Philippine Engineering Journal (2000) XXI (1): 50-58

TEXT AND BINARY DATA TRANSFER USING POSTFIX AND UUCP OVER AN HF RADIO LINK

Ivy Cabeza, Anne Margrette Caccam,
Mia Antonio, Robert de la Cruz and Roderick Durmiendo
Communications Engineering Division
Advanced Science and Technology Institute
Department of Science and Technology
Philippines

ABSTRACT

This paper discusses the first ever implementation of UNIX-TO-UNIX COPY (UUCP) and POSTFIX over an HF radio network. It allows reliable text and binary data transfer between computer terminals several hundreds of kilometers apart. The system was developed for use by PAGASA (Philippine Atmospheric, Geophysical, and Astronomical Services Administration) for collection of weather data from remote weather stations distributed around the country to a central server located in Diliman, Quezon City.

I. Introduction

Developing a reliable and timely way of exchanging information is one of the vital needs for national development. There are various means for exchanging information such as telephones, cellular phones, television, and the Internet. All of these are commonly used and proved to be useful for everyday communication. However, there are specialized communication needs wherein the data has to be collected from several remote stations to a central server several hundreds of kilometers away. Data traffic is not very heavy but is periodic over twenty-four hours a day and seven days per week.

Maintaining a dedicated telephone line would prove to be quite costly due to long distance charges and it would need one line per remote station. Another problem would be the possible non-existence of a telephone network near the remote terminal. This is usually the case when its location is on top of a mountain or in a small island. Our system introduces an alternative way of exchanging data between such remote stations.

Our system is HF radio-based. It makes installation to remote site much simpler. It is composed of a HF/SSB (High Frequency/Single Side Band) radio, a Pactor II modem, and a PC running a Linux operating system. The use of Linux makes the system very reliable because it is a stable operating system. Also, it comes complete with the Postfix and UUCP utilities including source codes. It is also cheap because it is a shareware. The Pactor II modem bridges the PC with the radio. It converts the RS-232 signals to audio signals, which the radio could transmit. It also uses advanced digital signal processing techniques that reduce communication errors. The HF/SSB radio enables long distance communication up to several hundred kilometers. See Figure 1 for an overview of the system.

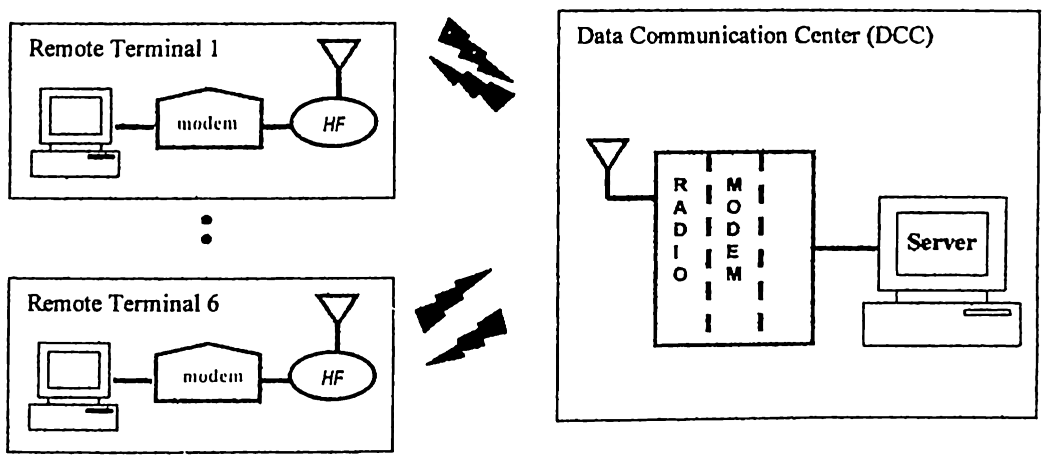

Figure 1 The HF Messaging System

For our initial implementation, the system design is for one central Data Communication Center and six remote stations, each containing a personal computer, a Pactor II modem and an HF/SSB radio. In the next section, we will give a more in-depth discussion of Linux, UUCP, Postfix, Pactor II Modem and the HF/SSB Radio. This will be followed by a more detailed description of the system integration, recommendations and then, the conclusion.

Linux

Linux is low-cost but highly stable operating system that runs on IBM-compatible PC's[1]. It works like UNIX but it boasts its efficiency and faster performance over the other conventional UNIX systems. It provides a complete system, including the user interface called X Windows System, TCP/IP, and the Emacs editor among others. It conforms to the POSIX standard user and programming interfaces. It provides true-multitasking capabilities, offers virtual memory and proper memory management.

The Linux package comes with its free source codes so users can customize their installation according to their preferences. It can be downloaded from various Internet sites. Programs for Linux are continuously being developed and released as a freeware, making them accessible to both developers and users.

UUCP

Unix-to-Unix Copy or UUCP was developed to provide simple networking protocol for UNIX systems[2]. Although it is commonly used as an email gateway, UUCP enables remote access of non-networked machines via modem connection. It can also be used for Usenet news and other similar services that do not require a dedicated connection. UUCP also has security features that will protect the systems from unauthorized accesses by setting up the security system properly. Linux is compatible with several UUCP versions. Two of the most used are Taylor UUCP (which is available with the Red Hat distributions) and HoneyDanBer (HDB) UUCP. The two are compatible except for their configuration and installation procedures.

Taylor UUCP was chosen over HDB because of the availability of its source codes and the entire files needed to make a UUCP connection[3]. It contains uucico (UUCP file transfer daemon), uusched (lists the jobs queued for UUCP and UUX), uuxqt (UUCP execution daemon), uux (remote command execution via UUCP), as well as uuchk, (helpful for checking the configuration setup of the system) and uuconv (used for conversion of the configuration files from one type to another). This package supports several protocols including two bi-directional protocols. Taylor UUCP was written by Ian Lance Taylor (ian@airs.com). It is covered by the GNU Public License and the code is available as a freeware, which can be found at file://ftp.gnu.org/pub/gnu/uucp/uucp-1.06.1.tar.gz. It is also included in the newer versions of Linux RedHat.

The good thing about Taylor UUCP is the availability of its source code. It comes as a freeware and contains almost all of its complement programs. It also uses less CPU time than other UUCP packages. Taylor UUCP also provides customizable chat scripts allowing modem parameters to be adjusted on a per system basis. It even specifies a failure string for chat scripts, which lets the modem disconnect if it returns a busy signal. Taylor UUCP supports a new protocol for rapid data transfer in both directions at once. One can also restrict file transfers by size based on the time of day and who placed the call.

UUCP is not the sole program that handles remote copy. As UUCP connects to a remote system, a series of steps are being executed by the uucico (UUCP Call In/Call Out) process. The calling system is usually called the master while the remote system is its slave. When invoked by UUCP, uucico checks the *sys* file to know if the remote system exists. It then refers to the *port* and *dial* files to start the physical connection. After a successful chat session and dialing, uucico will go back to *sys* file to initialize the chat script for logging in the remote system. If the login process is successful, the remote system will invoke its own uucico process and the two processes will establish a handshake. Finally, the uucico will continue the transfer that was queued. When the session is completed, the master will check whether the slave is finished with its own connection. Some protocols enable the two systems to change roles, thus enabling the slave to be the master and starts its own tranfer without the need to log in.

A connection between two systems can be established through either a direct serial port or a modem. In a direct connection, UUCP can be used as a simple network protocol for file transfer. To enable this feature, the *port* file should indicate DIRECT as its port type. The chat script in the *dial* file should be empty since there is no modem to be set up. But if a modem connection is to be used, chat scripts should be handled with care to successfully initialize the modem. Also, telephone numbers should be indicated to access lines.

For a successful login, the UUCP administrator should create a UUCP account for every remote access. To do this, one may use the *useradd* command or simply edit the */etc/passwd* file and then create a password via *passwd* command. The convention is to start every user with 'U'. This is logical for the *mgetty* process. Once a user logs in with a username starting with 'U', the *uucico* will be invoked. This can be configured in the *mgetty* local files.

Besides setting the access times in the *sys* file, UUCP could be set via *cron*. *Cron* is a daemon to execute scheduled command. For scheduled *uucico* execution, the *crontab* file may be edited. The second, minute, day, week and month can be specified.

To set the security of UUCP access, permissions of the UUCP configuration files should be properly set. All UUCP configuration files should be owned and be under the group of uucp rather than the root. A strict password implementation should be used so only those authorized can run the UUCP program. Also, file permissions should be set. Write-read access should only be given to the uucp owner. Other user accesses may be manually edited to satisfy their respective privileges. Critical files, such as the sys and passwd files, should be highly protected.

By default, the home directory of UUCP is uucppublic. Different logins could be used for a single remote system to allow different file and program access permissions. The control of commands that remote systems could execute should also be monitored. This can be configured in the *sys* file. Constant inspection of log files should also be implemented to monitor abused and unauthorized accesses.

Postfix (formerly Vmailer)

POSTFIX is a program conceptualized, designed and implemented by *Wietse Zweitze Venema*. Its goal is to provide an alternative to the widely used SENDMAIL program, which is responsible for most of the emails delivered on the Internet. In the UNIX operating system, SENDMAIL is the default program used in transporting (receiving, sending, etc.) mail from one user to another whether locally or using other remote connections. An attempt to replace SENDMAIL with POSTFIX has been made because of the apparent complexity of the SENDMAIL program, add to that, its configuration files are difficult to modify or tune particularly when dealing with a wide range of services. POSTFIX then proves to provide parallel utilities and services without much complex file configurations and system manipulations. Generally, the option to adapt the POSTFIX program for this project is due to the consideration of its features.

Performance-wise, POSTFIX can boast of being three times as fast as its nearest competitor[4]. A desktop PC running Postfix can receive and deliver a million different messages per day, still maintaining minimal creation of processes and reduced file system without sacrificing reliability. POSTFIX is also designed to be compatible with the custom sendmail program so there is no need to worry about migration problems. Their only difference is that, POSTFIX does not use the sendmail.cf file, which makes it easy to administer. When the local system encounters problems as regards disk space or memory, the POSTFIX program has the ability to shutdown itself (i.e. its currently running processes) in order to avoid worsening the problem. It is also deemed flexible since it is composed of several small programs that perform particular tasks. These tasks include multiple transport capability, which simply means that POSTFIX can operate in different environments or use relay hosts such as the Internet, UUCP and X.400, without the need for virtual domains. Another good feature of POSTFIX is that, in its most common case, adding support for a virtual domain requires change to only a single Postfix lookup table. Other mailers usually need multiple levels of aliasing or redirection to achieve the same result.

For this project, POSTFIX was used to actually replace the SENDMAIL program provided in the typical installation of Linux RedHat 6.x, and as a compatible mail support when integrated with the UUCP program. Since it is also a freeware, it is acquired by simply downloading from the site www.postfix.org. The developer merely has to follow a set of procedures to build, install, configure and implement POSTFIX.

There are only three main (mandatory) configuration files to be edited for POSTFIX, main.cf, master.cf and transport. All of them are found in /etc/postfix.

main.cf is the global POSTFIX configuration file. It lists a subset of more than a hundred parameters with sensible default values. In most cases, you only need to change a few of these parameters (i.e. hostname, domainname, daemon_directory) and you're half-ready to use the POSTFIX mail system.

master.cf file describes how a mailer component program should be run. Here, we have to indicate the service process, in our case, it's UUCP, then the transport type that is set to 'unix' since we use unix-domain sockets. We also include here the max_proc data that talks of the maximum number of processes that may execute the POSTFIX services simultaneously.

Lastly, the transport file specifies the mapping from domain hierarchies to message delivery transports and or relay hosts. It actually generates the table for sending mail to specific sites via UUCP.

Pactor II Modem

The Pactor II protocol provides seamless data transfer over the HF/SSB medium[5]. It uses Digital Signal Processing (DSP) techniques such as DPSK (Differential Phase Shift Keying) modulation scheme, date error correction and online data compression. The DPSK Modulation scheme leads to a very narrow spectrum, practically independent of the data rate. A narrow spectrum means power is not spread out over a wide frequency range. In our case, power is concentrated on two tones defined by raised cosine signals with a spacing of 200 Hz ("shift") in parallel. The complete signal has a spectrum of 450 Hz wide at –50 dB, which is relatively narrow as compared to other modulation scheme. There are four possible modes for this modulation scheme which differ in the number of possible phase changes: DBPSK or Differential Binary Phase Shift Keying has 2 possible phase differences between the two tones, QPSK or Quadrature Phase Shift Keying has 4 possible phase differences, 8-DPSK has 8 and 16-DPSK has 16. The different modes have trade-offs in bit rate, signal-to-noise ratio (SNR) requirements, and bit redundancy requirement for error coding (Figure 2). From the table, we could see that the throughput increases as the number of allowable phase increases but the signal-to-noise ratio requirement also increases. The modem detects and automatically sets the optimal mode in a given condition.

DPSK Mode	Allowable Phase Difference	Throughput (Bit/Sec)	Signal-to-Noise Requirement	**Redundancy** Character
DBPSK	2	100	Noisy	½
QPSK	4	200	Not very Noisy	½
8-DPSK	8	400	Quite Clean	2/3
16-DSPK	16	700	Very Clean	7/8

Figure 2: DPSK Modes

The modem also guarantees 100 percent error-free transmission. The assumption is that radio transmission is a noisy medium and there will always be errors present in transmission. The algorithm compensates for this by introducing redundant characters in the original signal. Single bit error in a given block of data can usually be recovered through these redundant characters.

It also implements online data compression. It uses both Huffman coding and pseudo Markov coding. Huffman coding is a compression algorithm for sending plaintext messages. It takes advantage of the statistical frequency distribution of the letters in plain language in order to compress data. For example, the letter 'e' which is the most frequently used letter is encoded using only one bit while the letters 'z' and 'x' are encoded using seven bits. On the average, it uses 4.7 bits per character that is much less than the required 7 bits per character in normal ASCII notation. Markov coding also uses statistical frequency distribution but it is applied to a pair of characters. However, its disadvantage is that there are so many character pairs. Pseudo Markov coding is a hybrid of Huffman and Markov coding. It notes only the 16 most commonly used character pair and then uses ordinary Huffman coding of the other characters. It was proven to be more efficient than normal Huffman compression.

The Pactor II modem is also capable of determining the highest common level between two Pactor modems during link start up. This ensures that the optimal modulation mode is used. The auto tuning capability detects frequency-offset differences between the transceivers and adjusts the offset frequency either to the left or to the right so that the center frequencies of both are the same in order to ensure accurate demodulation.

HF/SSB Radio

The JSB-171P model HF/SSB radio from JRC (Japan Radio Corporation) is an all solid-state 150 Watts Peak Envelope Power radiotelephone which covers the frequency range of 1.6 to 29.9999 MHz for transmission and 0.1 to 29.9999 MHz for reception in 100Hz steps[6].

The radios are capable of five transmission modes: J3E-USB (Upper Sideband), J3E-LSB (Lower Sideband), H3E (AM Compatible), A1A (Morse telegraph) and F1B (Narrow-band direct printing). We will be using the J3E-USB mode which the most ideal for data communication.

When a carrier is amplitude-modulated by a single sine wave, the resulting signal consists of three frequencies: the original carrier f_c, the upper sideband frequency ($f_c + f_m$), and the lower sideband frequency ($f_c - f_m$) [7]. This is an automatic consequence of the amplitude-modulation process, and will always happen unless steps are taken to prevent it. However, the upper side band (USB) and the lower side band (LSB) are mirror images of each other. The carrier component also remains constant in amplitude and frequency and therefore conveys no information. From this, we could see that all the information could be carried by only one sideband or a Single Side Band (SSB) and all the other components could just be filtered out or attenuated. This is important because this method saves up to 90% of the power consumption of the radio. This method was not very popular then because it requires a very accurate circuitry. However, recent advances have made this technology the choice for applications where bandwidth efficiency is at a premium.

In the 4 through 22 MHz HF (High Frequency) band, there are two types of radio frequency propagation; ground waves and sky waves. Ground waves follow the curvature of the earth slightly beyond the horizon (20 to 75 miles). The primary propagation of radio waves in the HF range is by means of the sky wave. The sky wave travels toward the ionospheric layers surrounding the earth and is refracted back toward the earth. Because these electrically charged ionospheric layers are 50 to 250 miles above the earth, the radio wave returns to the earth's surface hundreds of miles away. Due to this phenomenon, we are able to communicate great distances with HF communications, depending upon the particular characteristics of the ionospheric layers as a function of the time of day, season and sun spot activity.

Because the propagation range of a frequency is dependent of the time of day, there is a need to change radio frequency several times during a 24-hour period. The choices are 4 MHz, 8 MHz, 12 MHz, and 16 MHz bands. Each of these frequencies is.efficient at different times of the day. A radio protocol was designed in such a way as to choose the optimal frequency for use in the radio network.

The control of the radio is via the optional RS-232 serial port at the back. Through this port, all functions such as changing the squelch level, the volume of the transceiver, and the receive and transmit frequencies of the radio can be accessed using software commands. Every fixed time interval, a frequency-check algorithm is run. Every remote station has a specified response time wherein it expects a poll signal from the central station. At timeout, it automatically changes to a different frequency. A polling program is run at the central station every hour, which scans through all the remote stations in all available frequencies in order to determine the frequency of a certain remote station.

System Integration

The system has four network layers (Figure 3). The Postfix layer, UUCP layer, the Pactor II modem layer and the HF/SSB layer. The Postfix layer is the uppermost layer. It is used for sending and receiving e-mails. The UUCP layer is for transferring binary and text files. The Pactor II modem converts the RS-232 signals to audio. The HF/SSB layer transmits and receives audio data. Each layer is actually talking to its counterpart layer in the counterpart station. For example, the UUCP layer in the Central Server talks to the UUCP layer of the Remote terminal. However, communication passes through all the layers below it except for the HF/SSB, the lowest layer, which talks directly to the corresponding radio. System integration is therefore a step-by-step process. The radio-to-radio link first, then modem-to-modem, then UUCP-to-UUCP, then Postfix-to-Postfix finally.

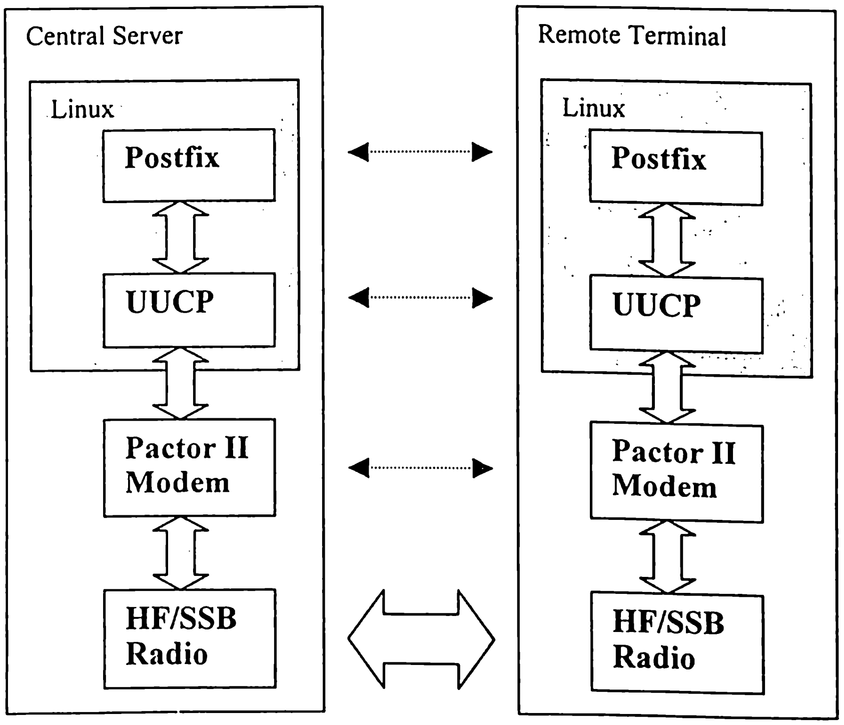

Figure 3: Network Layers

The radio-to-radio link is established by setting the transmit and receive frequencies of both radios. For our set-up, the server and the client radios are tuned to the same transmit and receive frequencies[6]. Check that both radios are in the same transmission mode (J3E-USB) or mode 1 of the radio. Also, adjust the squelch to a value, as low as possible while ensuring that the noise or static level is almost inaudible.

After establishing radio connection, we now establish modem-to-modem connection. The Pactor II modems are connected to the PC via the RS-232 serial port and to the radio via the audio output and audio input pins[5]. Initiating a connection to another Pactor II modem requires a serial port terminal emulation program. The modem is software controlled and the commands are entered from this terminal.

A unique callsign is assigned to every modem that is up to eight characters long. For example, we will call the central station modem as "central" and the remote station modem as "remote". Establishing a connection from one modem to another is done by entering a command format "Connect *callsign*". When "central" wants to connect to "remote", it initiates it by the "Connect *remote*" command. The call command will then be transmitted by "central" and heard by all Pactor II modems on that particular frequency and then matches the called callsign with its own. If it is not its callsign, the call is ignored, else it answers back and proceeds to negotiate the most optimal transmission mode with the calling modem. A "Connected to..." message is then received at the initiator side and data transmission could then proceed between modems.

Connect UUCP-to-UUCP over an HF link proved to be quite difficult. Several problems were encountered mainly due to the incompatibility of the UUCP protocol and the Pactor II modem. UUCP was originally designed as a protocol to transfer data between two terminals linked via a telephone modem or a TCP/IP connection. The connect script which initiates communication between two telephone modems usually involve waiting for a dial tone then dialing the particular telephone number of the other party through the use of ATDT command set. Radio modems are different. Each modem is identified by a particular callsign. The command set of the radio modems is different, too.

There are also some special characters[8], such as [Ctrl-H] and [Ctrl-R], normally accepted by telephone modems that are not accepted by the Pactor modems because they are part of its command set. These special characters are an integral part of the UUCP protocol, especially [ctrl-R] because it is used to signal an end-of-string argument.

Synchronization problems between the two UUCP terminals were also encountered. The UUCP protocol detects linefeed character in its connect script. Each linefeed is equivalent to the [Enter] keyboard command. However, extra line feed characters were being sent by the radio modems. Every time the terminal sends a linefeed character, it is echoed by the radio modem to let the user know that it received a linefeed. However, when that same linefeed gets to the other modem, another echo is sent. Two are echoed for every linefeed sent.

Because of these problems, revisions were made on the original UUCP. The UUCP source code was modified and re-compiled. The UUCP program we used is a special version designed especially for use over the radio modem link. We call this version as HF-UUCP.

Once the UUCP-to-UUCP connection is working, implementation of Postfix as the mail protocol was straightforward. Changes were made on both the server and the client terminals as specified in the Postfix configuration files. There were minimal installation problems. UUCP and Postfix allow e-mail capability and file transfer of both text and binary data between terminals.

Conclusion and Recommendations

The system was designed for the specifications of the data communication needs of PAGASA. Their basic requirement was for the system to be able to get an updated report from each of around six remote stations once every three hours. The data size ranges from 500 to 4000 bytes. Now, these are not difficult to satisfy. For the case of a noisy medium, the radio modems can transmit at around 100 bits/second. A 4000-byte file could be transferred in around 5 minutes. The total amount of time to get data from six remote stations is just around 30 minutes assuming the worst case HF noise and largest data size. The system has more than 2 hours of idle time in between communications.

For now, we are running the system to poll all the remote stations once every hour. Additional remote stations could still be accommodated given the large idle time of the system. The idle time could also be used for transfer of data other than the required weather information. The ability of UUCP and Postfix to transfer binary and text files could allow operators to exchange document and image files.

The central Linux server could also be directly connected to the Internet. This capability will give operator e-mail over an HF radio link. This will give meteorologists the opportunity to exchange e-mail messages with colleagues from other countries.

The set-up is also easily reconfigurable to suit the demands of other agencies or offices in need of a low-cost, high-speed link to interconnect remote offices. This is most useful in inter-island offices. The initial cost of setting up a radio link is high but still cheaper as compared to the long-term cost of maintaining a dedicated telephone line for communication purposes.

Acknowledgements:

The authors would like to acknowledge Engr. Denis Villorente and Ian Lance Taylor. Mr. Villorente, from ASTI, helped in the modem configuration and testing. Mr. Taylor, the author of the original Taylor UUCP, gave advice though e-mail (and through the UUCP mailing list) on how to modify the UUCP protocol to make it run over an HF link.

References:

[1] What is Linux. http://www.whatis.com/linux.htm, assistance was provided by Gordon Blair and Christian Kirsch. November 8, 1999.
[2] Taylor, Ian Lance. Taylor UUCP Version 1.06. http://www.airs.com/ian/uucp-doc/.
[3] Purcell, John and Amanda Robinson (1997). Chapter 12: Managing Taylor UUCP. Linux Encyclopedia. Aurora, CO: Workgroup Solutions, Inc., 1995.
[4] Venema, Wietse Zweitze. MANUAL and README pages of POSTFIX program. http://www.postfix.org.
[5] PTCII-e Manual. Special Communications System, Inc..
[6] Text Handbook for SSB Radiotelephone Model JSB-171P. Japan Radio Co., Ltd.
[7] Kennedy, George. Electronic Communication Sytems. McGraw-Hill Book Company, 1988.
[8] Jourdain, Robert. Programmer's Problem Solv er. Peter Norton's Programming Library, 1992.