＜報文＞

VoIPパケットモニタプログラムの開発

九州職業能力開発大学校
附属川内職業能力開発短期大学校　　古　屋　　保

Development of a VoIP packet monitor program

Tamotsu FURUYA

要約　Voice over Internet Protocol（VoIP）とは、IPネットワーク上で音声データ通信を行う技術である。本研究では、PCapライブラリとGTK+ライブラリを使用し、IP電話と同一セグメントのPC上で、リアルタイムに動作するVoIPパケット専用モニタプログラムを開発した。このプログラムは、電話接続の遷移状態、および通話中の音圧レベルの変化をリアルタイムに確認することが可能である。IPネットワーク上の通信状況を直感的にイメージすることにより、ネットワーク実習などの教育現場においては、受講者に対し、VoIPの仕組み（プロトコルおよびパケット等）を目に見える形で提供することができ、ネットワーキング技術の能力開発に効果があると考えられる。実際に、専門課程の学生に総合制作実習の課題として与えたところ、開発においては、ネットワーキングと音声の両方の知識、併せてGUI、マルチスレッドなど複数のプログラミング手法を組み合わせるという技術が必要であり、学生の能力開発において非常に効果があった。

Ⅰ　はじめに

IP電話とは、Voice over Internet Protocol（以下、VoIP）の技術を利用した音声電話サービスのことをいう。つまり、既存の一般電話網（PSTN：Public Switched Telephone Network）ではなく、IP網いわゆるインターネット技術を利用した電話のことである。IP電話は、通話データをパケット化することで、WWWや電子メールと同様に、インターネット回線を利用し音声電話サービスを提供するものである。すなわち、電子メールやWWWなど、主にアスキー文字ベースでやり取りされるサービスと同様に、IPネットワークを介して電話のサービスが受けられるのである。

一方、学生の能力開発現場において、そのような環境下で、VoIPの仕組みを教える場合、その取り掛かりとして、ネットワーク上に音声データが流れる概念が受講者に伝わりにくい。通信状況をイメージできるものがあれば、VoIPの仕組みについてより理解が深まると考えられる。

そこで、ネットワーク上を流れるパケットを、キャプチャし解析するソフトウエアおよびハードウエアには、機能および価格も共にさまざまなものがあるが、リアルタイムにパケットデータの流れをグラフィカルにイメージできるもので、かつ複雑なフィルタ設定操作を必要とせず、手軽に利用できるものが必要であると考えた。

本研究では、IP電話における音声データによる通信のやり取りが、IPネットワーク上でどのような形で行われているか、実際にIPネットワーク上に流れているVoIPパケットをキャプチャし、直感的に通信状況がイメージできるプログラムを作成したので、ここに紹介する。

Ⅱ　VoIP技術についての概要

1．IP電話のしくみについて

IP電話の実現においては、まずは、VoIPゲートウェ

論文受付日　H17.4.19

イ装置（以下、VoIP GW）に電話機を接続しておき、その装置により、電話機から発せられた話し声などのアナログ音声信号を、ディジタル信号に変換し、それをIPパケット化する。IPパケット化された情報は、他のIPパケットと同様、パケットに含まれている宛先IPアドレス情報を頼りに目的の場所まで転送される。相手側VoIP GWに届けられたIPパケットは、送信とは逆の処理でアナログ音声信号化し、相手側電話機に届けられる。

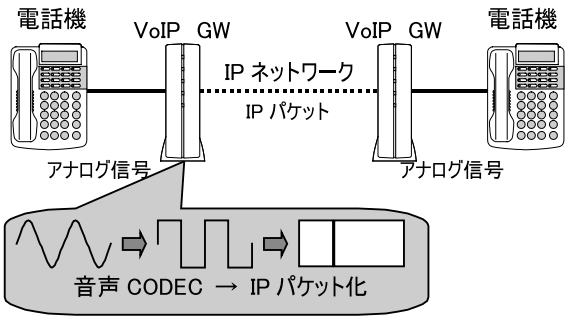

図1　IP電話のしくみ

一般電話同様、電話番号をダイヤルしたり、接続先機器名称などを指定したりして相手を「呼（call）」し、通話のための接続を確立する必要がある。このような制御を呼制御（シグナリング）という。すなわち、電話においては、音声情報の他に、シグナリング情報という2種類の情報のやり取りが行われていることになる。

図2　電話間をやり取りする2つの情報（パス）

2．セッション制御（シグナリング）プロトコル

シグナリングの実現には、セッション情報プロトコルを使用する。IP電話で主に利用されているのは、H.323とSIP（Session Initiation Protocol）であるが、ここではH.323を取り上げる。

H.323は単一のプロトコルではなく、パケットネットワーク上でテレビ会議を実現させるためのプロトコルスイートであり、1996年からITU-T（国際電気通信連合－電気通信標準化部門）で標準化が進められている[1]。H.323は既存のPSTNとの接続性を考慮して、PSTNのシグナリングの考え方を基に設定されている。古くから使われており実績も多いが、PSTNの手順が複雑であるように、H.323もSIPと比較すると複雑なプロトコルになっている。

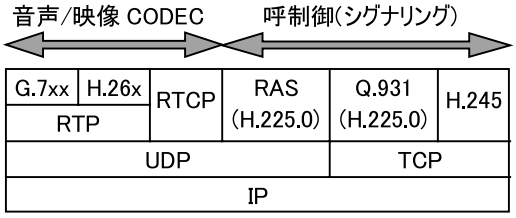

図3　H.323プロトコルスイート

H.323は音声情報をやり取りするG.7xx、RTP/RTCPと、シグナリング制御を行うH.225.0と端末の能力交渉（符号化方式などの確認）をおこなうH.245などのプロトコル群で構成されている。

Ⅲ　システム概要

使用するVoIPパケットの解析システムは、IP電話環境（電話機＋VoIP GW）2組と、同一LAN上のパケット解析用PCで構成する。VoIP GWには市販のIP電話実験キットを使用し、パケット解析用PCには、Linuxの稼動するPC/AT互換機を使用した。

1．VoIP実験キット（HT1070-SVP）

IP電話実験キットHT1070-SVP（以下、キット）（図4）は、株式会社アットマークテクノと梅沢無線電機株式会社で開発・販売されているキットで、Linuxが動作するCPUボードArmadilloと、電話制御ボード（HT1071-U00）で構成されている。CPUボードには、LAN（10BASE-T）ポートを装備し、電話制御ボードには、一般に市販されているアナログ電話機をそのまま接続して利用できるようになっている。加えて、電源(5V,12V)と、イーサネットの環境があれば、IP電話実験用の環境が構築できるようになっている。

図４　IP電話実験キット（HT1070-SVP）

表１に、使用したIP電話実験キットの主な仕様を掲載しておく。

表１　IP電話実験キットの主な仕様 (2)

CPU	CirrusLogic CS89712CB （ARM720Tコア採用）74MHz
メモリ	SDRAM 32MB / Flash 4MB
I/Oバス	16bit PC/104準拠
通信プロトコル	H.323
音声コーデック	G.711 u-Law
電話ポート	１ポート（RJ-11）
LANポート	１ポート（RJ-45）10BASE-T
電源	5V, 12V
その他	CompactFlash™（TrueIDE）対応, シリアル×2,パラレル×1,LCD

コンパクトフラッシュ™には、あらかじめLinux（kernel 2.4.16）と、オープンソースのプロトコルスタックであるOpenH323ライブラリを利用したVoIPアプリケーションが組み込まれており、サービスとして自動起動されるようになっている。

２．パケットモニタプログラム

パケット解析用PCにて、IP電話の通信状況を監視・解析を行うためのパケットモニタプログラムを作成した。パケット解析を行う際のツールとして、市販およびフリーのソフトウエアが幾つかあるが、そのほとんどがフレームキャプチャデータを一旦ファイルに保存し、その後データを解析するスタイルになっている。

そこで、本取り組みにおいては、VoIPのみを対象にパケット解析処理を行うことで、リアルタイムかつグラフィカルに通信状況を表現できると考え、VoIP専用のパケットモニタプログラムを、一から開発することにした。

パケットの取り込みにおいては、パケットキャプチャライブラリであるPCapライブラリ（libpcap）を、ボタン等を配置するGUI画面構成にはGTK+ツールキットをそれぞれ使用し、VoIPパケット解析専用のソフトウエアを開発した。なお、開発環境はLinux上のGCC（GCC-2.89）を利用した。

Ⅳ　パケットモニタプログラムの作成

パケットモニタプログラムを完成させる工程として、まず手始めに、キャプチャされたパケットデータから、必要なセッション情報および音声データ情報をフィルタリングし、キャラクタレベルで表示するプログラムを作成した。その後、GUI実装を施し、パケットキャプチャ動作およびデータ表示中に生じる種々の問題を解決するために、マルチスレッド処理を実装した。プログラムは図５のようなモジュールで構成されている。

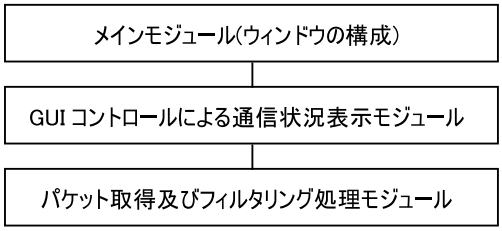

図５　モジュール構成図

１．PCapライブラリによるパケットキャプチャ

ネットワークインターフェース（デバイス）が取得できるすべてのパケットを、データとして取り込むことのできる関数を集めたものが、このPCapライブラリ（libpcap）で、UNIXのツールとして良く利用されるtcpdumpもこのライブラリを使用している。

このPCapライブラリ関数を使用して、キャプチャしたパケットデータ（イーサネットフレーム）は、IPデータグラムを包括しているIPパケットであり、その中の、TCPセグメント、およびUDPデータグラムを解析することで、現在IP電話同士で取り交わされているサービスを確認することができる。

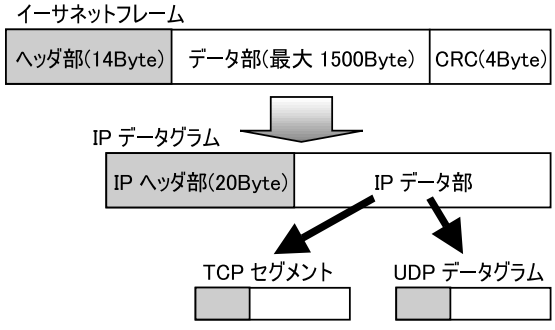

図６　イーサネットフレームとIPパケット

2．セッション制御中におけるパケットデータ解析

セッション制御時にキャプチャしたパケットサンプルを図7に示す。

```
0000: 00 04 14 12 34 56 00 04 - 14 12 34 57 08 00 45 00
0010: 00 9d 38 ab 40 00 40 06 - 1d fc ac 10 45 ca ac 10
0020: 45 c9 06 b8 04 1c a1 2e - 2b 81 a0 ed cf b3 80 18
0030: 19 20 87 96 00 00 01 01 - 08 0a 00 10 13 40 00 10
0040: 12 f4 03 00 00 69 08 02 - 8f 0a 01 28 06 67 75 65
0050: 73 74 00 7e 00 55 05 23 - 80 06 00 08 91 4a 00 04
0060: 22 c0 09 00 00 3d 1e 48 - 54 31 30 37 30 2d 53 56
0070: 50 20 50 72 6f 6a 65 63 - 74 20 48 54 31 30 37 30
0080: 2d 53 56 50 00 00 06 31 - 2e 30 2e 31 00 00 00 d0
0090: c0 11 00 9a ed f1 f8 ba - c0 d3 11 80 d2 00 04 14
00a0: 40 02 7e 01 00 01 00 02 - 80 01 80
```

図7　セッション制御時のキャプチャデータ

最初の14バイトはイーサネットヘッダ部で、その次の20バイトはIPヘッダ部である。IPヘッダ部の10バイト目（パケット全体では0x17バイト目）が0x06であることから上位プロトコルはTCPであることが確認できる。0x22バイト目からのTCPヘッダ部は図8のような構造になっている。

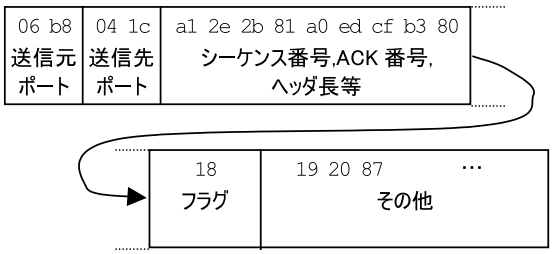

図8　TCPヘッダ部

TCPヘッダ部を解析することによって、上位サービスがわかる。送信元ポート番号0x06b8（1720）はH.225.0（Q.931）プロトコルを意味し、現在セッション制御中のパケットであることが確認できる。TCPセグメントデータ部（0x46バイト目以降）は、H.225.0（Q.931）チャネルになる。

H.225.0チャネルの5バイト目（イーサネットフレーム先頭から0x4Aバイト目）の値は、Q.931のメッセージタイプで、図14の例では0x01となっているが、メッセージタイプの1は“ALERTING”を意味する。ALERTINGは、着端末側から発端末側へ送信されるメッセージで、このALERTINGメッセージを受信した発端末は、ユーザ（電話をかけている人）に、相手呼び出し中であることを音で通知する。

他にも、SETUP,CALL PROCEEDINGなどのメッセージがあり、発信から、通話までに図9のようにメッセージがやり取りされている。

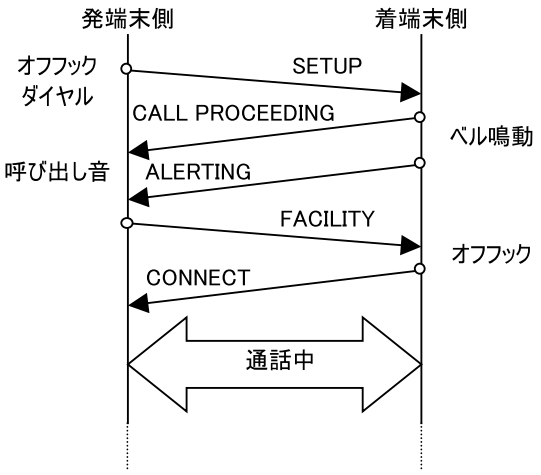

図9　セッション制御（通話までのプロセス）(3)

最終的には、発端末側がCONNECTメッセージを受信することで、呼が確立する。その後、H.245メディアコントロールチャネルを利用して、発端末と着端末との間でネゴシエーション（能力交渉）が行われる。ここでは、使用する符号化方式（コーデック）の決定、および制御の主従関係などを決定する。その後、通話が行われる。

3．通話中におけるパケットデータ解析

つづいて、通話中にキャプチャしたパケットサンプルを図10に示す。

```
0000 00 04 14 40 02 7e 00 04 - 14 40 02 7d 08 00 45 10
0010 01 18 00 00 40 00 40 11 - 56 11 ac 10 45 ca ac 10
0020 45 c9 13 88 13 9e 01 04 - 6c 63 80 00 ec 76 00 01
0030 43 70 d6 c6 bf 33 f3 f4 - f7 fb 7f 78 78 7a 79 77
0040 73 75 77 79 7c 7f fb fb - fb fe fc f9 fb 7e 7e 7d
0050 7c 7d 7f fd fd 7d 7a 7a - 7c fc f5 f4 fa 79 76 7e
0060 fa f9 fb 7f 7f fd fe 7e - 7b 7a 7d 7e 7c 7e fd 7e
0070 7e fd fd 7b 78 79 7e fb - f7 f9 ff 7e 7f fd fc fd
0080 7f 7e 7b 79 7b 7d fe fc - fe fe ff 7f 7f 7f fe fe
0090 fa fa 7f 7e 7e ff fe fe - fd fc fe 7a 75 75 79 7c
00a0 78 75 76 7c fa f3 f5 f9 - f7 fb fe fc fb fc fc 7d
00b0 79 7e fd fd fe 7e 7d 7e - 7f 7b 7a 7a 7a 7d fb fc
00c0 fd fb f9 fc 7d 79 77 77 - 79 7d fc fd 7e 7d 7e fe
00d0 fd 7f 7d 7f fc f9 f2 f1 - f9 fb f8 f7 7f 78 77 76
00e0 77 78 7c 7d 7f fe 7e 7b - 7a 7c fe 7f 7f 7f fe fb
00f0 fa 7f 7c 7e fd fe 7e 7c - 7e fd fb fc 7f 7d 7f 7f
0100 fe fe 7f fe fc fd 7d 7c - 7e fc fb 7d 78 79 7e fb
0110 fd 7e fc fb fb fb 7f 7e - 7f fc fb fc 7d 78 7a 7e
0120 7d 7d 7e 7f 7c 79
```

図10　通話中のキャプチャデータ

セッション制御時と同様、最初の14バイトはイーサネットヘッダ部で、その次の20バイトはIPヘッダ部である。0x17バイト目が0x11であることから上位プロトコルはUDPであることが確認できる。

UDPヘッダ部は図11のような構造になっている。

13 88	13 9e	01 04	6c 63
送信元ポート	送信先ポート	UDPパケット長	Checksum

図11　UDPヘッダ部

UDPには、TCPには備わっているエラーチェック機能やエラー訂正機能、およびシーケンス番号が実装されていない。そこでUDPによるリアルタイム通信にはRTPプロトコルが使用される。

RTPヘッダ部は図12のような構造になっている。

80 00	ec 76	00 01 43 70	d6 c6 bf 33
バージョン，フラグ情報	シーケンス番号	タイムスタンプ	SSRC 識別子

図12　RTPヘッダ部

UDPデータグラムの、RTPヘッダ部を引いた残り0x36バイト目から始まる“f3 f4 f7 fb 7f 78 78…”の240バイト分の領域はペイロードと呼ばれ、音声コーデック情報が格納されている。キットの音声コーデックはITU-T G.711を用いていることから、音声データは、サンプリングレートが8kHz、ビットレートが64kbpsでコーデックされている。その１バイト分のデータは125μs（１/8000s）分の音声符号化データであり、これが１パケットあたり240バイトあることから、１パケット分で30ms分の音声符号化データを送出していることになる。

今回、ジッタ等の影響を考えず、１パケット分の音声データの絶対値を平均することで、30ms間隔のキャプチャごとに音圧レベル値を割り出すことにした。

4．GTK+ツールキットによるGUIの実装

パケットキャプチャを行うモジュールと、GUIを実装モジュールとを連携することにより、パケットモニタリングの様子を、グラフィカルに動作させることにした。

今回、開発環境がLinuxであったため、X Window上で動作するXアプリケーションを作成した。Xアプリケーションを作成するために、XLibのラッパーであるGTK+ツールキットを使用した。これはボタン、テキストエントリなどのGUI部品（ウィジェット）が用意され、ウィンドウ上にその部品を配置して利用することができる。

図13　作成したパケットモニタプログラム

ウィンドウ上のStartボタンによりパケットキャプチャを開始し、Stopボタンで停止する。上部２つのエントリボックス（一行入力用ボックス）には、VoIP GWのIPアドレスとセッション情報におけるメッセージを、ドローイングエリアには音圧レベルの増減の様子を表示させ、下部のテキストボックス（複数行入力用ボックス）には、パケットデータの16進ダンプを表示させている。これらを左右にそれぞれ設けて、発信元および着信先それぞれから送出されたパケットデータを表示するようにした。

5．マルチスレッド処理の実装

パケットキャプチャ処理中にも、Stopボタンやメニュー操作などを有効にするために、パケットキャプチャ処理を別スレッドで動作させることにした。スレッド処理には、Linuxにおいて標準で用意されているThreadライブラリを使用した。PThreadライブラリは、POSIX 1003.1c-1995に準拠したスレッドライブリである(4)。スレッドを新たに作成するには、PThreadライブラリ関数の１つであるpthread_create 関数を使用すれば良い。

実際には、スレッド開始と同時にパケットキャプチャを開始させ、処理をループさせておく。メインモジュールとパケットキャプチャモジュールとで互いに共有するフラグ変数を設け、そのフラグがTRUEであればループを維持し、FALSEであればループを終了し、パケットキャプチャを停止させる。キャプチャ停止と同時に関数がリターンされるので、結果的にその時点でスレッドが終了するという仕組みにしている。

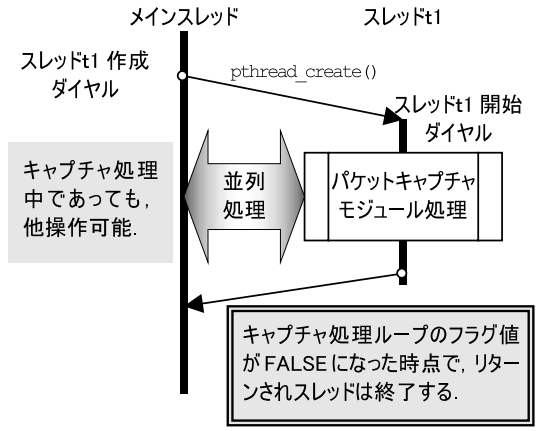

図14　キャプチャ処理を別スレッドで動作させる

実際にGTK+アプリケーション（Xアプリケーション）にスレッド処理を摘要する場合、最初にg_thread_init関数を呼び出して、GLibをスレッドセーフになるよう初期化する必要がある。(5)

上述のようにスレッドを摘要し動作させてみたところ、ボタン、メニューなどのGUI操作には問題ないが、16進ダンプを表示するテキストボックスの更新処理と、音圧レベルを表示するエントリボックスの更新処理とが、互いに足を引っ張り合うことで、表示に遅延が生じた。そこで、さらにスレッドを追加し、テキストボックス（16進ダンプ表示）の更新処理を別スレッドにしたところ、遅延は解消された。

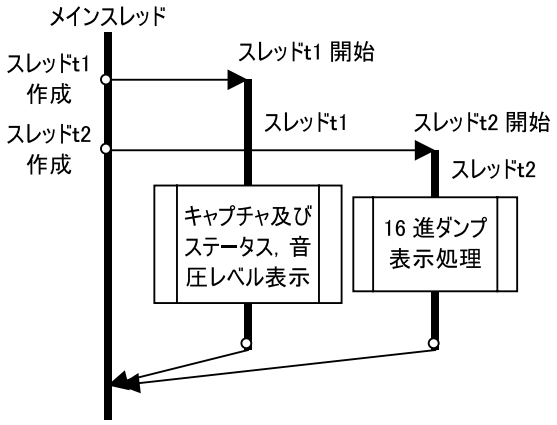

図15　最終的なスレッドモデル

16進ダンプ表示もスレッド（スレッドt2）上で処理をループさせているが、表示の更新については、キャプチャ処理スレッド（スレッドt1）と共有するフラグ変数にてタイミングを計っている。ただし、複数のスレッドでフラグ変数を読み書きしているので、スレッド同士で同期を取る必要があった。今回はPThreadのmutexという仕組みを利用することにより、デッドロックを解消している。(4)

V　まとめ

実際に作成し動作させてみた結果、以下のようなことが確認できた。

(1) 電話機のダイヤル操作から、相手呼び出し、接続（通話）、切断までのシグナリングに関するステータス情報について、リアルタイムに表示させることが可能である。
(2) 通話時において、UDPの音声パケットデータを検出し、音圧レベルの変化について、グラフィカルかつリアルタイムに表示させることが可能である。
(3) シグナリング情報および音声パケットデータの16進ダンプを表示させ、後から通信履歴を確認することが可能である。
(4) POSIXスレッド機能を利用し、パケット取り込み中であっても、メニューやボタンクリックなどのGUIの操作が可能である。

IP電話の通信状況をリアルタイムで表現するという研究目的は達成することができた。しかし、このパケットモニタプログラムが、数十人で行う集合実習という現場で使用に耐えうるものかどうかは、実際にトラフィックの大きいネットワーク上でテストする必要がある。

今回、本研究を専門課程の総合制作実習における課題として取り上げ、開発・実験にかかわった学生は、2004年2月に行われた第2回九州ブロックポリテックビジョンの学生展示および学生発表の部に参加し、発表の部で優秀賞という評価を得ることができた。

今後は、解析用PC上で、符号データを音声信号にデコードさせ、どの程度音声が再現できるか、音声コーデック処理について理解を深める。これはIP電話が盗聴も可能であることにつながるが、学生に対しては、IP電話だけでなく、ネットワーク上でやり取りされるデータは盗み見される危険性があることを認識させ、改めて暗号化等によるセキュリティの重要性を伝えることができる良い材料になると考えている。

謝　辞

本研究を進めるにあたり、開発・実験にかかわった当校第18期卒業生の塩月雅也君と東條裕輔君に深謝する。

[参考文献]

(1) 川西素晴、OPEN DESIGN 2003年3月号掲載記事（基礎から理解するVoIP技術)、CQ出版社、pp.9-55
(2) Armadillo公式サイト
http://armadillo.atmark-techno.com/
(3) 村上健一郎、Interface 2003年6月号掲載記事(TCP/IPの現在とVoIP技術の全貌)、CQ出版社、pp.47-123
(4) 新城　靖、スレッドプログラミングについて、
http://www.coins.tsukuba.ac.jp/~yas/classes/dsys-1998/1999-01-19/pthread.html
(5) 日本GNOMEユーザ会、文書一覧（GTK+FAQ等)、http://www.gnome.gr.jp/docs/
(6) 小高知宏、基礎からわかるTCP/IP アナライザ作成とパケット解析、オーム社
(7) Paul Simoneau、都丸敬介訳、TCP/IPプロトコル徹底解析、日経BP社
(8) たなかひろゆき、GTK+入門 第2版、ソフトバンクパブリッシング
(9) 梅沢無線電機株式会社、Armadillo HT1070 hardware manual Version 1.11
(10) 梅沢無線電機株式会社、HT1070-SVP VoIP実験セット USER'S MANUAL Version 1.02