykładowe, otwarte rozwiązania.

5.1. Serwer VoIP

Centralną częścią sieci będzie systemowy serwer VoIP, a wśród popularniejszych rozwiązań wskazać należy:

— 3CX Phone System,
— AS5300,
— AskoziaPBX,
— Asterisk PBX,
— Brekeke SIP Server,
— Brekeke PBX,
— CallMax Softswitch,
— Cisco Unified Communications Manager,
— ClearSea,
— CommuniGate Pro,
— Elastix,
— FreeSWITCH,
— GNU Gatekeeper,

— MediaCore Softswitch,
— Murmur,
— Mysipswitch,
— Objectworld UC Server,
— Yate.

W związku ze słusznie rosnącą w edukacji popularnością systemów rodziny Linux, bliżej omówiony zostanie jeden z popularniejszych serwerów VoIP działający właśnie pod tymi systemami. Asterisk[4], bo o nim mowa, jest kompletnym rozwiązaniem, dającym szereg możliwości. Jest to projekt o otwartym kodzie, udostępnianym na licencji GNU General Public Licence (GPL), co czyni go darmowym. Umożliwia to również użytkownikom rozszerzanie funkcjonalności, poprzez ładowalne moduły czy AGI (ang. *Asterisk Gateway Interface*). Oprócz Linuksa, wspierane są następujące systemy operacyjne: NetBSD, OpenBSD, FreeBSD, Mac OS X oraz Solaris.

Asterisk wspiera takie protkoły jak omówione uprzednio SIP, H.323, ale też MGCP, IAX, ADSI oraz SCCP. Współpracować może z większością popularnych urządzeń wykorzystujących te protkoły, działając jako PBX, czy też różnego rodzaju bramy, przykładowo do sieci PSTN czy ISDN. Protokół IAX2 (ang. *Inter-Asterisk eXchange*), zapewnia możliwość połączenia kilku instancji Asteriska w jedną, jeszcze wydajniejszą sieć. Przy użyciu odpowiednich rozszerzeń sprzętowych, wspierany jest również protokół sygnalizacyjny SS7.

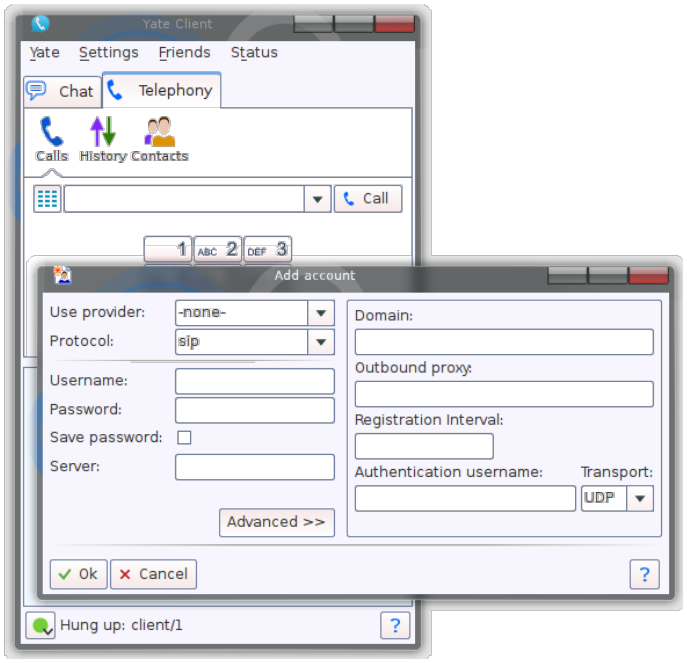
Oprogramowanie oferuje funkcjonalności, których na ogół oczekuje się od PBX: poczta głosowa, telekonferencje, interaktywna obsługa dzwoniącego (IVR), LCR, kolejkowanie czy nagrywanie rozmów. Współpracować może z bazami danych, takimi jak MyQSL czy PostgreSQL. Oprócz rozbudowanej możliwości konfiguracji poprzez pliki konfiguracyjne (domyślnie w */etc/asterisk/*), udostępnione jest również GUI, co w połączeniu z bogatą dokumentacją[15], czyni system prostym w użyciu.

5.2. Urządzenia końcowe

Różnorodność dostępnego oprogramowania typu *softphone* jest ogromna i znacznie wyższa niż wymienionych uprzednio serwerów. Oto część z darmowych rozwiązań[19]:
— Blink,
— Ekiga,
— Empathy,
— Jitsi,
— KPhone,

— Linphone,
— MicroSIP,
— PhoneGaim,
— QuteCom,
— SFLphone,
— Telephone,
— Twinkle,
— Yate.

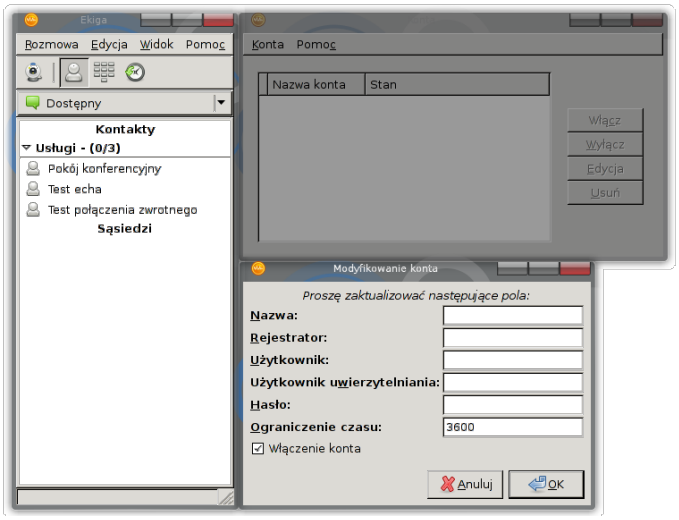
Przykładowym, proponowanym do użycia rozwiązaniem, jest Yate. Dostępny jest zarówno pod Windowsa jak i Linuksa, podobnie jak większość wymienionych aplikacji jest prosty w konfiguracji i użyciu poprzez GUI (rys. 5.1). Obsługuje protokoły takie jak SIP, H.323, MGCP czy IAX. Zaimplementowane ma wszystkie potrzebne do przeprowadzenia laboratorium funkcje.



Rysunek 5.1. Okno konfiguracji oraz klienckie aplikacji Yate.

Kolejnym z proponowanych programów jest Ekiga, równie przejrzysty i zarazem funkcjonalny (rys. 5.2). Również wspiera on wszystkie popularniejsze protokoły oraz funkcjonalności i kodeki.

Oprócz powyższego, stanowisko wykorzystać może również dostępne fizyczne urządzenia końcowe, podłączone do sieci przez Ethernet. Przykładem owych urządzeń jest seria Cisco Unified IP Phone[2]. Wśród serii tej znajdziemy zarówno proste urządzenia, które wykorzystane będą mogły być do ukazania działania podstaw



Rysunek 5.2. Okno konfiguracji oraz klienckie aplikacji Ekiga.

protokołu SIP, jak i te bardziej skomplikowane, dzięki którym podczas zajęć laboratoryjnych przeanalizować będzie można ruch sieciowy generowany przez te bardziej zaawansowane usługi.

Przedstawicielem pierwszej grupy jest Cisco Unified SIP Phone 3905 (rys. 5.3), który udostępnia jedynie podstawowe funkcje.

Rysunek 5.3. Telefon Cisco Unified SIP Phone 3905.

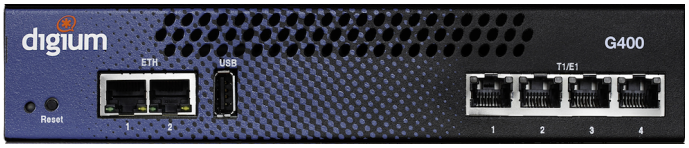
Spośród drugiej grupy ciekawym i zarazem ekonomicznym rozwiązaniem wydaje się Cisco Unified IP Phone 8945 (rys. 5.4), oferujący bardziej rozbudowane funkcje, z wideokonferencjami na czele.

Rysunek 5.4. Telefon Cisco Unified SIP Phone 8945.

5.3. Bramka

Aby ukazać jakże ważną funkcjonalność sieci VoIP, jaką jest możliwość integracji z innymi technologiami, w tym bardzo popularną do tej pory PSTN, konieczna będzie bramka. Może ona zostać rozwiązana zarówno bardziej po stronie oprogramowania, jak i kompletnie sprzętowo.

Jako rozwiązanie sprzętowe proponuje się urządzenia firmy Digium o symbolach G400 lub G800. Dają one możliwość kolejno podłączenia czterech lub ośmiu wejść T1/E1, konwertując otrzymany sygnał do postaci definiowanej przez SIP.



Rysunek 5.5. Bramka Digium G400.

Rozwiązaniem systemowym może być natomiast wspomniany wcześniej Asterisk, który współpracować może z kartami PCI czy PCI Express, rozszerzającymi interfejsy komputera o wejścia telefoniczne. Również tutaj firma Digium zapewnia szereg rozwiązań, począwszy od kart dających jedno wejście (rys. 5.6), po takie z czterema.

5.4. Analizator pakietów

Jako analizator pakietów SIP proponuje się program opracowany w ramach tej pracy. Najbardziej kompleksowym rozwiązaniem jednak nadal jest Wireshark, po-

Rysunek 5.6. Karta Digium TE131.

zwalający przede wszystkim na analizę innych protokołów. Jako że jest to dominujące narzędzie na rynku, warto zapoznać się również z jego obsługą.

6. Podsumowanie

Konwergentne usługi świadczone za pośrednictwem sieci pakietowych bez wątpienia będą się rozwijać. W celu świadczenia tego typu usługi multimedialnej, koniecznym jest dobór odpowiednich protokołów. Protokoły te dopasowane powinny być do oczekiwań zarówno providera, jak i klientów. Aby dokonać odpowiedniego wyboru, koniecznym jest dogłębne zapoznanie się z procedurami zachodzącymi w niższych warstwach, co uczyniono w powyższej pracy. Sam proces sniffowania i analizowania ruchu generowanego przez protokół podczas przeróżnych operacji wykonywanych przez aplikacje klienckie i serwerowe, może przekazać wiele informacji i umożliwić ujrzenie teorii w praktyce.

W pracy zaprezentowano koncepcję stanowiska laboratoryjnego, za pomocą którego owa wiedza na temat protokołów VoIP może zostać przekazana. Zaproponowano rozwiązania pozwalająca na analizę najpopularniejszy protokołów: SIP oraz H.323. Przygotowana aplikacja, służąca do wychwytywania zdarzeń sieciowych protokołu SIP, w połączeniu z programem Wireshark, posłużyć może za dopełnienie tego typu laboratorium. Aplikacja ta, przez wzgląd na zastosowany język i mnogość dostępnych bibliotek, a także modularną budowę, może zostać w łatwy sposób rozbudowana o kolejne funkcje.

Instrukcja laboratoryjna

Analiza protokołów VoIP: H.323, SIP

1. Zapoznać się z teorią odnośnie najpopularniejszych protokołów sieci VoIP: H.323 oraz SIP.
2. Zainstalować wybraną przez siebie aplikację serwerową (preferowany Asterisk).
3. Zapoznać się z możliwościami aplikacji oraz dokonać podstawowej konfiguracji.
4. Uruchomić urządzenia końcowe:
   — *softphone* — zgodnie z preferencjami, np. Yate,
   — *hardphone* — dostępne w laboratorium telefony.
5. Zapoznać się z możliwościami oraz dokonać podstawowej konfiguracji urządzeń końcowych.
6. Na jednym z komputerów biorących udział w transmisji uruchomić aplikację analizującą ruch sieciowy.
7. Wykonać różne scenariusze połączeń, wykorzystując różne z oferowanych przez VoIP usługi.
8. Za pomocą aplikacji analizującej ruch sieciowy zaobserwować zdarzenia zachodzące wewnątrz sieci i odnieść je do zdobytej uprzednio wiedzy teoretycznej.

Bibliografia

[1] Bromirski, M. *Telefonia VoIP. Multimedialne sieci IP.* 2006.

[2] Cisco. Cisco unified ip phones porfolio.

[3] Digium, I. Asterisk. http://www.asterisk.org.

[4] IETF. Extensible messaging and presence protocol (xmpp): Core. https://tools.ietf.org/html/rfc6120.

[5] IETF. Sip: Session initiation protocol. http://tools.ietf.org/html/rfc2543, 1999.

[6] IETF. Reliability of provisional responses in the session initiation protocol (sip). https://tools.ietf.org/html/rfc3262, 2002.

[7] IETF. Sip: Session initiation protocol. http://tools.ietf.org/html/rfc3261, 2002.

[8] ITU-T. Q.931: Isdn user-network interface layer 3 specification for basic call control. http://www.itu.int/rec/T-REC-Q.931/en/.

[9] ITU-T. Recommendation h.323 (11/1996). http://www.itu.int/rec/T-REC-H.323-199611-S/en, 1996.

[10] ITU-T. Frameworkservices recommendation for multimedia. www.itu.int/rec/T-REC-F.700-200011-I/en, 2000.

[11] ITU-T. Recommendation h.323 (12/2009). http://www.itu.int/rec/T-REC-H.323, 2009.

[12] J, D. *Voice over IP. Podstawy.* 2005.

[13] PJSIP. Pjsip hello world. http://trac.pjsip.org/repos/wiki/Python_SIP/Hello_World.

[14] PJSIP. Pjsip homepage. http://trac.pjsip.org/repos/wiki.

[15] Russell Bryant, Leif Madsen, J. V. M. Asterisk: The definitive guide. the future of telephony is now., 2013.

[16] Wallingford, T. *VoIP. Praktyczny prewodnik po telefonii internetowej.* 2007.

[17] Wikipedia. Comparison of packet analyzers. https://en.wikipedia.org/wiki/Comparison_of_packet_analyzers.

[18] Wikipedia. H.323. http://en.wikipedia.org/wiki/H.323.

[19] Wikipedia. List of sip software. http://en.wikipedia.org/wiki/List_of_SIP_software.

[20] Wikipedia. pcap. http://en.wikipedia.org/wiki/Pcap.

[21] Wikipedia. Voip calls on wireshark wiki. http://wiki.wireshark.org/VoIP_calls.