http トンネリングを利用した
SIP セッション確立方式の検討

提出日：2005年2月2日

指導：後藤滋樹教授

早稲田大学 理工学部情報学科
学籍番号：1G01P110-7

山田 大輔
目次

1 序論 .......................... 6
   1.1 研究の背景 .......................... 6
   1.2 研究の目的 .......................... 7
   1.3 本論文の構成 .......................... 8

2 SIP (Session Initiation Protocol) .......................... 9
   2.1 SIP の特長 .......................... 9
   2.2 SIP による通信の構成要素 .......................... 10
       2.2.1 ユーザエージェント .......................... 11
       2.2.2 SIP サーバ .......................... 11
       2.2.3 ブロキシサーバ .......................... 11
       2.2.4 リダイレクトサーバ .......................... 11
       2.2.5 登録サーバ .......................... 12
       2.2.6 ロケーションサーバ .......................... 12
   2.3 SIP メッセージ .......................... 12
       2.3.1 SIP URI .......................... 12
       2.3.2 メッセージの構成 .......................... 13
       2.3.3 スタートライン .......................... 13
       2.3.4 ヘッダフィールド .......................... 15
   2.4 各種プロトコルとの連携 .......................... 16
       2.4.1 SDP .......................... 17
       2.4.2 RTP/RTCP .......................... 17

3 HTTP (Hyper Text Transfer Protocol) .......................... 18
   3.1 HTTP の特長 .......................... 18
   3.2 HTTP による通信の構成要素 .......................... 18
       3.2.1 HTTP クライアント .......................... 19
3.2.2 HTTP サーバ ................................. 20
3.3 HTTP メッセージ ................................. 21
  3.3.1 HTTP URL .................................. 21
  3.3.2 メッセージの構成 ............................ 21
  3.3.3 スタートライン ............................... 22
  3.3.4 メッセージヘッダ ............................. 23
  3.3.5 HTTP ヘッダの必須項目 ........................ 24
3.4 HTTP トランザクションの例 .......................... 26
  3.4.1 GET メソッド ................................ 26
  3.4.2 POST メソッド ............................... 28

4 SIP を利用した通信での問題と解決手法 ............... 29
  4.1 SIP を利用した IP 電話における問題 .................. 29
    4.1.1 NAT 越え IP に関する技術 ....................... 29
  4.2 HTTP over SIP トンネリング ........................ 31
    4.2.1 HTTP トンネリング ............................ 32
    4.2.2 本研究の関連技術 ............................. 32
  4.3 本研究の提案方法 ............................... 33
    4.3.1 マシンの構成 ............................... 34
    4.3.2 各構成要素の機能仕様 ......................... 35

5 実装と動作検証 ..................................... 38
  5.1 HTTP over SIP Tunneling Server の実装 ............... 38
    5.1.1 実験の目的 ................................ 38
    5.1.2 実験の環境 ................................. 39
    5.1.3 プログラムの仕様 ............................. 39
    5.1.4 実装の環境 ................................ 40
  5.2 動作検証と性能評価 .............................. 40
    5.2.1 動作検証 ................................ 40
    5.2.2 性能評価 ................................ 41

6 本研究を用いた応用例 と 今後の課題 ................... 43
  6.1 本研究を用いた応用例 ............................ 43
    6.1.1 HTTP/SIP proxy Server を用いた応用例 ......... 45
  6.2 結論と課題 .................................. 46
6.2.1 結論 ................................. 46
6.2.2 今後の課題 ............................. 46
# 図一覧

2.1 SIP の構成要素 .................................................. 10

3.1 HTTP の構成要素 .................................................. 19

4.1 通信のマシン構成図 ............................................... 34

4.2 HTTP-Encapsulation Server 通信仕様 .......................... 35

4.3 HTTP カプセルングデータ構造 ................................... 36

4.4 HTTP-Uncapsulation Server 通信仕様 .......................... 37

5.1 HTTP over SIP Tunneling Server 実験環境 ...................... 39

6.1 HTTP/SIP proxy Server の機能概略図 .......................... 43

6.2 HTTP/SIP proxy Server の適用例 .............................. 45
### 表一覧

<table>
<thead>
<tr>
<th>章目</th>
<th>内容</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>SIP URI の一般形式</td>
<td>12</td>
</tr>
<tr>
<td>2.2</td>
<td>リクエストラインの一般形式</td>
<td>14</td>
</tr>
<tr>
<td>2.3</td>
<td>ステータスラインの一般形式</td>
<td>15</td>
</tr>
<tr>
<td>3.1</td>
<td>HTTP URL の一般形式</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>リクエストラインの一般形式</td>
<td>22</td>
</tr>
<tr>
<td>3.3</td>
<td>ステータスラインの一般形式</td>
<td>23</td>
</tr>
<tr>
<td>4.1</td>
<td>HTTP Header</td>
<td>36</td>
</tr>
<tr>
<td>5.1</td>
<td>HTTP over SIP Tunneling Server 実装環境</td>
<td>40</td>
</tr>
<tr>
<td>5.2</td>
<td>遅延時間測定結果</td>
<td>41</td>
</tr>
</tbody>
</table>
第1章

序論

1.1 研究の背景

近年，インターネットの接続形態は DSL や CATV，FTTH の普及により，誰でも手軽に高速な常時接続環境を持つことができるようになってきた。それに伴い，ISP や通信事業者などで様々なサービスが展開され，その一環として VoIP (Voice over IP) の技術を利用した IP 電話サービスが提供始めている。IP 電話は従来の固定電話とは異なり，利用者間で回線を占有しないために，通話による通信コストを下げることができるという大きな特長を持っている。また総務省において，2002 年 11 月から 050 から始まる 11 枚の IP 電話専用番号の配布が開始され，IP 電話サービスを提供するための基盤が作られた。このように，一般の家庭や小規模なオフィスでも IP 電話を利用できる条件が整ってきたことで，IP 電話は今後ますます普及していくものと期待されている。

IP 網内で従来の電話と同様なサービスを提供するためには，接続する端末間で適切なネゴシエーションを行ない，通信セッションを確立するためのプロトコルが必要になる。このセッション確立の手続き，つまり呼出御（シグナリング）を実現するためのプロトコルとして，既にいくつかのものが標準化され，製品にも応用されている。ITU-T（International Telecommunication Union - Telecommunication sector）により承認・勧告された H.323 や，IETF（Internet Engineering Task Force）により標準化された SIP（Session Initiation Protocol）がその代表的な例である。それらのシグナリングプロトコルの中でも特に，SIP は 1999 年 3 月に発表された比較的新しい規格で，転送機能，発信者番号通知機能など他のプロトコルに比べて公衆電話網に近い機能を備えるほか，HTTP（Hyper Text Transfer Protocol）をベースとした軽量かつシンプルな仕様となっており，さらに拡張性についても充分な検討がなされている。そのため SIP は IP 電話への応用をはじめ，IP 網を利用した様々な通信セッションの確立を実現するために広く利用されるようになってきている。しかし実際は，プロトコルと実ネットワークとの間に少なからず溝が存在する。この問題は，SIP に限らず IP 電話を支えるプロトコル全般に言えることである。
様々な研究機関や企業などで問題の解決に力を注いでいるが、いまだ SIP を利用した VoIP サービスにも数々の問題点が存在している。

1.2 研究の目的

インターネットとの高い親和性や仕様の容易さ、拡張性の高さなどにより今後さらに普及・発展していくと考えられる SIP だが、前述の通りそこにはいくつかの問題点も存在する。例えば、IPv4 Private/Global 間や IPv4/v6 間のようにアドレス体系の異なるネットワーク間で通信を行なう場合に、SIP を利用した通信では従来の NAT (Network Address [Port] Translation) の仕組みがそのまま利用できない。また現在、ほとんどの企業ネットワークにはファイアウォールや HTTP-proxy サーバーなどが存在し、そのままでは IP 電話サービスを利用することができない。しかし IP 電話には、なるべくどのような状況でも使える柔軟性を持たせるのが好ましい。そのため、このような問題を解決するためのいわゆる NAT 越え Uniform に関する研究やファイアウォールを越えるための研究が、様々な場所で行なわれている。また、より柔軟性をもたせることができる目的と共に、セキュリティを確保したり管理を容易にするなど、SIP を用いた IP 電話サービスには様々な試みが行われている。特に、セキュリティは近年のインターネット全体の問題でもあり、IP 電話サービスの大きな顧客先となる企業など、機密事項が直接利益に関係してくる場所においてセキュリティの確保は大変重要な問題である。例えばファイアウォールや HTTP-proxy サーバーが存在するネットワークで IP 電話サービスを利用しようとすると、ファイアウォールのルールを改善する。proxy サーバを通らないルートを作ることで留まるなどの対策を施さなくてはならない。しかしこのことは、管理人の負担が増えてしまうだけでなく、コストが高くなりファイアウォールのルール設定をミスしてしまう危険を伴うなどの問題点が存在する。

そこで本論文では、柔軟性とセキュリティの観念に基づき、SIP による VoIP セッションのデータを http プロトコルを用いてシリアル化し転送することで、最低限の機能しか持たない IP 電話でもファイアウォールや proxy サーバーを越えた通信を確立できるようにする手法について提案する。
1.3 本論文の構成

本論文は以下の章により構成される。

第1章 序論

本研究の概要について述べる。

第2章 SIP (Session Initiation Protocol)

SIP の概要とその特長を述べる。また、IP 電話サービスを提供する際に SIP と併用して用いるプロトコルである SDP，および RTP/RTCP について述べる。

第3章 HTTP (Hyper Text Transfer Protocol)

HTTP の概要について述べる。HTTP の仕様について簡単に述べ、HTTP を用いたデータ受け渡しの手順について、簡単な例を用いて述べる。

第4章 SIP を利用した通信での問題と解決手法

SIP を利用した通信での問題点について述べる。具体的には、アドレス体系の異なるネットワーク間や、ファイアウォールが存在するネットワーク内での制限について述べる。そしてこの問題のいくつかの解決手法について取り上げた後、本研究の解決手法について述べる。

第5章 実装と動作検証

本研究の検討内容についてその一部を実装し、実際に SIP フォンを用いて実験を行う。動作検証では、SIP フォン間でシグナリングが成功し、音声がやり取りできるかを検証する。

第6章 結論

本論文のまとめと今後の課題を述べ、本研究の技術を用いた実際のネットワークでの応用例について述べる。
第2章

SIP (Session Initiation Protocol)

2.1 SIPの特長

SIP (Session Initiation Protocol) は、双方向型セッションの開始、変更、終了を行うための呼制御（シグナリング）を実現するプロトコルである。インターネット技術の国際的な標準化組織であるIETF（Internet Engineering Task Force）によって1999年3月にRFC2543として策定され、その後2002年6月にRFC3261として改版されている。さらにドラフトやその他のRFCによって拡張が検討、策定され、現在でもSIPはより実用性の高いプロトコルを目指して急速な発展を続けている。また、SIPの他にもIP網上でのシグナリングプロトコルは複数のものが策定されている。中でもITU-T（International Telecommunication Union - Telecommunication sector）によって承認、勧告されたH.323は、策定時期が早かったことなどの理由から現在までに数多くの適用実績がある。今日H.323と肩を並べるまでに台頭を果たし、今後さらに主流となりつつあるSIPには、H.323にはない次のような特長がある。

インターネットとの親和性が高い

H.323が従来の公衆網におけるシグナリングの手順をもとに考え出されたものであるのに対し、SIPはIP網上での動作を基本として考え出されたプロトコルである。その動作は、Webコンテンツの転送に使われるHTTP（Hyper Text Transfer Protocol）や、メールの送信に使われるSMTP（Simple Mail Transfer Protocol）といったプロトコルの動作を基本としていて、インターネットで利用される様々なプロトコルとの親和性が非常に高いものとなっている。

軽量で拡張性が高い

H.323は一般的電話モデルとしているため、IP網上でのサービス実現にはプロトコルが複雑となり拡張性も欠ける。それに対しSIPは、シグナリングの手順をはじめ仕様がシンプルに定義されており軽量である。また、他のプロトコルと組み合わせることによる機能
追加が容易で、拡張性も高い。

テキストベースで管理が容易である

SIP はテキストベースで記述されるプロトコルである。そのためバイナリで記述される H.323 と比較すると、開発や運用保守に際して特別な機器を用意する必要がなく扱いやすい。

このような特長を持つ SIP は、IP 電話だけでなく、ビデオ会議やチャットなどにも用いられるようになり、急速に普及が進んでいる。しかし、普及の速度に対して拡張仕様に関する標準化が遅れており、SIP プロトコルの課題となっている。

2.2 SIP による通信の構成要素

SIP はクライアント/サーバ型と呼ばれるトランザクションモデルを採用している。つまり、SIP による通信はクライアントからリクエストが送出され、それに対するレスポンスがサーバから返されるという形式で成り立っている。しかし SIP 全体ではピアツピア型の通信を行なうため、单一のセッションの間でもエンドポイントの役割が変化するのが普通である。つまり通信の内容に応じてクライアント（リクエストを送る側）/サーバ（レスポンスを返す側）の両方になりえる。SIP による通信の構成要素は大きく、ユーザエージェント、SIP サーバ、ロケーションサーバの 3 つに分けることができる。エンドポイントとなるユーザエージェント間で、まず各種サーバを経由した SIP によるシグナリングが行われる。セッション確立後は他のプロトコルを用いて音声や動画像などメディアの交換を直接行ない、最後にセッション確立時と同様に SIP によるシグナリングを行なってセッションを終了する。

図 2.1: SIP の構成要素
2.2.1 ユーザージェント

SIP におけるエンドポイント、つまり電話網における電話機に相当する部分をユーザージェント (UA: User Agent) と呼ぶ。UA はセッションを確立するユーザのフロントエンドとして SIP メッセージやメディアの送受信を行なう。

UA は、ユーザージェントクライエント (UAC: User Agent Client) とユーザージェントサーバ (UAS: User Agent Server) という 2 つの機能モジュールに分けることができる。UAC はリクエストを開始する機能モジュールで、UAS は受け取ったリクエストに対するレスポンスを開始する機能モジュールである。SIP の各 UA には必ず UAC と UAS、両方の機能が含まれており、単一のシグナリングの中でどちらの機能も利用されるのが普通である。

UA の実装例としては、SIP 電話端末、パソコンや PDA 上で動作するソフトウェア (softphone) が挙げられる。

2.2.2 SIP サーバ

SIP サーバは、UA 間で行なわれるシグナリングの経路上に位置し、セッションの確立を仲介する処理を行う。SIP サーバはプロキシサーバ、リダイレクトサーバ、登録サーバの 3 つの役割に分けられる。これらの 3 つの役割は、一般には同じホスト上で動作するが、それぞれ単体のソフトウェアであるため、負荷分散のためにそれぞれ別々のサーバとして機能させることも可能である。

2.2.3 プロキシサーバ

プロキシサーバ (Proxy Server) は、UA の代理として SIP メッセージを中継する。プロキシサーバは、SIP メッセージを受け取るとロケーションサーバに問い合わせを行なうことで宛先を判断する。そしてリクエストメッセージを UAC へ、レスポンスを UAS へ転送する。レスポンスは Via ヘッダフィールドという SIP メッセージのシグナリングパスの記録を用い、プロキシサーバを経由する限り、リクエストと逆の経路を巡る。実際には、ネットワークの構成に応じて複数のプロキシサーバを経由することもある。

2.2.4 リダイレクトサーバ

リダイレクトサーバ (Redirect Server) は、通信相手のアドレスが変更されていた場合に、新しいアドレスをロケーションサーバに問い合わせて UA やプロキシサーバに送りなおすべき宛先が記述されたリダイレクトレスポンスを返す。リダイレクトサーバは SIP メッセージを転送することはない。
2.2.5 登録サーバ

登録サーバ (Register Server) は、UA からの要求を受けて、ロケーションサーバに対して UA の新規登録処理や更新処理、削除処理などを行なう。

2.2.6 ロケーションサーバ

ロケーションサーバ (Location Server) は、登録サーバによって登録される UA の情報を保持し、その情報をプロキシサーバやリダイレクトサーバによって利用できるデータベースとして提供する。SIP ではロケーションサーバへのアクセス方法を規定していないため、レコードに対する操作には別のプロトコルが利用される。

2.3 SIP メッセージ

SIP によるシグナリングは、UA 間で SIP メッセージをやりとりすることによって行なう。この章では、SIP メッセージの中から本論文で利用されるメッセージについておおまかな説明を行う。

2.3.1 SIP URI

SIP による通信では、リソースの特定に URI (Uniform Redource Identifier) を用いる。SIP で新たに定義された URI には SIP URI と SIPS URI の 2 つがあり、この他にも tel URI を用いることもできる。SIPS URI は SIP URI に暗号化が施されたセキュアなバージョンである。

SIP のトランスポートプロトコルには TCP と UDP が必須とされているが、現実にはプロキシサーバのようにシグナリングパス上に位置するサーバが長時間存続するコネクションを維持することが困難な場合があるため、実際には多くの場合に UDP が利用される。このような理由から、本論文でも SIP のトランスポートプロトコルは UDP を用いるものとして扱う。

次に、SIP-URI の一般形式を示す。

<table>
<thead>
<tr>
<th>表 2.1: SIP URI の一般形式</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip: user:password@host:port;uri-parameters?headers</td>
</tr>
</tbody>
</table>

下線で示した部分は必ず指定しなくてはならない。各要素の意味は次の通りである。

user

宛先となるホストでのユーザ ID。ユーザ ID を指定しない場合は、@までを省略することができる。宛先のホストが電話番号を扱える場合には、ここに電話番号を指定する。
password
ユーザ ID に関連付けられたパスワード。ただし、認証情報を平文で記述することになるため、RFC3261 では使用を推奨していない。

host
SIP リソースを提供するホスト。FQDN、IP アドレスのいずれでも指定することができるが、FQDN を利用することが推奨される。

port
リクエストの宛先ポート番号。SIP では 5060 が、SIPS では 5061 がデフォルトで利用される。

uri-parameters
URI パラメータ。 □ < パラメータ名 >=< パラメータ値 > □ の形式で各組を □ ； □ で区切ることで複数指定することができる。パラメータ名には transport、user、method、maddr、ttl、lr などがあり、独自に追加定義することも可能である。

headers
リクエストのヘッダフィールド。 □ < ヘッダ名 >=< 値 > □ の形式で各組を □ & □ で区切ることで複数指定することができる。特別なヘッダ名として body が定義されており、この値はメッセージのボディとなる。

2.3.2 メッセージの構成

SIP はテキストベースのプロトコルであり、文字コードセットとして UTF-8 (Universal Transformation Format, 8-bit Form) を利用する。UTF-8 は、Unicode または USC (Universal Character Set) で符号化された文字をインターネットで転送する際に利用される方法である。

SIP メッセージは、クライアントからサーバへのリクエストメッセージか、サーバからクライアントへのレスポンスかのいずれかである。これらは、RFC2822 で定義された基本フォーマットを用いて記述され、スタートライン、ヘッダフィールド、ヘッダフィールドの終了を表す空白行、およびオプションのポディからなる。また、メッセージ内の各行は CRLF で区切られる。

2.3.3 スタートライン

リクエストライン

リクエストメッセージにおいて、スタートラインは特にリクエストラインと呼ばれる。リクエストラインは、メソッド名 (Method)、リクエスト URI (Request-URI)、プロトコルバージョン
第2章 SIP (SESSION INITIATION PROTOCOL)

(SIP-Version)からなりCRLFで終了する。また、それぞれが1つの空白文字(SP)で区切られる。

次に、リクエストラインの一般形式を示す。

| Request-Line = Method SP Request-URI SP SIP-Version CRLF |

Method
リクエストの種別を表す。RFC3261ではUAのコンタクト情報を登録するためのREGISTER、セッション確立のためのINVITE、ACK、CANCEL、セッションを終了するためのBYE、およびサーバに能力を問い合わせるためのOPTIONSの6種類のメソッドが定義されている。また、これらのメソッドに加えて付随するその他のRFCでは拡張メソッドも定義されている。

Request-URI
リクエストメッセージの宛先で、SIP URI、SIPS URIの形で記述される。

SIP-Version
メッセージの記述に使われるプロトコルのバージョンを表す。RFC3261に準拠する場合は、SIP/2.0となる。

ステータスライン
レスポンスでは、スタートラインをステータスラインと呼ぶ。ステータスラインは、プロトコルバージョン(SIP-Version)、ステータスコード(Status-Code)、および関連付けられたフレーズ(Reason-Phrase)からなり、リクエストラインと同様にそれぞれが1つの空白文字で区切られる。
第2章 SIP (SESSION INITIATION PROTOCOL)

次に、ステータスラインの一般形式を示す。

表2.3 ステータスラインの一般形式

| Status-Line = SIP-Version SP Status-Code SP Reason-Phrase CRLF |

SIP-Version

メッセージの記述に利用するプロトコルのバージョンを表す。

Status-Code

リクエストに対する結果を表すコードで、100番台から600番台までの3桁の整数からなる。100番台が処理継続中、200番台がリクエストの成功、300番台がリダイレクト処理、400番台がクライアントエラー、500番台がサーバエラー、600番台がグローバルエラーと定義づけられている。特に100番台はプロビジョナル（暫定）レスポンス、200〜600番台はファイナル（最終）レスポンスと2つに分類される。

Reason-Phrase

テキストからなる、リクエストに対する結果についての短い説明。Status-Codeがオートマトンによる使用を意図しているのに対して、Reason-Phraseは人間のユーザによる使用を意図したものである。

2.3.4 ヘッダフィールド

SIPメッセージは、スタートラインに続いて「<ヘッダ名> : <ヘッダ値>」の形式で複数のヘッダフィールドが続く。数多く定義されたヘッダ名のうち、To、From、Via、Max-Forwards、CSeq、Call-IDの6つはすべてのリクエストメッセージに必須の項目である。

To

リクエストの受信者を指定するフィールド。オプションの表示名（受信者名）は、ユーザインターフェイスによって表示される。

From

リクエストのイニシエータ（生成元）を示すフィールド。Toヘッダフィールドと同様に表示名を指定するオプションがある。ここでクライアントの身元が隠されている場合には、表示名として「Anonymous」を使用するべきとされている。
第2章 SIP (SESSION INITIATION PROTOCOL)

Via

Viaヘッダフィールドには、リクエストが経由したパスのマシン上で、ログ情報としてマシンの情報が追加されていく。これはレスポンスが返されるときにちょうど逆向きに辿るべきパスとなる。また，z9hG4bK (マジッククッキー) で始まる branch パラメータは、トランザクション ID としてループ検知のためにプロキシサーバによって利用される。

Max-Forwards

リクエストをダウンストリームサーバに転送できるプロキシサーバの数を制限するために利用されるフィールド。値はリクエストが転送されることを認められている残り回数を示す0～255の整数で、リクエストを転送する各プロキシサーバ上でデクリメントされる。推奨値は70。

CSeq

10進数のシーケンス番号とリクエストのメソッド名の組み合わせて構成される値で、トランザクションの順序付けを行う。シーケンス番号は32ビットの符号なし整数で表現され、原則として新しいリクエストごとにインクリメントされる。例外的にACKは確認応答の対象となるINVITEリクエストと同値、CANCELはキャンセル対象のリクエストと同値を用いる。

Call-ID

特定の招待または特定のクライアントの全ての登録を一意に識別するフィールド。Call-IDの値はダイアログの構築に利用される。

Content-Type

受信者に送られたメッセージボディのメディアタイプを示すフィールド。メディアタイプの要素はIANA (Internet Assigned Numbers Authority)によって登録されている。このContent-Typeヘッダフィールドは、ボディが空でない場合には必ず存在しなければならない。

2.4 各種プロトコルとの連携

SIP自体の基本的な機能は、IPネットワークにおける音声や動画像などのメディアをやりとりするセッションの、接続や切断を制御するシグナリングプロトコルであり、音声伝送などの機能は含まない。そのため、SIPを利用したメディアの通信ではSIP以外にも複数のプロトコルが連携して動作するのが普通である。ここではSIPを利用したメディアの通信において、セッ
第 2 章 SIP (SESSION INITIATION PROTOCOL)

シオンのセットアップに利用される SDP、およびメディアを送受信する際に利用される
RTP/RTCP について触れることで。

2.4.1 SDP

SDP (Session Description Protocol) は、セッションを記述するための情報記述プロトコルで
あり、IETF の RFC2327 で規定されている。SIP メッセージのボディ部の記述に使われ、詳細な
セッション制御を行うことを可能にしている。SDP は、セッション記述、時間記述、メディア記述からなり、それぞれ
< タイプ名 >=< 情報 > の形式で記述される。ここにはセッションを識別するための情報、有効期限などに関する情報、アドレスやポート番号、メディアの種類に関する情報を記述することができる。

2.4.2 RTP/RTCP

RTP (Real-time Transport Protocol) は、音声や動画像のような実時間性 (リアルタイム性) が
要求されるデータを利用して多人数マルチメディア会議のようなサービスをインターネットで実現することを目的に設計されたプロトコルである。リアルタイムメディアの転送トランスポートプロトコルには通常、転送効率を重視して UDP が利用される。しかし UDP には転送途中でのパケット消失やデータ順序の入れ替わりなどに対応するすべがない。そこで RTP と、RTP を制御する RTCP (Real-time Transport Control Protocol) というプロトコルを用いることでメディアの転送品質の向上や送受信端末間でのクロック同期などを行なう。SIP とは異なり、RTP ではプロトコルの情報記述がバイナリで行なわれる。
第３章

HTTP (Hyper Text Transfer Protocol)

3.1 HTTP の特長

HTTP は、ハイパーエクディア情報配布システムのためのアプリケーションレベルプロトコルである。1990 年に、インターネット上で未加工のデータをやり取りするための HTTP/0.9 というバージョンが存在したのが始まりである。その後 RFC1945 で定義された HTTP/1.0 では、プロトコルの改善が行われ、効率化のために転送データの情報を記述し、通信の形態をリクエスト/レスポンスの形とした。そして現在、RFC2068 によりブロキシや永続接続、仮想ホストなどの効果の高い技術を熟慮した HTTP/1.1 というバージョンが定義された。HTTP は、インターネットの普及に大きく貢献した World Wide Web を支える重要な技術である。現在のインターネットの発展は、HTTP によるものが非常に大きい。

現在のインターネットにおいて、HTTP はなくてはならないプロトコルである。セキュリティを重視した企業内のネットワークからも、HTTP はなんらかの方法で外部への接続が許可されていることが多い。その方法は、ファイアウォールが HTTP を許可する、外部へ代理で接続するプロキシサーバーを介するなど様々である。HTTP は、複数のサーバ上でまとまりなく別々の場所に保管されたドキュメントを移動させることなく互いに結び付けあう、ハイパーリンクという技術を基盤としている。HTTP による通信において行われていることは、マシンから他のマシンへのデータ転送の繰り返しである。これらの特長から、HTTP に他のプロトコルを載せて転送するトンネル技術が近年急速に発展してきている。この章では、HTTP/1.1 のプロトコルの概要とデータ転送の仕組みについて触れる。

3.2 HTTP による通信の構成要素

HTTP は、クライアント/サーバ型と呼ばれるトランザクションモデルを採用している。しかし、第 2 章で取り上げた SIP とは異なり、通常クライアントとサーバは 1 つのトランザクション
第 3 章 HTTP (HYPER TEXT TRANSFER PROTOCOL)

内で入れ替わることはない。リクエストを開始するクライアントのことを特にユーザエージェン
ト (UA: User Agent) と呼ぶ。HTTP では、UA がサーバにリクエストを送り、そのリクエスト
に対するレスポンスをサーバが返すというリクエスト/レスポンス交換の形で通信を行なう。ま
た、トランスポートプロトコルには常に TCP が用いられる。

HTTP/1.0 においては、個々のリクエスト/レスポンス交換のたびに新しい接続を使用するよ
うになっていたが、HTTP/1.1 では、1 回の接続を複数のリクエスト/レスポンス交換に使用
できるようにデフォルトの動作が変更された。これは、Web ページを参照する際に Web ページ
と共にいくつかの画像を取得することが増えたため、つまり、1 回の Web ページの参照に複数回の
リクエスト/レスポンス交換が必要となるケースが増えたため、接続の仕様を効率化する必要が
あったためである。デフォルトのポートは 80 であるが、他のポートを選択することもできる。
HTTP は確実な転送のみを前提としており、そのような保証を供給するどのようなプロトコルで
も使用し、プロトコルの最上部として実装することができる。

この章では HTTP の通信における一般的な構成要素について触れる。また、HTTP の通信に
おける重要な要素であるいくつかの代表的なサーバについても触れる。

3.2.1 HTTP クライアント

HTTP におけるクライアントは、HTTP サーバに対してリクエストを送信する目的のために
接続を確立するプログラムである。実装例としては、Internet Explorer や Netscape, Opera と
いったいわゆるブラウザ、そして Web を渡って情報の収集を行なうスパイダー、その他エディ

図 3.1: HTTP の構成要素
タやエンドユーザツールなど多岐にわたる。クライアントは、HTTP メッセージを送ることに
よってリクエストをサーバに送出する。受け取ったデータの処理方法はプログラムの目的によっ
て様々である。

3.2.2 HTTP サーバ

HTTP におけるサーバは、レスポンスを送り返して要求を実行する目的で接続を受け入れる
アプリケーションプログラムである。また、1つのプログラムでクライアントとサーバの両方の
機能を持つことができる。HTTP におけるクライアント、サーバとは 1つのトランザクション
におけるプログラムの役割を指すものであり、プログラムの機能を指すものではない。HTTP
サーバには、オリジンサーバ、プロキシ、ゲートウェイ、トンネルなどの役割が存在する。もちろん、1つのプログラムがこれらの中から複数の役割を担うことも可能である。この代表的な 4
つのサーバについて説明する。

オリジンサーバ

オリジンサーバとは、与えられたリソースが存在もしくは制作されるサーバのことを指す。

プロキシ

プロキシ (proxy) サーバ。他のクライアントに代わってリクエストを作成する。目的のサーバ
にクライアントの情報を渡さないため、認証を行なって内部からの不正なユーザによる接続を許
可しないためなど様々な目的で立てられる。プロキシサーバにはクライアントとサーバの両方の
機能が備わっていなければならない。また、クライアントからプロキシに送られるリクエストで
は、リソースの識別に相対 URL を使うことは許されず、絶対 URL を指定しなくてはならない。

ゲートウェイ

プロキシと同じように別のサーバへの中継をするが、リクエストのリソースのオリジンサーバ
であるかのように振舞う。クライアントはゲートウェイと通信していることを意識する必要はな
い。

トンネル

2つの接続間で機械的な中継をする中間プログラム。トンネルによる通信が始まるとき HTTP
通信の当事者とはみなされない。
第3章 HTTP (HYPER TEXT TRANSFER PROTOCOL)

3.3 HTTP メッセージ

HTTP によるリクエスト/レスポンス交換は、HTTP メッセージを用いたクライアントからサーバへの要求と、サーバからクライアントへの応答で成り立っている。この章では HTTP メッセージの中から一般的なものについて説明する。

3.3.1 HTTP URL

HTTP URL (Universal Resource Locator) は、HTTP プロトコル経由でネットワークリソースの位置を定めるために使用される。

次に、HTTP URL の一般形式を示す。

![表 3.1: HTTP URL の一般形式](http:// host :port abs_path)

下線部で示した部分を省略することはできない。

host

HTTP リソースを提供するホスト。FQDN、IP アドレスのいずれでも指定することができるが、FQDN を利用することが推奨される。

port

リクエストの宛先ポート番号。省略されていればポート 80 が仮定される。

abs_path

abs_path とはリソースに対する Request-URI、つまりサーバ上にあるリソースを特定するための値のことを指す。また、省略されるときは / として与えられなければならない。host、abs_path において大文字と小文字を区別しない。予約された文字や危険な文字 (UNIX における' ~ 'など) はそれらの' % 'HEX HEX とエンコーディングと等しい。

(例：' ~ ' = 0 %7e 0)

3.3.2 メッセージの構成

HTTP はテキストベースのプロトコルであり、文字コードセットには US-ASCII が用いられる。HTTP メッセージの構成はスタートライン、メッセージヘッダ、メッセージヘッダの終了を表す空白行、およびメッセージボディとなる。また改行には CRLF が用いられる。
3.3.3 スタートライン

リクエストライン

リクエストメッセージにおいて，スタートラインは特にリクエストラインと呼ばれる．リクエストラインは，メソッド名 (Method) ，リクエスト URI (Request-URI) ，プロトコルバージョン (HTTP-Version) からなり，それぞれが1つの空白文字 (SP) で区切られ，CRLF で終了する．次に，リクエストラインの一般形式を示す．

表 3.2：リクエストラインの一般形式

| Request-Line = Method SP Request-URI SP HTTP-Version CRLF |

Method

Method は，Request-URI により識別されるリソースに行われるためのメソッドを示す．メソッドにはデータを要求する GET ，データをサーバへ送信する POST ，その他 HEAD (ヘッダ情報の取得) ，PUT (URL 先にエンティティボディの内容を格納) ，LINK ，UNLINK ，DELETE (URL を削除) が HTTP/1.0 で指定でき，HTTP/1.1 ではそれらに加えて OPTIONS (許されるオプションの情報を得る) ，TRACE (リクエストの伝送経路追跡) の2種類のメソッドを指定することができる．

Request-URI

リクエストを適用するリソースを識別するための情報．絶対パスと abs_path のどちらでも指定できる．サーバの自身へのリクエストは，ここに * を指定する．

HTTP-Version

HTTP のバージョン．RFC2068 に準拠する場合は，HTTP/1.1 と記述する．
ステータスライン

レスポンスでは、スタートラインをステータスラインと呼ぶ。ステータスラインは、プロトコルバージョン (HTTP-Version)、ステータスコード (Status-Code)、および関連付けられたフレーズ (Reason-Phrase) からなり、リクエストラインと同様にそれぞれが１つの空白文字 (SP) で区切られ、CRLF で終了する。
次に、ステータスラインの一般形式を示す。

表 3.3: ステータスラインの一般形式

| Status-Line = SIP-Version SP Status-Code SP Reason-Phrase CRLF |

HTTP-Version
メッセージの記述に利用するプロトコルのバージョンを表す。

Status-Code
リクエストに対する結果を表すコードで、100 番台から 500 番台までの 3 桁の整数からなる。100 番台が処理継続中、200 番台がリクエストの成功、300 番台がリダイレクト処理、400 番台がクライアントエラー、500 番台がサーバーのエラーと定義づけられている。

Reason-Phrase
テキストからなるリクエストに対する結果についての短い説明。人間のユーザによる使用を意図したものである。

3.3.4 メッセージヘッダ
HTTP メッセージは、スタートラインに続いて「< ヘッダ名 >: < ヘッダ値 >
の形式で複数のメッセージヘッダが続く。メッセージヘッダは、汎用ヘッダ、リクエストヘッダ (レスポンスヘッダ)、エンティティヘッダの 3 部で構成される。
異なるフィールド名を持つヘッダフィールドの受信される順序は重要性を持たないが、汎用ヘッダを最初に配置し、リクエストヘッダに続いて最後にエンティティヘッダを持つのが良い慣例とされている。

汎用ヘッダ
リクエストとレスポンスメッセージの両方のための一般的な適用性を持つヘッダフィールドの中で、エンティティ (メッセージボディ) には適用されないもの。明示的に接続を保つよう指定するために用いられる Connection などがこれにあたる。
リクエストヘッダ

クライアントが、リクエストやクライアント自身の情報をサーバに渡すためのヘッダ。受信処理が可能な言語やデータの種類を指定するヘッダ、仮想ホストを指定するための Host ヘッダなどがこれにあたる。HTTP/1.1 では仮想ホストの実現のために、リクエストにより指定されたリソースを Request-URI と Host ヘッダの両方で検査することを決めており、Host ヘッダは全てのリクエストヘッダに必須なものとされている。

レスポンスヘッダ

サーバが、ステータスラインに置けなかったレスポンスに関する追加的な情報をクライアントに渡すためのヘッダである。プロキシの認証機構やサーバプログラム情報の通知、さらにはクッキーというクライアント側の情報保存ファイルへの書き込みを指示する Set-Cookie ヘッダが含まれる。

エンティティヘッダ

ボディ部の情報を渡すためのヘッダ。主に、GET に対するレスポンスや POST リクエストなど、メッセージボディの存在するメッセージに付加される。エンコード方式やメッセージボディの長さ、メッセージボディのメディアタイプについて記述する。メディアタイプは、メッセージボディが空でない全てのメッセージに付けるべきだとしている。もし存在しない場合は application/octet-stream として扱われる。メディアタイプは SIP のものと同様 IANA で管理されており、そこに定義されていないタイプの使用は避けることが望ましい。

3.3.5 HTTP ヘッダの必須項目

ここまで HTTP のメッセージについて説明してきたが、メッセージヘッダは様々な項目がある。この中でどの項目を選択してメッセージに付加するかはプログラムの任意である。ここではメッセージヘッダの中でも必須な項目についてまとめる。

メッセージの目的や構成にもよるが、簡潔に言ってしまえば、全てのメッセージヘッダを省略してしまっても通信できる場合はいくつも存在する。

例えば HTTP/1.0 における GET メソッドであれば、GET / HTTP/1.0 のみのリクエストメッセージでも多くのサーバはクライアントの意図を十分に理解することができる。しかしながら、加えることが望ましいものや、HTTP/1.1 になって必須となったヘッダも存在するためそれを以下にまとめる。
第 3 章 HTTP (HYPER TEXT TRANSFER PROTOCOL)

Host ヘッダ

Host ヘッダは、HTTP/1.1において唯一の必須ヘッダであり、その対象はリクエストメッセージである。仮想ホストを認めていないオリジンサーバや、仮想ホスト情報を含んだ絶対URLがRequest-URIに指定されていた場合は必ず無視される値である。しかし全てのHTTP/1.1サーバに対して、Hostヘッダの足りないHTTP/1.1のリクエストメッセージを不正パケットとして400ステータスコードを返さなければならないとしており、必要な場合にも、値が空のHostヘッダ（Host:）を付加しておかなければならない。

Connection ヘッダ

HTTP/1.0 からHTTP/1.1への大きな変更点のひとつであるコネクション維持の仕様により、HTTP/1.1では何も指定しないとコネクションを維持するとみなされ、通信が終わってもサーバが接続を持続しようとしてしまう。そのためHTTP/1.0ではデフォルトの動作であったトランザクション後の切断処理を、close パラメータを指定した Connection ヘッダ（Connection: close）を最後のトランザクションに加えることによって、明示的に指示してやらなくてはならない。加えなかった場合は、サーバが設定されたタイムアウトの時間まで接続を維持したまま待機することになる。

メッセージボディ長の決定

HTTP/1.1では接続の維持がデフォルトの動作として決められているため、HTTP/1.0で有効な方法である、メッセージボディ長がわからないメッセージの終了位置を通信相手からの接続の切断によって知る手法が使えない。（ここでメッセージボディのないものはボディ長が0と判断できることに注意する）

そのため、メッセージボディのあるHTTPメッセージにはメッセージボディ長の指定が必須である。HTTP/1.1においてメッセージボディ長の指定の方式は2種類ある。HTTP/1.0から存在するContent-Length ヘッダでデータのバイト数を指定する方法と、暗号化による安全な転送ができ、転送前にデータ長が決まらなくてもよいという特長を持つchunkedエンコーディングを用いるという方法である。chunkedエンコーディングは汎用ヘッダのTransfer-Encodingヘッダにおいてchunkedを指定することによって使用を明示する。（Transfer-Encoding: chunked）chunkedエンコーディングは全てのHTTP/1.1を用いたアプリケーションに必須な機能としているが、相手がHTTP/1.1に準じていると明確にわかる場合を除いてContent-Length ヘッダの指定を義務付けてある。どちらかの方法で長さが決まらない場合は、サーバは400ステータスコードを用いてエラーを通知する。特にContent-Lengthを要求する場合は411（length required）が返される。Content-Lengthヘッダとchunkedエンコーディングの両方が解釈できた場合は，
3.4 HTTP トランザクションの例

HTTP は状態に依存しないシンプルなプロトコルである。HTTP を用いた通信では、まずクライアントがサーバに接続する。その後クライアントは 1 回のリクエストを発行し、それに対してサーバが 1 回のレスポンスを返すことで 1 つのトランザクションが終了する。HTTP/1.0 では指定がなければここで切断処理が行なわれ、HTTP/1.1 であれば全てのトランザクションが終わった後にヘッダの指示によって切断処理が行なわれる。

ここで、代表的なメソッドである GET と、本論文で使用する POST による HTTP トランザクションについて例をあげて説明する。

3.4.1 GET メソッド

GET メソッドは、クライアントがサーバ上のデータを受け取るためのメソッドである。主に Web ページを構成する、テキスト形式で記述された HTML ファイルをサーバから取得する際に使用される。GET メソッドにより、index.html を受け取るトランザクションの例を示す。

1. まず、クライアントはサーバの指定されたポートまたはポート 80 に接続する。

   接続の確立後、リクエストラインを送信する。

   GET /index.html HTTP/1.1

2. 次に必要に応じてメッセージヘッダを送信する。

   Host:
   Connection: close
3. 最後にメッセージヘッダの終わりを示す空行 (CRLF) を送信する。

   CR LF

   ここでリクエストメッセージの送信が終わり，次にサーバからのレスポンスが返される。

4. まずはステータスラインが送られてくる。

   HTTP/1.1  200  OK

5. それにメッセージヘッダ，メッセージボディ (index.html) が続く。

   Date: Sunday, 21-Dec-04 20:25:00 GMT
   Connection: close
   Server: Apache/2.0
   Content-Type: text/html
   Content-Length: 254
   CR LF
   - - - - - -

メッセージボディ
   - - - - - -

こうしてクライアントは，サーバからデータを受け取る。

6. 最後にサーバとの接続を切断して GET のトランザクションが終了する。
第3章 HTTP (HYPER TEXT TRANSFER PROTOCOL)

3.4.2 POST メソッド

POST メソッドは、クライアントがサーバの指定した URL にデータを渡すためのメソッドである。クライアントがサーバ上の /cgi-bin/a.cgi に b.txt を渡すようなトランザクションの例を示す。

1. クライアントがサーバとの接続を確立する。接続の確立後、リクエストラインを送る。

   POST /cgi-bin/a.cgi HTTP/1.1

2. メッセージヘッダとその終了を示す空行 (CRLF) を送る。

   Host:  
   Connection: close  
   Content-Type: text/plain  
   Content-Length: 143  
   CR LF

3. それに続いてデータ (b.txt) を送る。ここで注意しなくてはならないことは、有効なデータの長さは Content-Length で指定した範囲であることである。

   つまり、Content-Length: 6 とした上で、データは abcdef と送ると、サーバの URL 先は abcdef までしか受け取らない。

   ここまででリクエストが終了である。次にサーバからレスポンスが返ってくる。

4. ステータスライン、そしてそれに続いてメッセージヘッダが返ってくる。

   HTTP/1.1 200 OK  
   Date: Sunday, 21-Dec-04 20:25:00 GMT  
   Connection: close  
   Server: Apache/2.0  
   CR LF

5. 最後に接続を切断し、POST トランザクションは終了する。
第 4 章

SIP を利用した通信での問題と解決手法

4.1 SIP を利用した IP 電話における問題

SIP を利用する通信や、他の IP 電話を実現するプロトコルには、有名な共通の問題が存在する。その問題は、それらのプロトコルを用いた上で NAT 環境を実現するときに発生するものである。一般にそれらのプロトコルは、そのままの状態では NAT ようなアドレス体系の異なるエリア同士で通信をすることはできない。なぜなら、アドレス体系が変わる部分では NAT 等の仕組みが通信データのアドレスを変換している。しかし SIP 等のプロトコルの中にも通信相手を特定するためのアドレス情報が組み込まれており、従来の NAT の仕組みではその情報まで書き換えることができないためである。この問題は一般的に NAT 越え問題と呼ばれ、現在それを解決する NAT 越えに関する研究が様々な組織で行なわれている。

4.1.1 NAT 越えに関する技術

まず、SIP を用いた通信における NAT 越えの技術をいくつか紹介する。NAT 越え問題を解決するアプローチには様々なものがある。それらのアプローチは、それぞれ仕組みだけでなく実装が必要な箇所やそれぞれの実装における負荷の大きさ、そして機能の豊富さなどが大きく異なり、実装する環境によってどれを採用するべきかは一概には決まることができない。ここでは、それらのアプローチについて代表的ないくつかの手法を取りあげる。

UPnP

UPnP (Universal Plug and Play) は、UPnP Forum で規定された規格で、PC や AV 機器などの家庭内の様々な機器をネットワークを通じて接続し、相互に機能を提供しうるために作り出された技術である。UPnP を用いると、プライベートネットワーク内のホストは UPnP 対応のルータに対して要求を出すことにより、ルータの持つパブリックな IP アドレスの割り当てやポート番号のマッピングを指示することができる。
SIP による通信において、UPnP を用いて NAT 越えを行なうためには、ルータとホストの両方が UPnP に対応しなくてはならない。しかし、現在では UPnP に対応したルータは数多く存在するが、UA やプロキシなどの SIP アプライケーションで UPnP に対応したものは非常に少ない。

STUN

STUN (Simple Traversal of UDP through NATs) は、アプリケーションがバブリックネットワークとの間の NAT やファイアウォールの存在とそのタイプを知ることを可能にする軽量なプロトコルである。STUN を用いることで、アプリケーションに対してネットワーク内のホストに割り当てたバブリックな IP アドレスやポート番号を通知することができる。

STUN を用いて SIP による通信を実現するためには、バブリックネットワーク内に STUN サーバが必要となり、プライベートネットワーク内のホストでも STUN に対応する必要がある。さらに、NAT の機能にも制限があり、NAT の内の同一 IP アドレスからのパケットを送出する際に外側で同じ IP アドレスやポート番号を用いる必要がある。

ALG

ALG は、NAT ルータを通過する SIP メッセージを解析して IP ヘッダおよび TCP/UDP ヘッダだけでなく、メッセージ内のアドレス情報についても変換を行なうことで NAT 越えを行なう方法である。

ALG を用いて NAT 越えを行なうために必要な実装は、NAT ルーター箇所である。つまり、UA やプロキシでは特別な処理を一切行なう必要はない。これにより、ALG の実装箇所に負荷が大きくかかってしまい、ルータに高い処理能力が求められる。しかし UA などのアプリケーションに新たな機能を実装する必要が無く、既存のシステムに組み込む際には最も現実的な方法であるといえる。

トンネリング

トンネリングとは、アドレス空間の異なるネットワークや通過するパケットに制限のあるマシンを通過する際に、IP ヘッダや様々なプロトコルのヘッダをパケットに付加することによって、中身の内容にかかわらずデータを転送する手法のことである。その実装の方法や役割は様々である。近年では IPv6 のパケットを 2 点間でやりとりする際に、IPv6 対応ネットワークの間にある IPv6 非対応ネットワークを IPv6 のパケットに IPv4 のヘッダを付加することによって通じるが、IPv4 over IPv6 トンネリングなどが多くのルータに実装され広く使われている。

トンネリングは目的に応じて様々な実装が可能であり、その実装方法によって機能や制限も様々
である。ただ一つ共通なことは、パケットヘッダを付加したり取り除いたりする機能をトンネルの入力口と出力口にあたる2箇所に実装しなければならないことである。現在のインターネットでは、NATやファイアウォールの普及によってアドレス空間の異なるネットワークや、パケットを一定のルールで制限する環境が増えている。そのため、必要な通信をトンネリングの手法を用いて行なうとする方法がいくつか出てきている。本研究では、SIPによる通信における問題を解決するために、SIPを用いた通信に特化したトンネリングの手法を提案する。

4.2 HTTP over SIP トンネリング

現在のインターネットでは多くの企業や家庭でセキュリティの確保やIPアドレス資源の枯渇防止のために、様々な対策がなされている。それらはSIPなどの電話サービスを提供するためのプロトコルの正常な動作に対する障壁となる。しかし、電話サービスを利用するためにはそれらの対策を止めてしまうことは、セキュリティの低下やコストの増大につながってしまうため非現実的である。例えば、それまでNATによって内部のネットワークを提供してきた企業が、IP電話サービスの利用のためにNATの使用を停止する場合を考える。まず電話サービスをうける全てのマシンにグローバルIPを付与するために膨大なコスト増が発生する。次に、NATによる簡易ファイアウォールの効果がなくなることで必要なファイアウォールの設定が複雑になり管理者の負担が大きく増してしまう。さらに、ファイアウォールでは防ぎきれない外部からのアクセスに対する対策を全てのマシンに対して施さなくてはならない。これは小さな企業にとって想定しても非現実的であるし、まして大きな企業では到底実現することはできない。そのため、SIPなどのインターネットを利用した電話サービスを提供するために、ファイアウォールやNATを利用した環境下において動作する仕組みを確立することは非常に重要なことである。

第4.1.1節でも述べたように、SIPを様々なネットワーク環境下で利用するための技術は数多く存在し、今後もそれらの利用や新しい技術の利用のためにルータだけでなくSIPのUAやプロキシなどのSIPアプリケーションにも新たな機能が付加されていくことが予想される。しかし、それらを利用する組織ごとにネットワークの規模や構造は大きく異なり、必要となる機能は様々である。また、従来のNAT越えの技術では、ファイアウォールなどNAT以外の通信制限要素が存在するネットワークにおいて、何の変更も無しにはサービスを利用することはできない。本研究ではHTTPトンネリングを利用し、NAT環境でSIPを利用するための様々な技術と組み合わせることで様々なネットワーク環境に対してより柔軟な解決策を提供する方法を提案する。
4.2.1 HTTP トンネリング

本研究の主な HTTP トンネリングとは，送りたいパケットをひとつずつデータと見立て，そのデータを HTTP プロトコルによって転送するトンネリング手法である．この方法を用いると，組織の内部ネットワークを HTTP プロキシやアプリケーションプロトコルまで判断する機能なファイアウォールで保護されたネットワークの内部と外部でデータをやり取りできる．

なぜなら，HTTP は多くのユーザが利用するプロトコルであり，ほとんどの組織において HTTP による外部への接続を完全に遮断してしまうことはないためである．
一般的な企業のネットワークは，外部からの不正なアクセスやウイルスなどの危険から守るために，簡易ファイアウォールと呼ばれる NAT を含め，様々な機能を持つファイアウォールや，HTTP による通信を中継するプロキシサーバなどによって保護されている．そしてほとんどの場合，ネットワークの内部に新たなサービスを稼動しようとした際には，いくつかのネットワークの設定を改定しなくてはならない．このことは，ネットワーク管理者に負担をかけることとなり，同時に設定のミスなどで今まで正常に稼動していた内部ネットワークを破壊してしまう危険も伴う．これは，NAT 越え問題を解決する多くの手法についても言えることである．もちろん，熟練したネットワーク管理者にとってこれらの作業はそれほど難しい作業ではないであろうが，作業に危険を伴うことはどうしても避けられない．さらに，全ての組織において熟練したネットワーク管理者が存在するわけではない．そのため IP 電話サービスには，これらの危険を軽減する手段の確立も必須事項となっていくと考えられる．

4.2.2 本研究の関連技術

HTTP トンネリングの技術も，その有用性や仕組み，利用する際の実装箇所など様々なものが作られている．次に，本研究の関連技術についていくつかを取り上げる．

**SoftEther**

SoftEther は，非常に有名な HTTP トンネリングを用いた VPN (Virtual Private Network)実現ソフトウェアである．VPN とは，通信インフラとしてインターネットを利用しながら，2点間でさえ専用線を用いたかのようにネットワークを構成する技術である．SoftEther では，2点間で HTTPS プロトコルを用いてネットルを張り，Ethernet 通信をエミュレートすることによって L2 (Layer 2: OSI 参照モデルにおけるデータリンク層) レベルでの VPN を実現する．また，実現における条件は SoftEther 本体がインストールできる OS であることと，その 2 点間で HTTPS プロトコルによる通信ができることであり，非常に実現可能性が高い．また，外部に SoftEther VPN Server を置くことにより，直接 HTTPS 通信のできない 2 点間でも，SoftEther VPN Server を介することによって VPN を張ることができるようになるなど，VPN を手軽に利
用するには非常に有用な手段である。さらに現在も開発は進んできており、SoftEtherを使用する際の解説書がいくつも発刊されていることからもわかるように、使いやすく非常に注目をあたえている技術である。しかし人気の影には裏が存在する。現在のネットワーク事情にあまりに強力な親和性があるため、ネットワーク管理者でない人でも組織のネットワーク内から簡単にトンネルが張れてしまう。つまり、企業のネットワーク管理者から許可をもらうことなく内部のマシンを外部と接続できてしまうという問題が発生する。これでは、セキュリティの甘い自宅ネットワークと企業の間でトンネルを張った場合、セキュリティの特性から企業全体のセキュリティのレベルが自宅のセキュリティレベルまで落ちてしまい危険である。SoftEtherの配布元では、活用の際はネットワーク管理者に相談したり、十分注意をして利用するようにしている。

skype

skypeは、スウェーデンのニクラス・センストロム、ジャヌス・フリースが開発した高音質のビアツービアソフトフォンである。skypeの技術仕様は正式には公開されていないが音質や使い勝手が良く、電話サービスを提供することができるソフトとして注目をあびている。

skypeは、起動時にインターネットにアクセスし、ネットワーク上にある他のskypeにアクセスして自分の情報を登録する。インターネット上に存在するノードの中に、なんらかの方法で選出されたスーパーノードと呼ばれるものが存在する。端末の情報は、集中サーバによって管理されるのではなく、インターネット上のスーパーノードによって保持される。また、NATやファイアウォールの内部からでもIP電話のサービスが受けられる。

このソフトを利用すると、容易に比較的高品質なIP電話サービスが様々なネットワーク環境下で利用できるようになる。しかし、仕様が公開されていないなどの理由から、ネットワーク管理者の許可を得ずにこのソフトを利用する人が内部に存在すると、SoftEtherと同様に管理者にとって管理がしにくい状態になってしまう。

4.3 本研究の提案手法

本研究で提案するシステムは、HTTPプロトコルを利用したトンネリングの技術をSIPを用いた音声通信に特化したものである。本研究のシステムを現存するNAT越えALGの技術と組み合わせることで、様々な機能を持つファイアウォールや、外部との通信を制限されたネットワークにおいてそれらの技術を容易に利用、管理できる。

本研究の技術を用いると、HTTPプロトコルでの通信のみを許したネットワーク内から、別のネットワークとのSIPを用いたデータ通信を行なうことができる。また、ALGなどの技術と組み合わせることで、パブリックネットワークとのSIPを用いた通信も可能にすることができる。これにより、管理が安全かつ容易で適応環境も柔軟なサービスを提供することができる。
さらに本研究では、通信内容を暗号化したVPNの途中に外部との接続手段を付加した環境で通信を効率的に行ならう手法についても提案する。

### 4.3.1 マシンの構成

本研究で用いるネットワークの基本構成は図4.1のようである。

![ネットワーク構成図]

**HTTP-Encapsulation Server**

SIPを用いた通信を構成するデータをHTTPプロトコルによってカプセル化して送り出すサーバである。これは通信を行いたいUAと同じネットワーク内に置かれ、同じネットワーク上であればNAT内やファイアウォールの内側でもよい。1つのマシンを専用に用意する必要はなく、同等の機能を他のSIPアプリケーションに組み込むことで容易に実現できる。

**HTTP-Uncapsulation Server**

HTTP-Encapsulation Serverによってカプセル化されたデータを受け取り、同じネットワーク内へデータを渡すサーバである。HTTP-Encapsulation ServerからHTTPプロトコルのアクセスが受けられる位置に置かれる必要がある。もちろん、外部との境界にあるマシンからのポートフォワーディングによって通信をすることもできる。

**HTTP/SIP proxy Server**

HTTP/SIP proxy Serverは図4.1には示されていないが、本研究とNAT越えの技術を組み合わせるために必要となるサーバである。HTTP-Uncapsulation ServerとHTTP-Encapsulation Serverの両方の機能、HTTPでカプセル化されたデータを中継するHTTP-proxyサーバの機能、
外部の異なるアドレス領域と通信を確立するための NAT 越えの技術が組み込まれている。この仕様については、第 6 章で記述する。

4.3.2 各構成要素の機能仕様

次に、各構成要素の仕様について説明する。

図 4.2: HTTP-Encapsulation Server 通信仕様

HTTP-Encapsulation Server

HTTP-Encapsulation Server は、同 LAN 内の SIP, RTP/RTCP のパケットをイーサネットフレーム（データリンク層）で受け取り、HTTP カプセルリングを施して HTTP-Uncapsulation Server へ送る。

つまり、HTTP-Encapsulation Server は SIP の UA から SIP や RTP/RTCP のデータを受け取り、HTTP-Uncapsulation Server との間では、HTTP プロトコルのリクエストメッセージを使ってデータをやり取りする。これにより HTTP プロトコルでデータをやり取りする通信過程において、NAT などの異なるアドレス空間への変換があっても、HTTP-proxy サーバが存在して代理に通信が行わなくても、HTTP-Uncapsulation Server との HTTP プロトコルによる通信が可能である限り通信は成立する。

SIP のパケットは、初期設定では UDP のポート 5060 を利用したものを、RTP/RTCP のパケットは SIP メッセージ内の SDP に記述された情報から IP アドレスとポート番号を取得し、その情報をテーブルに格納して流れてくるデータと比較し判断する。
第4章 SIPを利用した通信での問題と解決手法

HTTP カプセリング

本研究でHTTP カプセリングを施したデータ構造は図4.3のようにになる。

<table>
<thead>
<tr>
<th>HTTP Header</th>
<th>SIP-URI</th>
<th>IPパケット(SIP or RTP/RTCP)</th>
</tr>
</thead>
</table>

図4.3: HTTP カプセリングデータ構造

HTTP Header

HTTP カプセリングを施す際に用いる HTTP Header の内容は、その通信環境によって改善もしくは、何度か再試行することによって適切なものを選択するのが理想的であるが、本研究では基本構成を次のようにする。改行には CRLF を使用する。
また[HTTP データサイズ] には、HTTP Header 以下の全てのデータ長つまり HTTP メッセージボディ長を入る。

表4.1: HTTP Header

<table>
<thead>
<tr>
<th>POST /a.cgi HTTP/1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host:</td>
</tr>
<tr>
<td>Connection: keep-alive</td>
</tr>
<tr>
<td>Content-Length: [HTTP データサイズ]</td>
</tr>
<tr>
<td>CR LF</td>
</tr>
</tbody>
</table>

SIP-URI

SIP-URI には、SIP であれば sip: username@domain で構成される宛先、RTP/RTCPであれば RTP: IP アドレス で、それぞれに改行 (CRLF) ひとつを続けて代入する。この情報に用いて HTTP-Uncapsulation Server もしくは HTTP/SIP proxy Server は、相手の UA がどこに存在するのかを判断し、UA ヘデータを送信またはルーティングを行う。SIP データ部はカプセリングされた状態での通信ではその内容によらないため、暗号化されて送られてもかまわない。その際に宛先 UA の情報が中に隠されてしまうと、HTTP/SIP proxy Server でデータをルーティングする際にどこに送ってよいのかわからない。そのため暗号化を解く必要のないデータについても、その都度暗号化を解かなくてはならない。これはリアルタイム通信としては重大な問題である遅延につながる。暗号化された SIP データ部の外に UA の宛先情報を記述した SIP-URI を付加し UA の宛先の情報として利用することによって、遅延の問題を少しでも解決することができる。
HTTP-Uncapsulation Server

HTTP-Uncapsulation Server は、HTTP-Encapsulation Server から送られたカプセルングされたデータを解き、SIP-URI から宛先を判断し UA までデータを送る。若しくは宛先の方向へデータをルーティングする。UA が同じ LAN 内に存在するならば、HTTP-Uncapsulation Server は UA の宛先情報から MAC アドレスを調べ、その MAC アドレス宛にデータリンクフレームを送信する。

つまり、HTTP-Uncapsulation Server は HTTP-Encapsulation Server との間で HTTP プロトコルのリクエストメッセージを使ってデータのやり取りを行わない。UA へは SIP や RTP/RTCP のデータを送る。MAC アドレスは SIP の URI と IP アドレスと共にキャッシュテーブルに格納する。MAC アドレスを調べる時は、まずそのキャッシュテーブルを参照して一致するものがなければ arp プロトコルを使って MAC アドレスを調べる。

本研究では、HTTP-Encapsulation Server と HTTP-Uncapsulation Server の 2 つの機能を合わせて、HTTP over SIP Tunneling Server と呼ぶこととする。
第5章
実装と動作検証

5.1 HTTP over SIP Tunneling Server の実装

HTTP over SIP Tunneling Server を HTTP や SIP のプロトコルスタックに忠実に実装して動作検証を行うとすると、プログラム量や動作検証が非常に大掛かりなものとなってしまう。そのため本研究では、提案仕様の核となる SIP と RTP/RTCP を HTTP メッセージでカプセル化して転送する機能のみを実装し、その動作検証を行なった。HTTP メッセージは、第 4 章で設定した基本構成に則り作成する。

5.1.1 実験の目的

本研究において、SIP と RTP/RTCP を HTTP メッセージにのせて転送する機能を実装し動作検証を行う目的について述べる。

SIP を用いた IP 電話サービスを始め、リアルタイムにメディアをやり取りするアプリケーションにとって重要なことのひとつとして、遅延時間が短くなければならぬ。SIP を用いた IP 電話サービスでは、音声メディアの通信に RTP という UDP のプロトコルを用いている。UDP の特徴はデータの信頼性を犠牲にする代わりに転送の負荷を下げることである。これは電話サービスに置き換えると音質低下を犠牲にすることで遅延の抑制をしていることとなる。しかし、本研究の HTTP プロトコルによるトンネリングでは、データを HTTP メッセージでカプセル化する処理が付加され、さらに通信には TCP を用いることとなる。TCP は、UDP とは対照的に信頼性を重視するために様々な処理を行う。そのため、UDP と比較すると通信に遅延が発生しやすい。

IP 電話サービスでは、音声の時間軸に対する正確性を最も重視している。これはつまり、音質が一部聞き取れないほどに悪化してしまうと言葉は前後の文脈から推測可能であるが、音が欠けて音声到着のタイミングがずれてしまうと意味が取れなくなってしまう可能性が高いという検証結果による。そのため、IP 電話サービスを提供する RTP では遅れて届いた音声データは破
棄してしまう。本研究の実験では、遅延が原因となって音声データの破棄が大量に発生し、音質が著しく低下してしまうかを実験によって検証する。

5.1.2 実験の環境

次に、動作検証実験を行なうマシン構成を示す。（図 5.1）

![図 5.1: HTTP over SIP Tunneling Server 実験環境]

5.1.3 プログラムの仕様

HTTP over SIP Server に実装する機能は次の通りである。

1. HTTP-Encapsulation Server

   - 起動時に相手の HTTP over SIP Server の 80 番ポートにコネクションを張る
   - BPF (Berkeley Packet Filter) を用いて、イーサネットから UDP パケットをキャプチャリングする
   - 5060 ポートを利用していたら、SIP-URI を `sip: username@domain` と、それ以外であれば `RTP: IPAddress` として HTTP Header で SIP-URI, UDP パケットの順にデータをカプセルングする (username, domain, IPAddress には適切な値を代入)
   - HTTP-Uncapsulation Server にカプセルしたデータを送る
第5章 実装と動作検証

2. HTTP-Uncapsulation Server

- 起動時に80番ポートでコネクションをリスンする
- コネクションが張られ、データが送られてきたらContent-Lengthを参考にしてHTTPパケット1個分を受信
- SIP-URIを参照して宛先を取得する
- BPFを利用して、適切な宛先のMACアドレスにUDPパケット部を送る

また、サーバ間にipfwによるファイアウォールを設置し、SIPとRTP/RTCPのデータを通さないルールを設定した。これにより、ファイアウォールを隔てた通信環境を実現した。

5.1.4 実装の環境

以下に実装に関する詳細を示す。（表5.1）

表5.1: HTTP over SIP Tunneling Server実装環境

<table>
<thead>
<tr>
<th>ソース規模</th>
<th>639行</th>
</tr>
</thead>
<tbody>
<tr>
<td>仕様コンパイラ</td>
<td>gce 2.95.4</td>
</tr>
<tr>
<td>動作環境</td>
<td>FreeBSD 4.10-RELEASE</td>
</tr>
<tr>
<td>SIP UA</td>
<td>livedoor SIP フォン for PC</td>
</tr>
<tr>
<td>UA環境</td>
<td>Windows XP Home Edition</td>
</tr>
</tbody>
</table>

5.2 動作検証と性能評価

動作検証は、実際にUAを起動し通信することによって行なった。

5.2.1 動作検証

まず実験環境下における動作検証を行なった。UA同士の通信は呼制御、音声通信共に問題なく動作が確認できた。また、トンネルを張ったふたつのサーバ間では、SIPやRTP/RTCPのデータは一切流れておらず、HTTPメッセージによる通信のみが確認できた。本研究のシステムを用いてSIPやRTP/RTCPを通さないファイアウォールを隔てて、IP電話サービスが実現できることを示せた。

この結果は、本研究のシステムの有用性を示している。
5.2.2 性能評価

次に本研究のシステムの性能評価を行なった。性能評価は、音声品質の評価、音声の遅延時間の評価の2点で行なった。特に、遅延時間に関しては総務省で定められたIP電話の評価基準に則り、評価と考察を行なった。

音声品質の評価

まず、音声品質については音飛びやかすれなどは全く確認されず問題はなかった。音飛びやかすれが生じる原因としては、送信の遅延にムラがあること、つまりUDPパケットをTCPパケットとして送っていることにより発生する。この結果、本研究の実験におけるマシン構成では、UDPパケットをTCPパケットとして送信することによる大きな問題は発生しなかったことがわかる。

遅延時間の評価

次に、遅延についての測定を行なった。片方のUAから音声を飛ばして相手のUAで届いたタイミングに合わせて次の音声を飛ばすことを繰り返し、60秒の間にカウントできる回数を測定する。

次に遅延時間の測定結果を示す。（表5.2）

<table>
<thead>
<tr>
<th>接続状況</th>
<th>遅延時間 (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>直接接続</td>
<td>0.909</td>
</tr>
<tr>
<td>実験環境</td>
<td>0.933</td>
</tr>
</tbody>
</table>

直接接続とは、UA同士をクロスケーブルで接続した状態での遅延時間である。どちらもUA間片道の音声遅延時間である。

音声の遅延は、実験環境下と直接接続とのネットワーク構成の違いによる少々の遅延も含むが、ほぼHTTPカプセルリングにかかる時間によると考えられる。2つの環境下での音声遅延の差は0.024secで直接接続時の遅延時間の2.64%と小さく、会話が成り立たなくなるほどの遅延は明らかに発生していないことがわかる。

総務省により規定されたIP電話の通信品質基準は厳しく、最高品質で固定電話と同等の品質と規定されるクラスAでは遅延は0.100sec未満、050番号の割り当てが受けられる最低の基準であるクラスCでも0.400sec未満と規定されている。本研究のシステムを用いることで新たに
生じた遅延時間は 0.024sec であった．本研究のマシンの構成要素は最低限のものである．つまりマシンの性能を考えないとすると，どんなに短くてもクラス A で規定された遅延時間の約２５％が本システムの遅延によって生じてしまう．しかし，一般のユーザがサービスを利用するネットワーク環境はより複雑なものとなることが予想される．本サービスを市場に投入することを考慮した場合，サービス全体で生じる遅延時間は，本システムの遅延時間に UA による音声エンコーディング及び音声デコーディングの時間を加えたものとなる．本研究が付加的要素であることを探まえても，実際にサービスとして利用するためには，トンネリングの手法に遅延の軽減を行う改善が必要である．
第6章

本研究を用いた応用例と今後の課題

6.1 本研究を用いた応用例

本研究の技術はSIPを用いたVoIP通信を様々なネットワーク環境に柔軟に対応できるようにするためのものであり、その適用例も様々な例が考えられる。その中でも本研究では実装において、HTTP/SIP proxy Server、現存するネットワークと高い親和性を持たせるのに非常に有効な手段である。ここでは応用例のひとつの例として、HTTP/SIP proxy Serverを利用してIP電話サービスの実現例を取り上げる。

HTTP/SIP proxy Serverの仕様

まずは、HTTP/SIP proxy Serverの仕様について述べる。

HTTP/SIP proxy Serverは、HTTP-Encapsulation Serverの機能とHTTP-Uncapsulation ServerとしてALGによるNAT越えの機能を実装しており、またHTTPトンネリングのルーティングも行っている。外部との通信においてALGによるNAT越えの機能を実装し、内部とのHTTPトンネルによるVPNの接続を複数拠点と張ることができる。

図6.1: HTTP/SIP proxy Serverの機能概要図
次に動作順序にのっとり、仕様を箇条書きにまとめた。

- 起動時 80番ポートでコネクションをリスンする
- ALGを通った外部からの SIP, RTP/RTCP パケットと HTTP トンネリングを利用して送られてきたパケットを受ける
- Server 本体内部に SIP の宛先と IP アドレス、そして宛先が存在するネットワークへのルーティング情報を格納しておく
- ALGを通った外部からの SIP と RTP/RTCP パケットであれば、その宛先情報に従い HTTP トンネルを利用して宛先にパケットを送る
- HTTP トンネルからのパケットであれば、その SIP-URI 情報から他拠点の VPN 内にある宛先か外部の宛先かを判断し、VPN 内の宛先であればカプセルを解かずに宛先のある VPN へとデータを転送する。外部の宛先であればカプセルを解き、ALGを通じて外部へパケットの中身を送る
6.1.1 HTTP/SIP proxy Server を用いた応用例

前節でまとめた HTTP-SIP proxy Server を用いた実際のネットワークにおける応用例を図 6.2 に示す。

これは複数の支店を持つ企業を想定したネットワーク構成である。支店 A では、LAN からグローバルネットワークへの出口にはファイアウォールが設置されており、メールを除いて HTTP-proxy サーバを介さない通信を許していない。支店 B では、アプリケーションレベルでプロトコルを判断する L7 ファイアウォールが設置されており、同様にメールと HTTP プロトコル以外のパケットの通過を許可しない。

それぞれの LAN 内に HTTP-Encapsulation Server と HTTP-Uncapsulation Server が設置され、全ての SIP フォンはそれらのサーバを介して通信を行なう。

本店には HTTP/SIP proxy Server が置かれ、支店 A、支店 B、本店全ての IP 電話がそこを通じて外部のネットワークと通信する。

まず、それぞれの支店から LAN 外への SIP、RTP/RTCP 通信時に、データがカプセル化されて本店に置かれた HTTP/SIP proxy Server へ HTTP トンネルを利用して送られる。そこで SIP-URI を参照して宛先が外部にあるのか、それとも支店 A、支店 B、本店のどこかの LAN 内にあるのかを判断する。宛先が外部にあるときはカプセルを解いて ALG による NAT 越えの処理を施されて外部に送信される。宛先がいずれかの LAN 内にあるものは、カプセルを解かずにその宛先へと HTTP トンネルを利用して送られる。

このようにして、HTTP/SIP proxy Server を用いて支店 A、支店 B、本店で VPN を利用した IP 電話サービスと IP 電話による外部との通信を実現する。
第6章 本研究を用いた応用例と今後の課題

6.2 結論と課題

6.2.1 結論

本研究では、SIPを用いたVoIPセッションをHTTPトンネリングによりシリアル化して転送することで、ファイアウォールを越えてIP電話サービスを提供できることを示した。また、サービスの実現可能性についての品質の問題点の指摘やサービス形態については提案をした。

本研究により、SIPを利用したIP電話サービスの様々なネットワーク構成における問題が解決されることが示された。これにより、IP電話サービスの採用に障壁となっていたコストの増大やネットワーク再構築の必要性、ネットワーク管理の負荷増大が解決され、IP電話サービスの普及を促進することができる。

6.2.2 今後の課題

今後の課題として、本研究を組み込んだNAT越えシステムの実現性の検証、規模性、遅延の軽減が挙げられる。

本研究を組み込んだNAT越えシステムの実現性については、おおまかな仕様については本研究の第6.1節で提案をした。そのため、実装をした上で動作検証を行うが課題となる。

また、そこで問題となってくると思われることが規模性についての問題である。本研究における実験では、サーバ1対1、UAもそれぞれのLAN内に1体のみと最小限の規模で行なった。本研究の提案では、ALGに加え全ての機能をHTTP/SIPproxyServerに実装する設定となっているため、規模が大きくなればなるほど負荷が集中して大きくなってしまう。そのため、機能の効率的な分散やその状態での遅延を最小限に抑えるための手法が必要となる。

そして最後に一番重要な課題が遅延の軽減である。ここで言う遅延とは、音声遅延だけでなくパケットの転送時の処理にかかる遅延である。本研究のシステムではUDPパケットをTCPパケットとして転送する。本研究の実験ではTCPパケットでの通信区間がスイッチ1つを挟んだだけの短い区間であった。そのためTCPが原因となる音質の悪化は感じられなかった。しかし、規模が大きくなりTCPでの通信区間が伸びたり、通信区間内でパケットの転送が発生し通信効率が悪くなってしまうことは充分に考えられる。その場合に音質の悪化を避けることはできない。一般的にはIP電話網に専用線を設けることで、QoS(Quality of Service)、つまりIP電話サービスの音声品質や音声遅延の軽減、を確保する手法が用いられるが、この方法ではコストの増大を避けることができなくなってしまう。さらに、専用線を設けずに通信手段を確保するという本研究の特長を打ち消してしまうこととなる。本研究ではセキュリティの確保のためという目的よりも、ネットワーク環境への柔軟な適応を重点においているため、この問題を解決するならかの改善策を提案する必要がある。
そこで今後は，本研究を総合的なサービスに実装し動作検証を行なうと共に，UDP パケットを TCP パケットとして送る際の改善策について検討する．
謝辞

本学士論文の作成にあたり日頃より御指導を頂いた早稲田大学理工学部の後藤滋樹教授に深く感謝致します。そして、本学士論文のテーマについて助言や御指導を頂いた株式会社イメージパートナーの田中信顕氏に深く感謝致します。最後に、研究を進める上で貴重なアドバイスや御指導を頂いた後藤研究室の匂坂岳志氏、多大なる御協力を頂いた後藤研究室の諸氏に感謝致します。
参考文献


